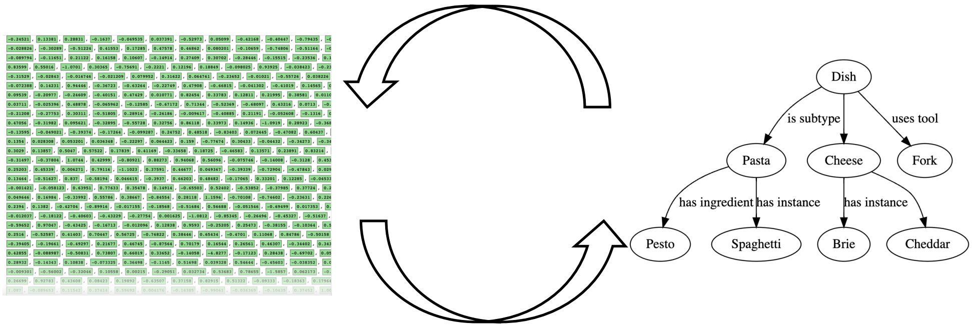
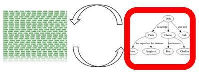
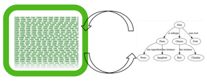


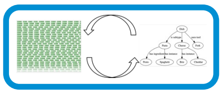
Interaction between knowledge graphs and large language models

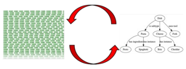
Felix Sasaki, Chief Expert Knowledge Graphs and Semantic Technologies, SAP AI Unit
Conference “KI in Bibliotheken - Neue Wege mit großen Sprachmodellen?” (“AI in libraries – new approaches with large language models?”)
December 8th 2023
Public

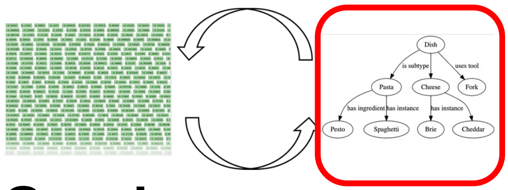
Overview

Knowledge Graphs (KGs) 

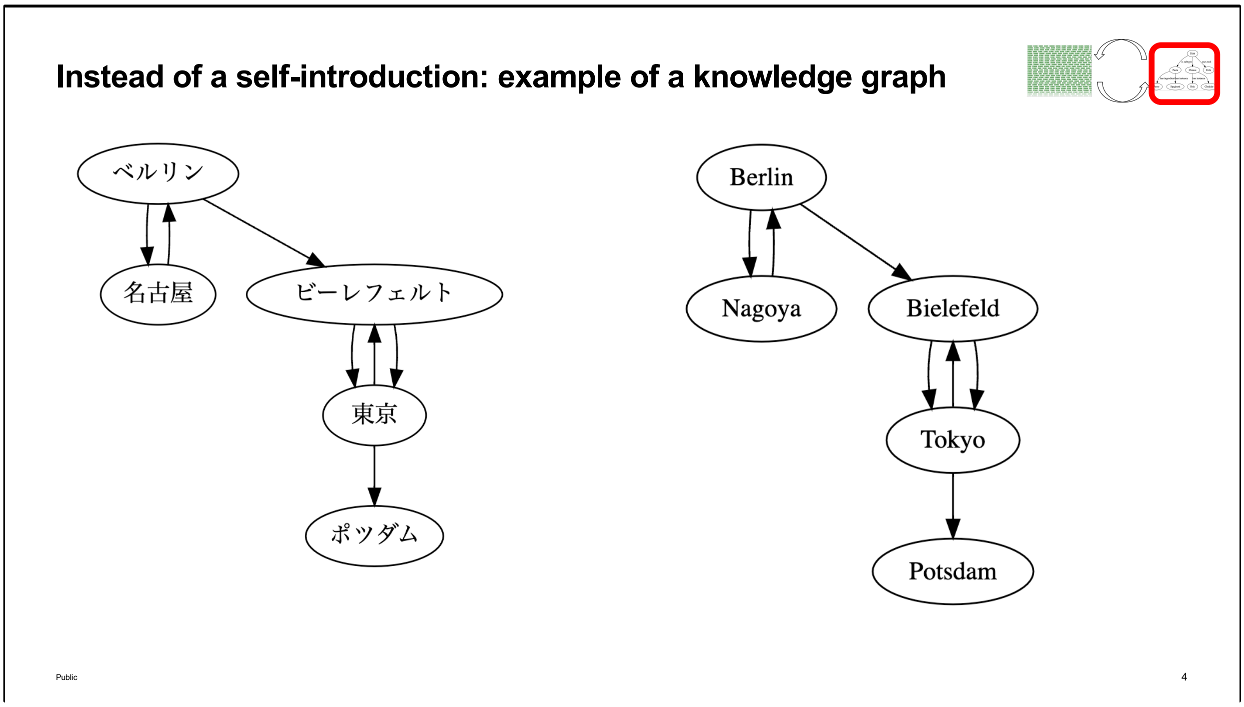
Large Language Models (LLM) 

Retrieval, LLMs and KGs 

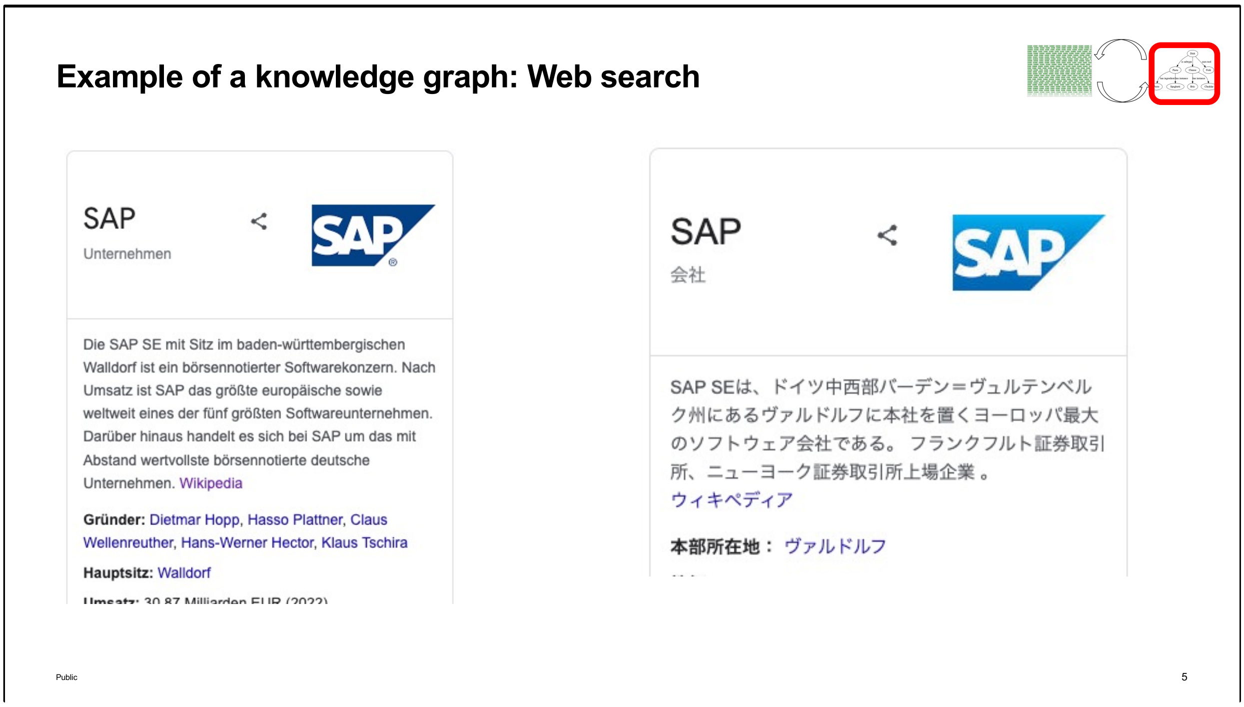
Best Practices: KGs and LLMs 



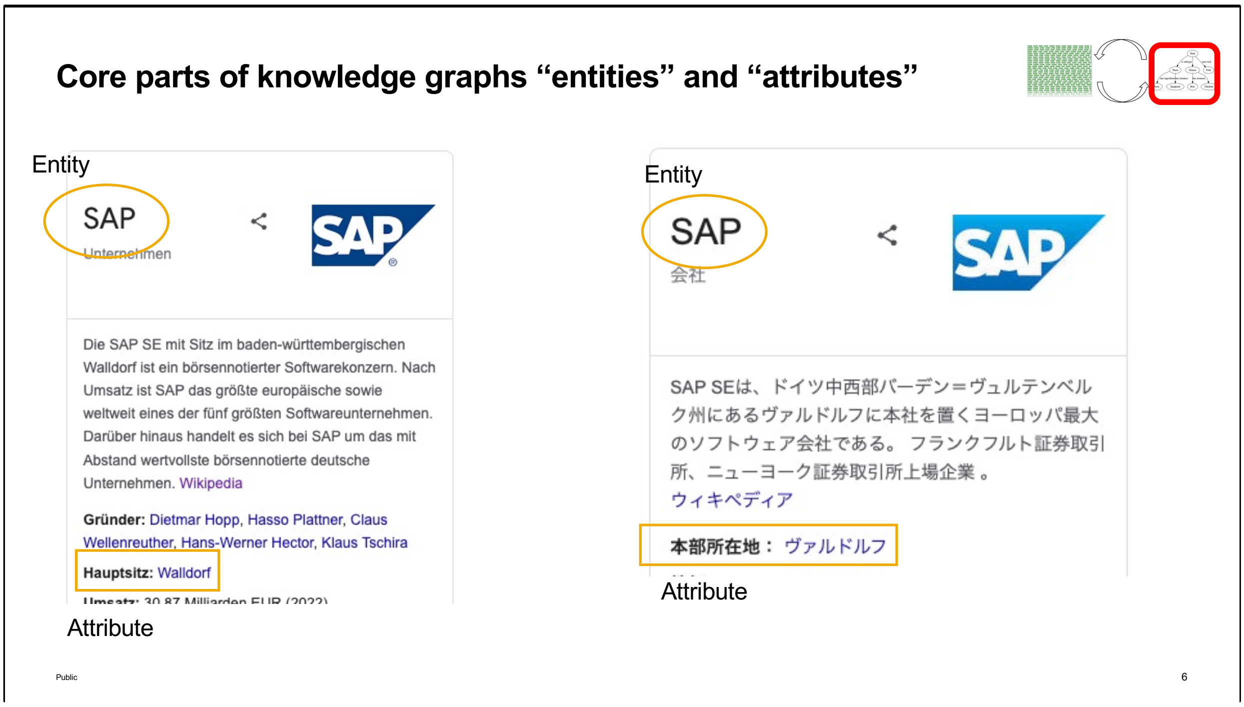
Knowledge Graphs



Here you will find two variants of knowledge graphs that represent stations of my education and work. This slide illustrates an important property of knowledge graphs. They are independent of human language and represent the structure in a domain. Descriptions in different languages can be expressed in knowledge graphs. Here you can see an example: a knowledge graph with a description in German and the same knowledge graph with a description in Japanese.

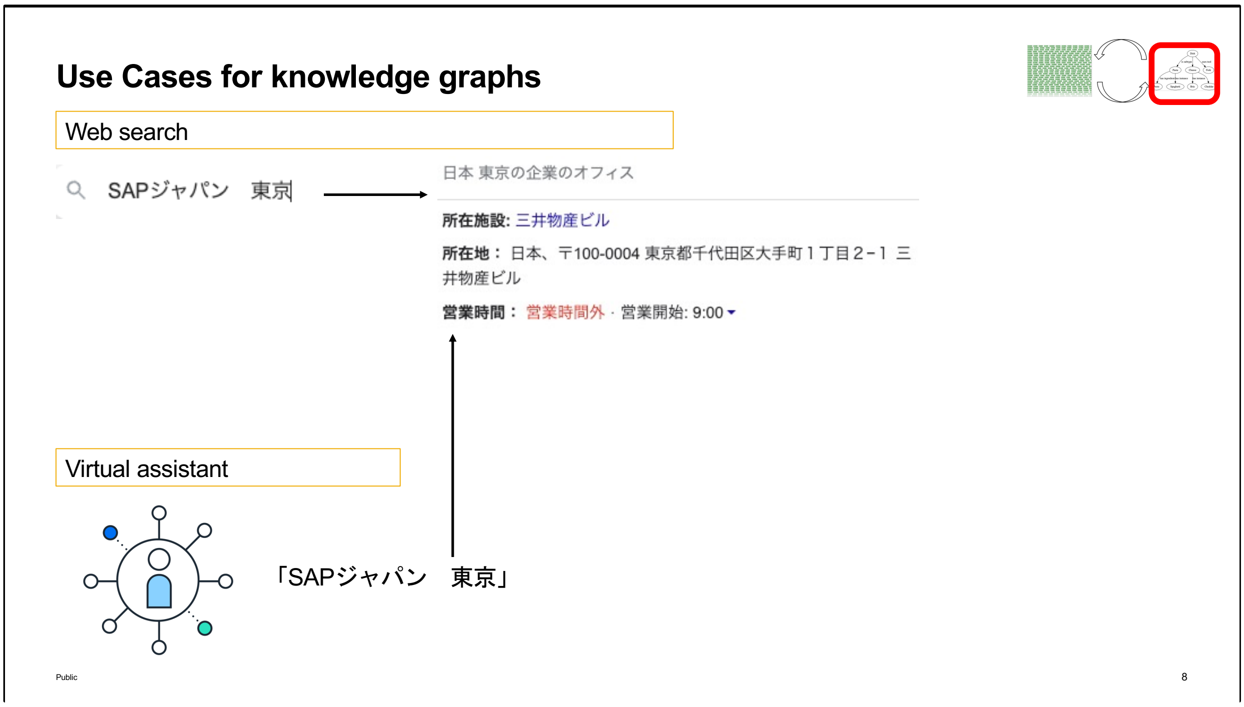


Google coined the term Knowledge Graph. An important use case for the knowledge graph is still web search. Here you can see the search results from SAP Japan. You can see that the same knowledge graph can be displayed in both Japanese and English. This underlines the importance of multilingualism for knowledge graphs.

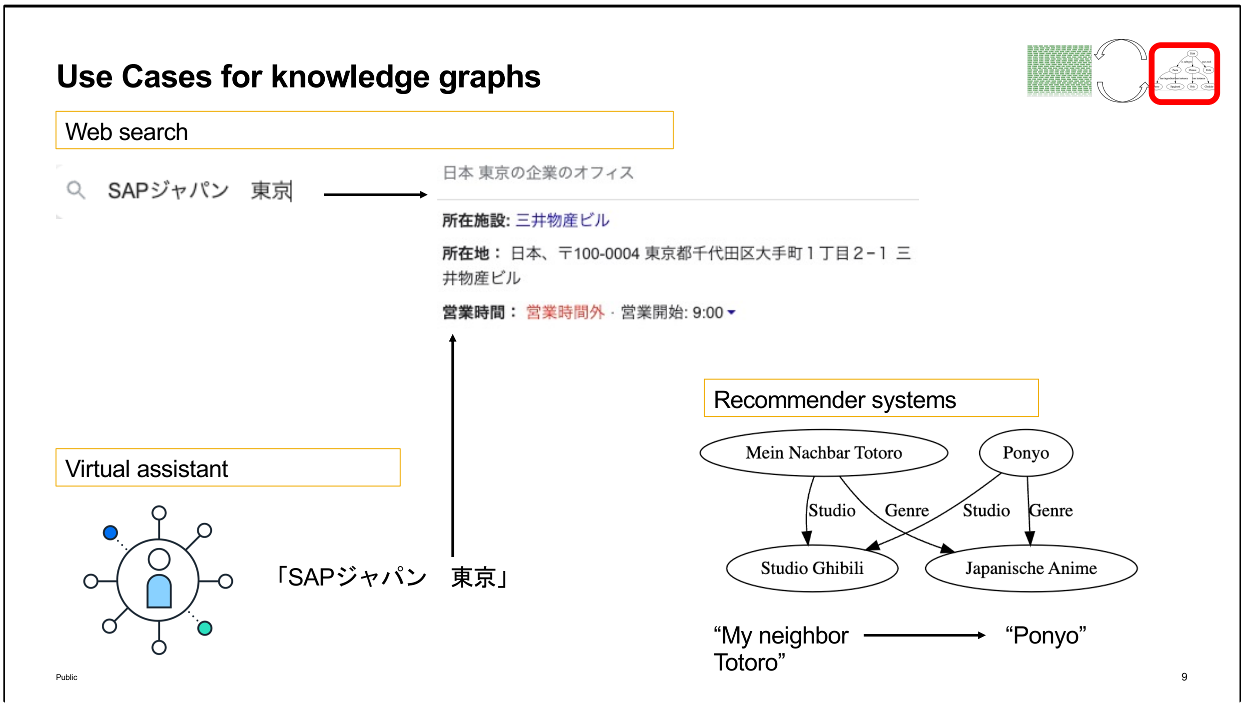


The main components of a knowledge graph are nodes and edges. Nodes are used to represent entities such as "SAP" or "Walldorf". Edges are used to define attributes. For example, the attribute "Headquarters" is used to connect an entity called SAP with an entity called Walldorf.

Now let's introduce some common use cases for knowledge graphs.
The web search is still an important use case. Here you can search for SAP Japan Tokyo. The search results contain various characteristics of SAP Japan, e.g. location and opening hours.

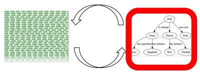


Another important use case for the knowledge graph is the virtual assistant. Virtual assistants can use the same knowledge graph as the web search. As you can see from this example, the difference between the web search and the virtual assistants is that they offer a different user interface for the knowledge graph.

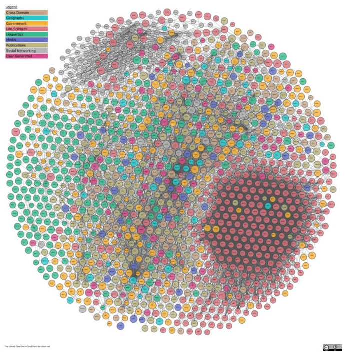


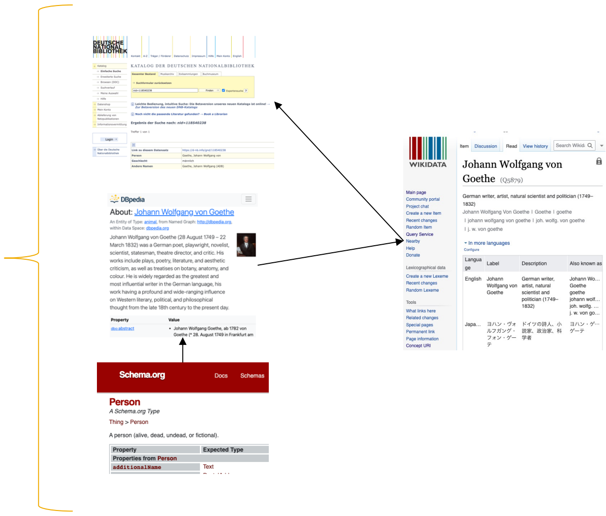
Another common use case for knowledge graphs are recommendation systems. Here you can see a knowledge graph that contains the film "My Neighbor Totoro" and the film "Ponyo". On the basis of such a knowledge graph, a recommendation for "Ponyo" can be made when searching for "My Neighbor Totoro".

Publicly available and interlinked knowledge graphs



Linked Open Data – FAIR * data





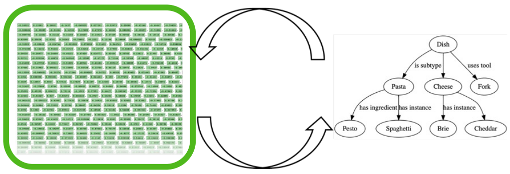
* https://www.go-fair.org/fair-principles/

Public

In the domain of libraries see e.g. Semantic Web in Libraries

10

There is an increasingly growing amount of publicly available knowledge graphs, also known as "Linked Open Data". Here you can see the Linked Open Data Cloud, an automatically generated diagram of the LOD. The colors express different domains. Well-known general and linked knowledge graphs are Wikidata or DBPedia, and common vocabularies such as Schema.org. The DNB also provides catalog data in the format of knowledge graphs. And the library community has been active in this area for some time, for example in the conference series "Semantic Web in Libraries".



Large Language Models (LLM)

LLM in context

Generative AI

Image generation

Music generation

Code generation

GitHub Copilot

Text generation

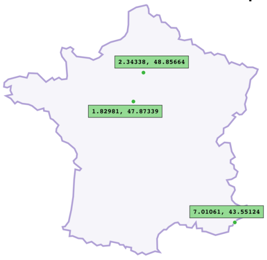
Foundation models: AI models that can be adapted to new, multiple tasks

Large Language Models: generative AI for text content

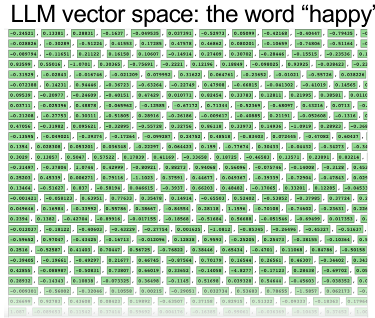
Large language models are a type of "generative AI". Generative AI is an approach to generating content. "Content" can be images, such as "teddy bears working on new AI research underwater using 1990 technology", or code, music, etc.
Foundation models are AI models that can be adapted to different and unforeseen tasks. Large language models are generative AI for processing text content.
In this presentation, the term "large language models" is mostly used. But what I say can also be applied to foundation models in general.

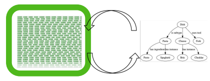
How do LLMs work (1/2)

Positions in 2D vector space



LLM vector space: the word "happy"





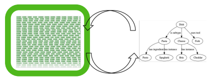
Public

Source: [How AI chatbots like ChatGPT or Bard work – visual explainer](#) Interactive demo: https://bbycroft.net/llm

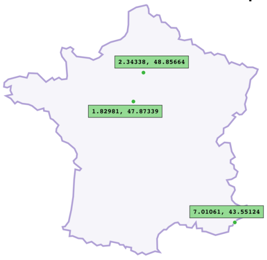
13

Large language models capture the meaning of words in a vector space. Physical objects can be anchored in a two-dimensional vector space, for example on a map. The vector spaces of language models have a much larger number of dimensions. Here you can see the description of the word "happy" via vectors.

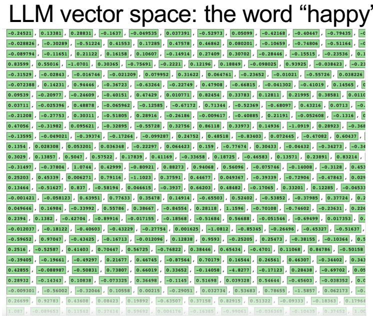
How do LLMs work (2/2)



Positions in 2D vector space



LLM vector space: the word “happy”



What is a **tidy** thing to eat pasta with?

`A` `tidy` `thing` `to` `eat` `pasta` `with` `is` `a` `fork.`

What is a **nice** thing to eat pasta with?

`A` `nice` `thing` `to` `eat` `pasta` `with` `is` `pesto.`

- **Transformer**: re-calculation of vectors for each token (“word”)
- **Attention**: consider salience of previous text into account

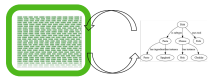
Public

Source: How AI chatbots like ChatGPT or Bard work – visual explainer Interactive demo: https://bbycroft.net/llm

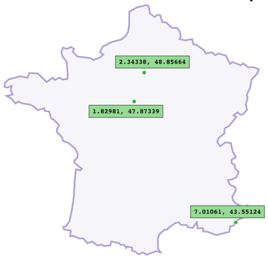
14

In large language models, the relationships between words are reweighted depending on the previous words in the input and the vectors are recalculated. The relation between, for example, "tidy" with "utility" and "tools", influence the result for the word "with" in the first sentence. It is interpreted in the sense of "using" or "as a tool". In the second sentence, the word "nice" influences the interpretation of "with" in the sense of "together with" or "with the supplement"
The consideration of the context is referred to as "attention". The process of reweighting relationships between words and recalculating the vectors is called a "transformer". The processes are applied to each part of a conversation. As a result, the meaning of the words in relation to the overall context is constantly recalculated.

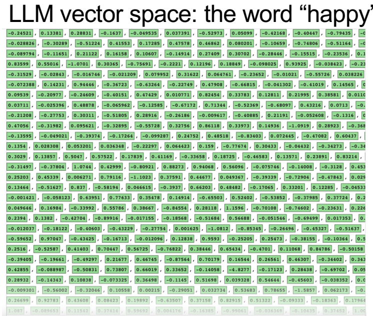
How do LLMs work, and what that means



Positions in 2D vector space



LLM vector space: the word "happy"



What is a **tidy** thing to eat pasta with?

`A` `tidy` `thing` `to` `eat` `pasta` `with` `is` `a` `fork.`

What is a **nice** thing to eat pasta with?

`A` `nice` `thing` `to` `eat` `pasta` `with` `is` `pesto.`

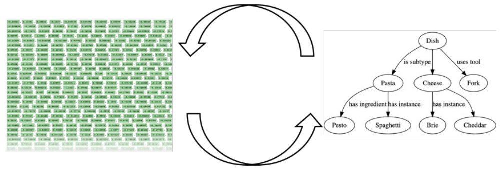
- Transformer: re-calculation of vectors for each token ("word")
- Attention: consider salience of previous text into account
- **Billions of parameters used for re-calculation → (computationally) expensive, slow, based on probabilities**

Public

Source: How AI chatbots like ChatGPT or Bard work – visual explainer Interactive demo: https://bbycroft.net/llm

15

The described functionality of large language models has consequences. Firstly, vector calculations are very complex. In the first version of ChatGTP, the recalculation of weights uses 175 billion variables, so-called parameters. This means computationally intensive, time-consuming and expensive operations. Secondly, the result is probability-based: the model calculates a statistical probability for the next word in the output.

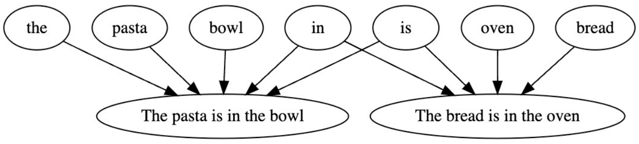


Retrieval, LLM and Knowledge Graphs

What role do LLM and knowledge graphs play in information retrieval? To answer these questions, we want to shed light on the relationship between retrieval approaches.

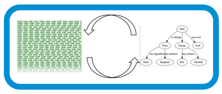
From inverted index to LLMs (1/3)

Inverted index

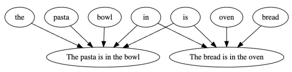


A basic form of retrieval is the inverted index. The advantage is that words are clearly assigned to documents. Processing is therefore fast and precise. One disadvantage is that the semantic relationships of terms are not taken into account.

From inverted index to LLMs (2/3)



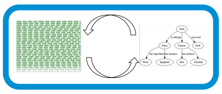
Inverted index



Word embeddings

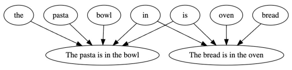
Source: [WebVectors: word embeddings online](#)
"[a word is characterized by the company it keeps](#)"

Word embeddings capture the meaning of words as vectors. The calculation is based on the embedding in the context of large amounts of data. You can immediately see the added value for retrieval. Words such as "cat" and "dog" appear in a similar context to "animal" or "rabbit". These relationships can be used for a semantic search. However, the context in a specific sentence is not yet captured.



From inverted index to LLMs (3/3)

Inverted index

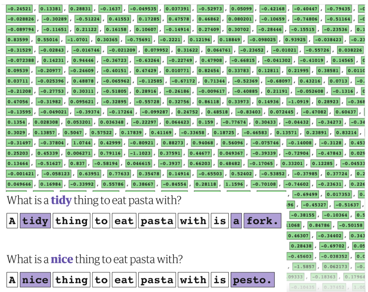


Word embeddings

Source: [WebVectors: word embeddings online](#)

"[a word is characterized by the company it keeps](#)"

LLMs



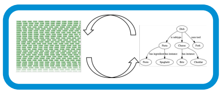
"[attention is all you need](#)"

Public

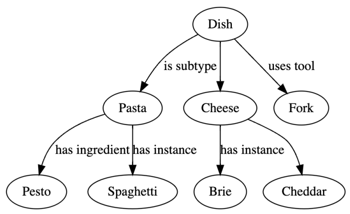
19

Large language models are able to do this. They provide the described mechanisms of attention and recalculation of weights. This makes it possible to capture further contextual relationships that go beyond word embeddings, such as the relationship between "tidy" and "a fork" or "nice" and "pesto" in the example, and the different meanings of "with" described above.

Knowledge graphs and LLMs



KGs



Explicit modeling

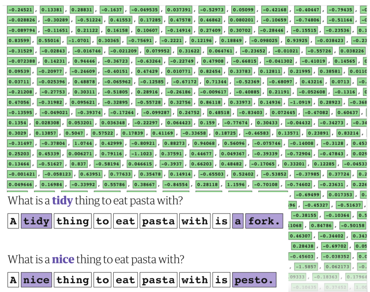
Word embeddings

Source: [WebVectors: word embeddings online](#)

"[a word is characterized by the company it keeps](#)"

Implicit modeling

LLMs

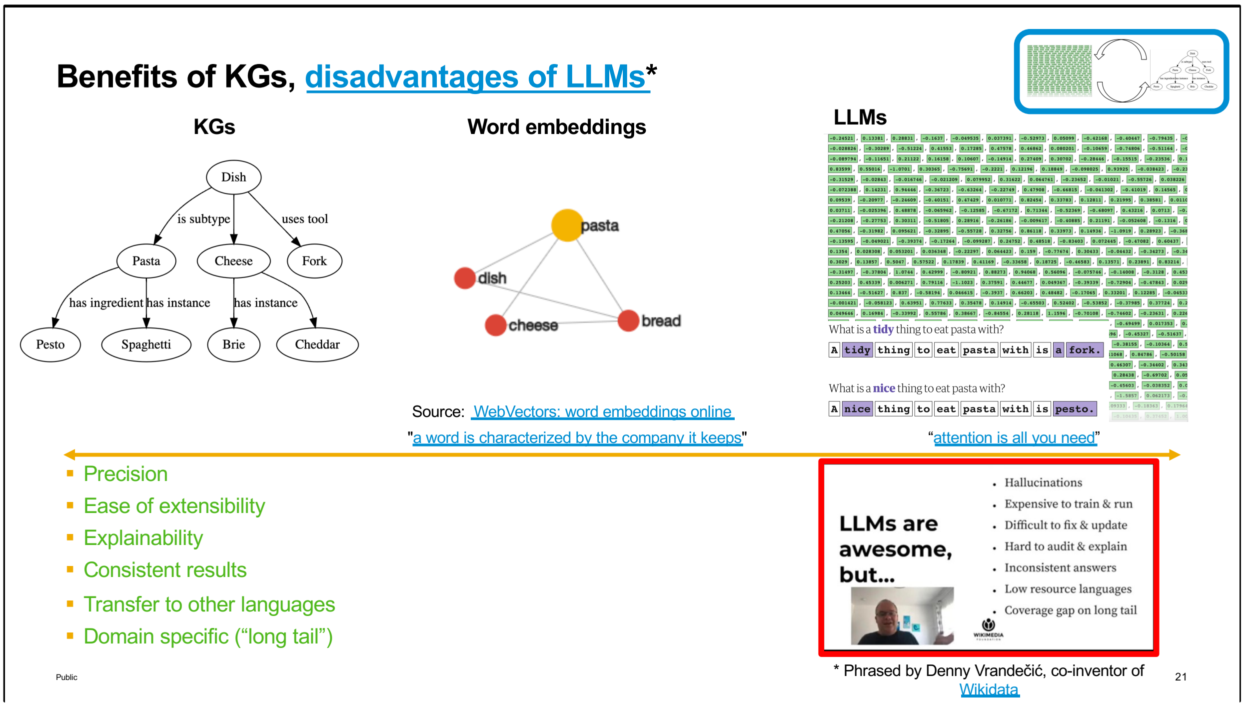


"[attention is all you need](#)"

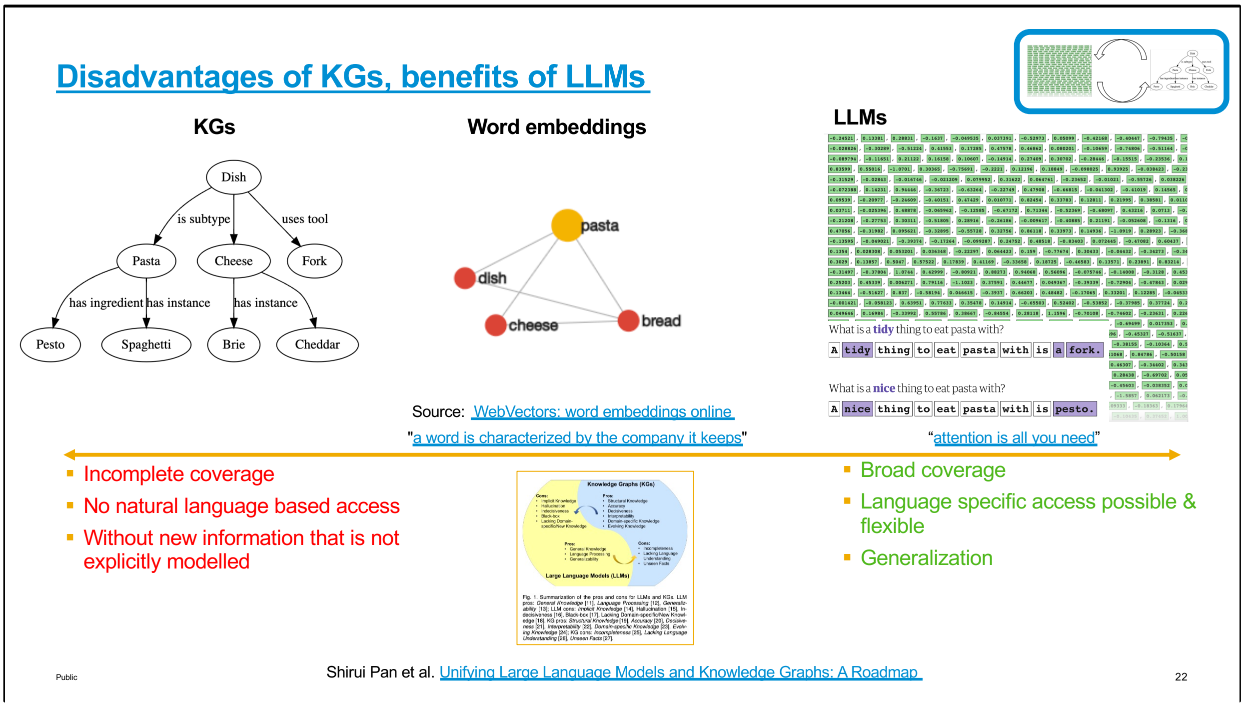
Public

20

Knowledge graphs are an explicit form of knowledge modeling. Here is a knowledge graph for the domain "food". It can be used to capture the relationships between "pesto" and "pasta", for example. However, the graph would not capture the relationship between "nice" and "pasta", or the meaning of "with" in a concrete usage context.



Knowledge graphs have various advantages. The knowledge is precise, the graphs can be easily extended, the results of a search in the graph are explainable. The results are also consistent and applicable to other languages, cf. my CV in Japanese. In addition, unlike LLMs, knowledge graphs can be used to capture domain-specific "long tail" knowledge without extensive data.
Denny Vrandečić, one of the co-founders of the knowledge graph "Wikidata", has described the challenges of LLMs in an impressive video, which is linked here. This slide summarizes these challenges. The phenomenon of hallucination should be emphasized particularly. One example is the request from December 3, a few days before this conference took place. We ask a language model "Tell me about the conference "AI in libraries: new directions with large language models". The model replies that the conference has already taken place and was a complete success, and that it provided valuable impetus for the practical implementation of AI-supported solutions in libraries.



Knowledge graphs do not hallucinate, but they also have disadvantages, which the article linked here summarizes and relates to LLMs. Although knowledge graphs are easy to extend, the coverage of a domain can only be achieved manually with great effort and ensured by continuous updating.
Knowledge graphs do not have flexible language-related access. This knowledge graph has all the information for a question such as "Which side dish goes well with pasta?". But a corresponding query to the graph requires knowledge of query languages. In addition, information that is not explicitly modeled is not taken into account in the knowledge graph. The relationship between "nice" and "pesto" again serves as an example. It makes sense in the context of an utterance. However, the graph does not have this context, and it is difficult to formulate all conceivable contexts in the knowledge graph.
An LLM can help to compensate for these disadvantages. It has a large knowledge coverage due to the extensive training data. An explicitly defined index cannot be asked questions in natural language. Explicitly modeled information that arises from the context, such as the relationship between "nice" and "pesto", can be inferred by the LLM.

LLMs and KGs: Research and best practices

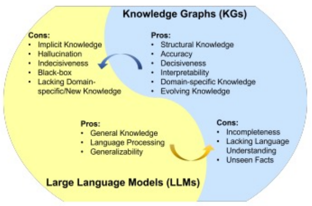


Fig. 1. Summarization of the pros and cons for LLMs and KGs. LLM pros: *General Knowledge* [11], *Language Processing* [12], *Generalizability* [13]; LLM cons: *Implicit Knowledge* [14], Hallucination [15], Indecisiveness [16], Black-box [17], Lacking Domain-specific/New Knowledge [18]. KG pros: *Structural Knowledge* [19], *Accuracy* [20], *Decisiveness* [21], *Interpretability* [22], *Domain-specific Knowledge* [23], *Evolving Knowledge* [24]; KG cons: *Incompleteness* [25], *Lacking Language Understanding* [26], *Unseen Facts* [27].

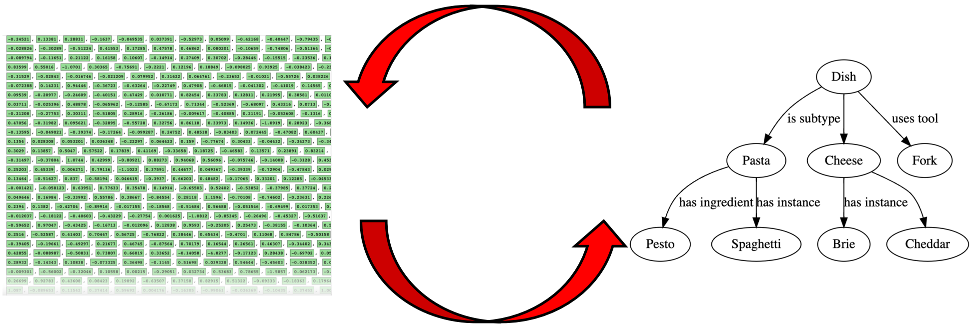
- Curt Cagle: Nine ChatGPT Tricks for Knowledge Graph Workers
- Dean Allemang: AI's Woolf at the door – LLMs and Knowledge Graph, Figuring out vs. telling, LLM's Closing the KG Gap
- Denny Vrandečić: The Future of Knowledge Graphs in a World of Large Language Models
- Knowledge graphs and large language models – chapter in Dagstuhl seminar report "Knowledge Graphs and their Role in the Knowledge Engineering of the 21st Century", September 2022

Shirui Pan et al. Unifying Large Language Models and Knowledge Graphs: A Roadmap (2023)

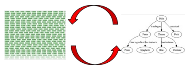
Public

23

In addition to the article described above, many discussions have been published this year that describe the addition of knowledge graphs and LLMs in detail.
What we need now are best practices. How can knowledge graphs and large language models benefit from each other? What role do they play in a particular task? How can the two be used together? A lot of such best practices have been developed in the knowledge graph community this year. Especially the articles by Curt Cagle and Dean Allemang linked here are worth reading.



Best Practices: KGs and LLMs



Decide based on use case needs

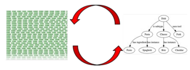
- Are LLMs and KGs needed at all?
- Divide scenarios in smaller use cases, with or without KGs or LLMs
- Costs are still important: latency and money
- In general
  - Embeddings and Retrieval Augmented Generation (RAG) are of high value, with or without KGs
  - LLM tools and agents can address hallucinations

Scenario: Retrieval in the context “library catalogue”

Public

We have chosen the library catalog as a suitable scenario for this conference. We want to see in which use cases LLMs and knowledge graphs can create added value.

Use Case “targeted search”



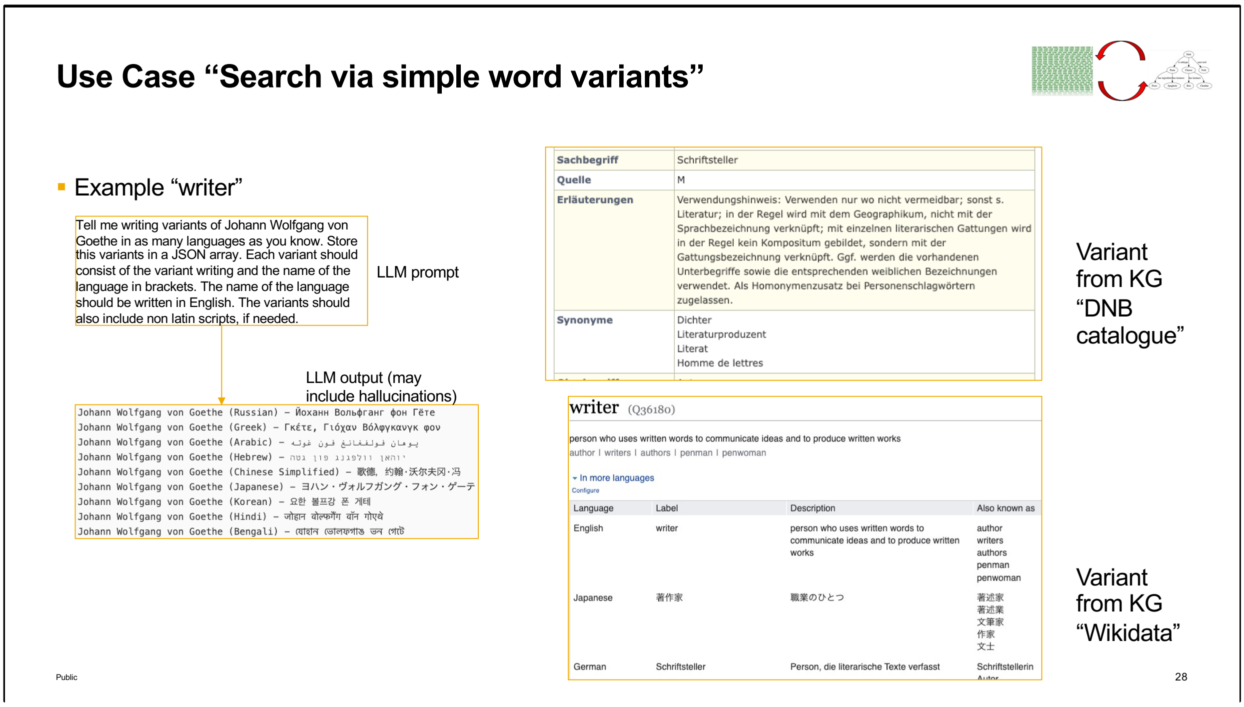
- Search for ID ”118540238” does not need a KG or an LLM ☺
- Example “life dates of J. W. v. Goethe”: verified knowledge can be stored explicitly, e.g. in a KG


```
<https://d-nb.info/gnd/118540238> gndo:dateOfBirth "1749-08-28"^^xsd:date;
  gndo:dateOfDeath "1832-03-22"^^xsd:date .
```

Public

27

If you are looking for specific information in a selected category, you do not need an LLM. Here is an example: if the ID "118540238" for Goethe is known from the DNB, a database search is sufficient. It is fast, precise and ultimately cheap compared to LLM processing. More extensive, verified information such as the life data can also be stored in the knowledge graph. This makes it easier to reuse the verified knowledge.



For general entities, you can find word variants in publicly available knowledge graphs, for example for different languages. Here is an example for Goethe from the knowledge graphs Wikidata and the DNB catalog. LLMs also have this knowledge. Here you can see a query, an LLM prompt, to get the knowledge. The LLMSs can fill gaps in the knowledge graphs for general entities. However, there is a risk of hallucinations. Enriching a search index with these variants therefore increases the number of hits, but not the precision of the search.

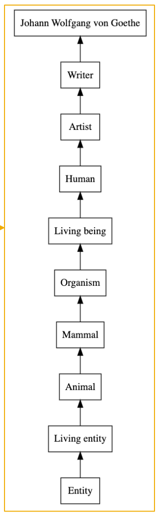
Use Case "Search via semantic hierarchies"

- Example "concept hierarchy"

I want to know something about concepts related to Johann Wolfgang von Goethe. I want to know all concepts that are broader than Johann Wolfgang von Goethe. For example, "human" is a broader concept, or "mammal" is an even broader concept. Create a list of these concepts. Have a maximum of 10 concepts. The most abstract concept should be "entity". Output the list as a file in the DOT language. Each concept should be encoded as a node. The broader concept relations should be encoded as arrows between the nodes.

LLM prompt

LLM output (may include hallucinations)



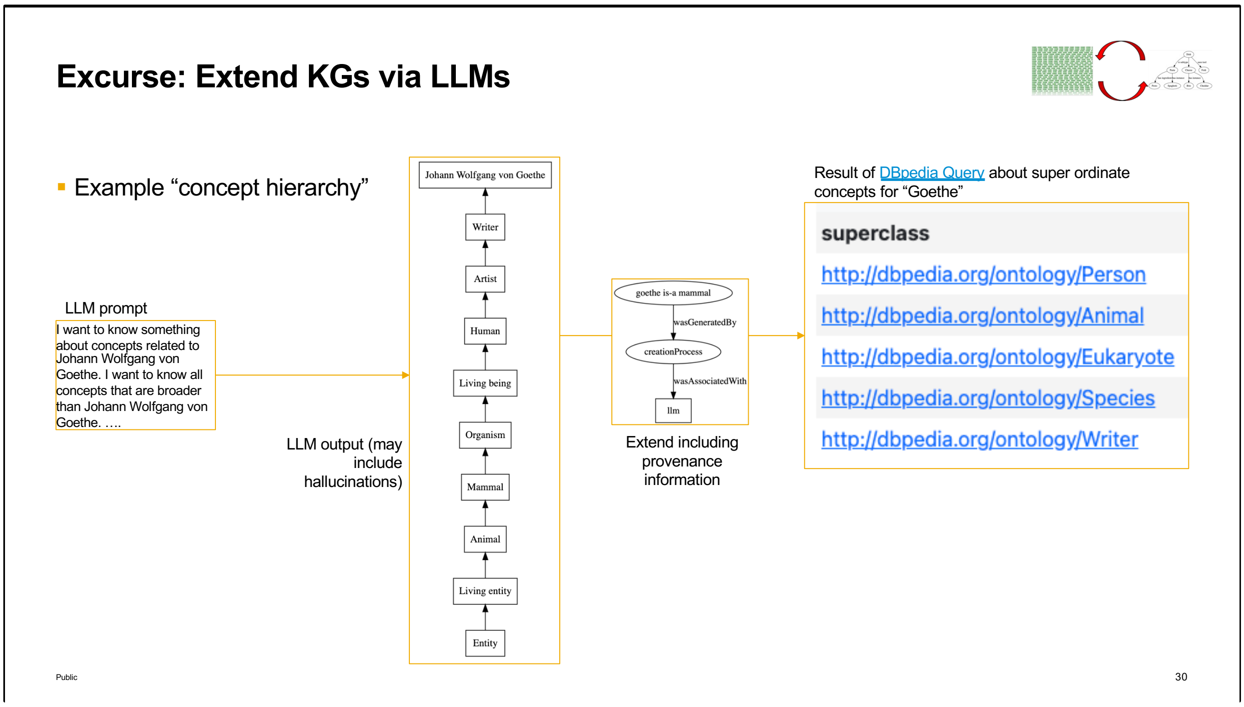
Result of DBpedia Query about super ordinate concepts for "Goethe"

superclass
http://dbpedia.org/ontology/Person
http://dbpedia.org/ontology/Animal
http://dbpedia.org/ontology/Eukaryote
http://dbpedia.org/ontology/Species
http://dbpedia.org/ontology/Writer

Public

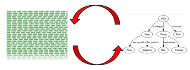
29

Both LLMs and knowledge graphs can provide further information for retrieval. Here you can see an LLM prompt to achieve semantic enrichment with generic terms. You get a variety of terms, not all of which are relevant, such as "Mammal". The query to the DBPedia knowledge graph gets fewer results, but with higher relevance.



Here is an excursus on the topic of "Extending knowledge graphs with LLMs". The described example of generating concept hierarchies with LLMs could be used for this purpose. This jeopardizes the quality of the secured information. One approach is to make the origin of the information explicit in the extension. Here you can see how the statement "Goethe is-a mammal" can be identified as having been generated by an LLM. This then allows selected handling in further processes, e.g. filtering of saved information generated by LLMs.

Excurse: Generate KGs via LLMs



(1) LLM prompt

System prompt

You are an expert agent specialised in analysing person descriptions.

Your task is to identify the entities and relations requested with the user prompt, from a given person description specification.

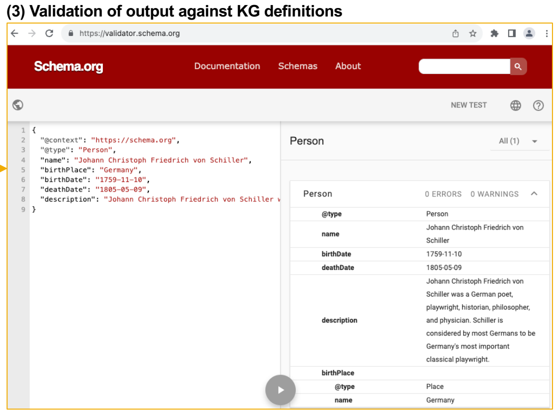
User prompt

```
Use the following entity types:

# ENTITY TYPES:
entity_types = {
  "person": "https://schema.org/Person",
  "country": "https://schema.org/Country"
}

Use the following relation types:
relation_types = {
  "name" : "https://schema.org/name",
  "birthPlace" : "https://schema.org/birthPlace",
  "birthDate": "https://schema.org/birthDate",
  "deathDate": "https://schema.org/deathDate",
  "description": "https://schema.org/description"
}

Example input: …
Example output: …
```

(2) LLM output

```
{
    "@context": "https://schema.org",
    "@type": "Person",
    "name": "Johann Christoph Friedrich von Schiller",
    "birthPlace": "Germany",
    "birthDate": "1759-11-10",
    "deathDate": "1805-05-09",
    "description": "Johann Christoph Friedrich von Schi
}
```

(3) Validation of output against KG definitions



Description of approach: Automated Knowledge Graph Construction using ChatGPT

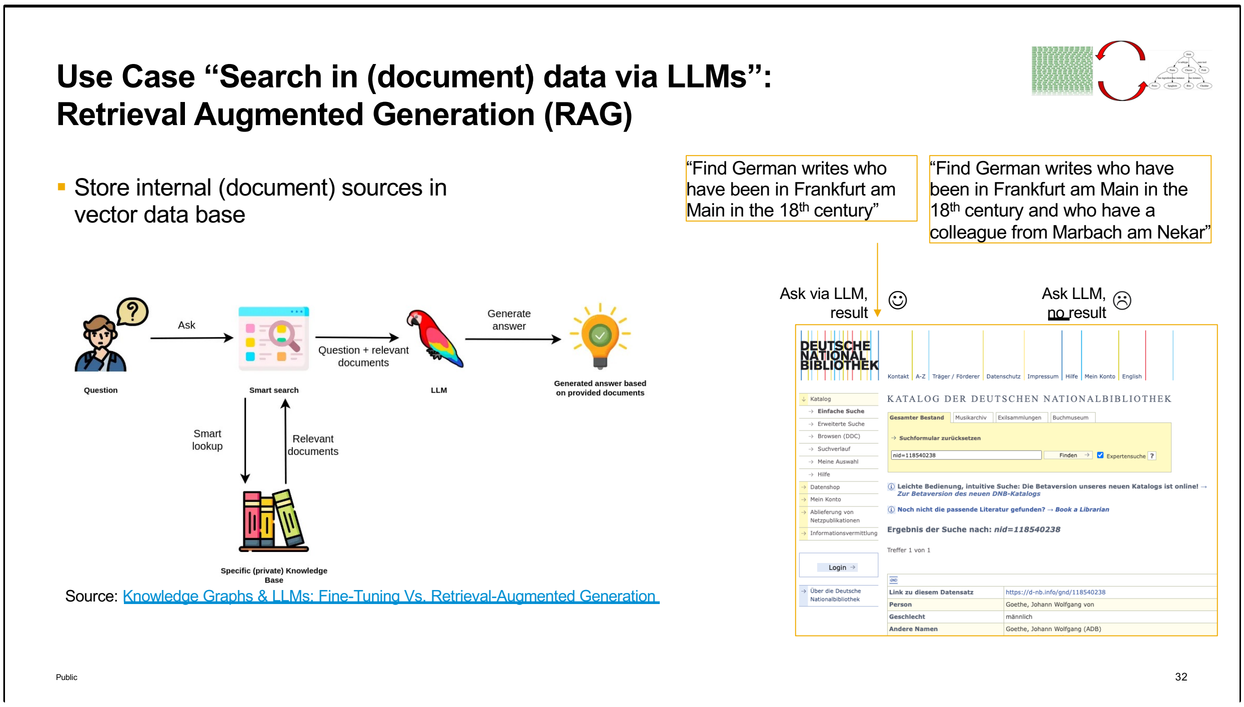
Advantage of using standardized KG vocabularies like Schema.org:

- LLMs have processed lots of Web contents including these vocabularies (see the statistics at Webdatacommons) → increases quality of output
- There are tools for further processing, including validation

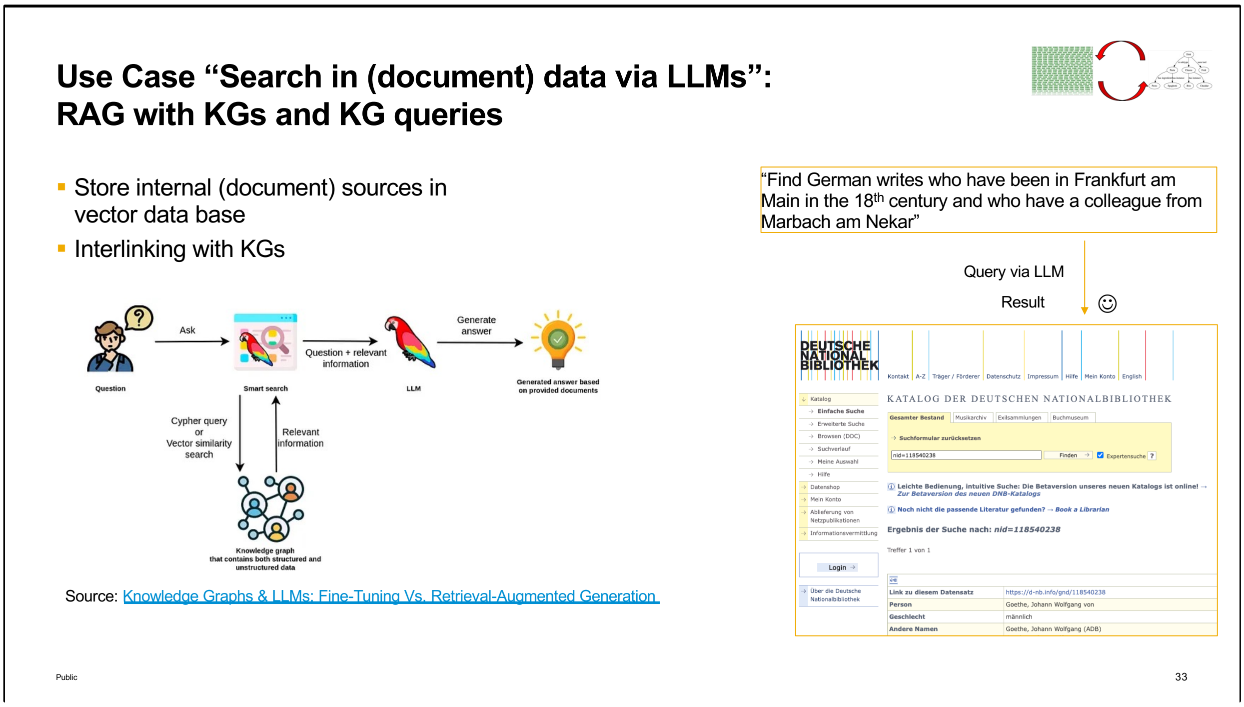
Public

31

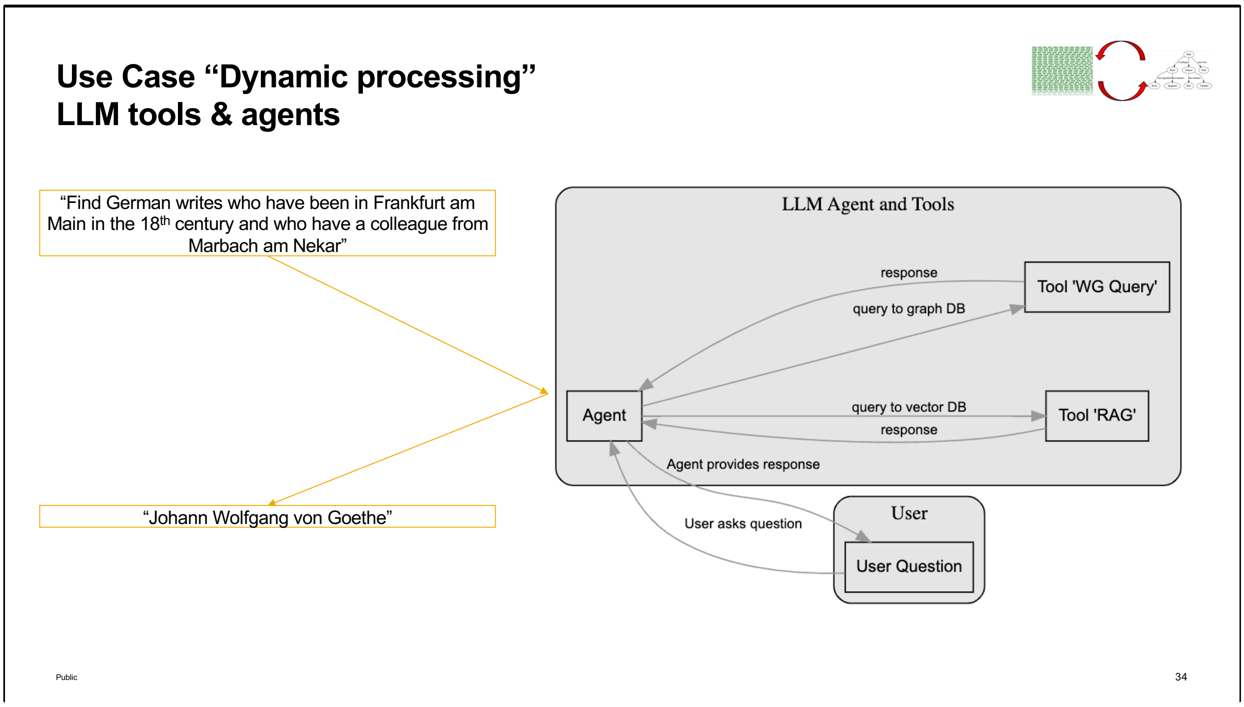
LLMs can also be used to generate knowledge graphs. Here is an example workflow in which the prompts and especially the provision of examples play a major role.



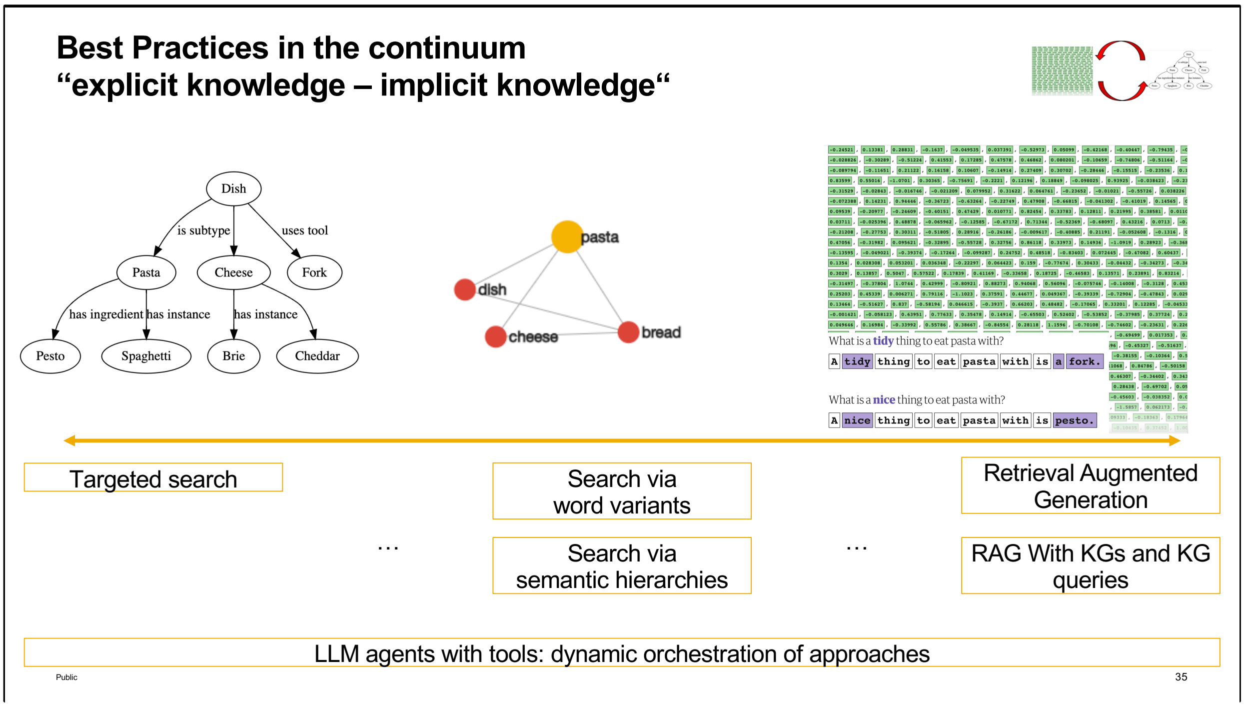
An increasingly common pattern in the use of LLMs is retrieval augmented generation. Mostly text documents or even simple structured data records are stored as embeddings in a vector database. A user query can then be processed against the embeddings and the matching document(s) can be found. The query "Find German writers born in Frankfurt am Main in the 18th century" will then probably return a result, as the necessary information can be found in a document or data record. What you can't achieve with it: Successfully submit queries based on interlinked information from different sources. "Find German writers who were born in Frankfurt am Main in the 18th century and who have a colleague from Marbach am Neckar". This query would have to take into account information from the data set on Goethe and Schiller. However, these semantic links are not stored in the vector embeddings.



This type of functionality can be achieved by combining retrieval in the vector database and processing in the knowledge graph. By searching in the vector database, entities such as "Goethe" or "Schiller" can be recorded. A query can then be executed in the knowledge graph for these entities, taking into account information such as the birthplaces of the two writers.

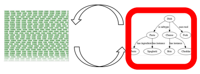


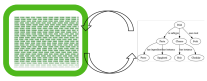
Such a combination of LLM and knowledge graphs is very powerful. The challenge, however, is that it is not possible to predict which processing is required before a query is made. There are two relevant mechanisms for this in the context of LLM: tools and agents.
Tools enable the LLMs to trigger external processing. These can be arbitrary, for example a web search or a database query. Agents allow tools to be called dynamically. This slide outlines which tools can be helpful for the complex question about Goethe and Schiller described above. These are a tool for retrieval augmented generation and a tool for querying knowledge graphs. The agent decides in which order the tools are called and how their intermediate output is further processed.

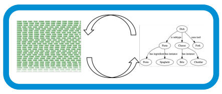


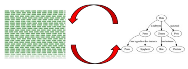
This concludes the presentation of best practices for the use of LLM and knowledge graphs. The best practices are on a continuum, from the use case of targeted search to the use of agents. And as mentioned earlier, non-technical aspects such as cost-effectiveness or latency must also be taken into account when selecting an approach. Like AI in general, the topic is very dynamic. And we did not address some of the methods currently being discussed in this presentation, such as the extension of LLMs with knowledge graphs via so-called fine-tuning. We will be happy to do this in the discussion or on another occasion.

Overview

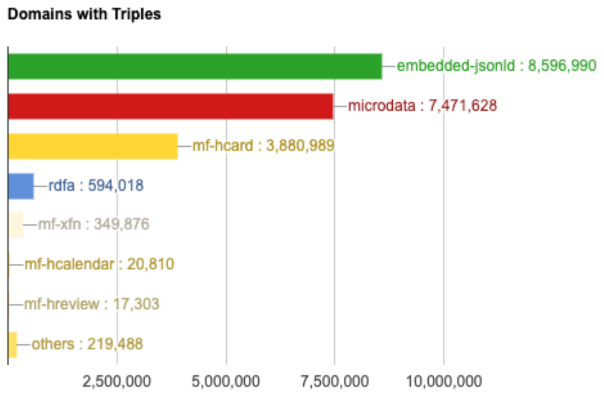
Knowledge Graphs (KGs) 

Large Language Models (LLM) 

Retrieval, LLMs and KGs 

Best Practices: KGs and LLMs 

[Web Data Commons Data Sets – October 2022](https://webdatacommons.org)



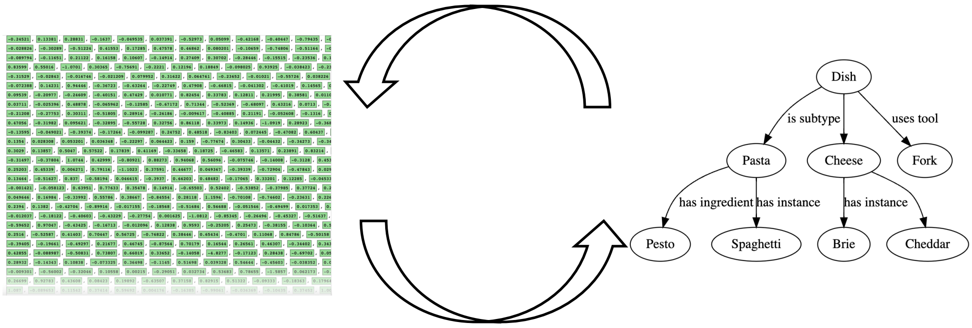
At the end a call to action: publish your data FAIR*
You will contribute to the quality of LLMs ☺

* https://www.go-fair.org/fair-principles/

Public

37

Finally, an appeal, especially to the library community. Publish your data on the Web FAIR. LLMs are largely based on web content, including knowledge graphs. These are therefore also incorporated into LLMSs. They thus contribute to the quality of LLMs. To conclude with the words of Denny Vrandečić: in a world of unlimited amounts of content, knowledge is very valuable.



Interaction between knowledge graphs and large language models

Felix Sasaki, Chief Expert Knowledge Graphs and Semantic Technologies, SAP AI Unit
Conference “KI in Bibliotheken - Neue Wege mit großen Sprachmodellen?” (“AI in libraries – new approaches with large language models?”)
December 8th 2023
Public