Language Models and Knowledge Bases: A Review and a Small Experiment on Graph-Only Facts

Seminar on Big Data Management, University of Helsinki, Autumn 2026

Tim Kicker
University of Helsinki
Helsinki, Finland

Abstract

I have read five papers on language models and knowledge bases [1–3, 6, 7] in this seminar. In this report I review how the field moved from the cost of building and maintaining knowledge bases, to the hope that language models could replace them, to the evidence that much of their apparent knowledge is guessing and finally to proposals for combining both. I compare knowledge bases and language models along five properties: access, editability, consistency, reasoning and explainability. I then discuss the strengths and weak points of the papers. The weak point they share is the yardstick, because none of them compares language models with a knowledge base that is maintained over years.

As my own contribution I test whether a language model takes its facts only from a knowledge graph when it is told to. On a small graph from Wikidata with deliberately wrong facts, local models from 2 to 27 billion parameters followed the graph in almost every case under a strict instruction, even when they knew the true answer. Such a design keeps the weights out but moves all responsibility for correctness to the graph.

Keywords

knowledge bases, knowledge graphs, language models, factual knowledge, knowledge conflicts

1 Introduction

Knowledge bases store facts as explicit entries. Each fact can be looked up, checked against a source and corrected. The knowledge base that Deshpande et al. [3] describe shows what this means at the scale of big data. It was built mainly from Wikipedia and held 13.2 million concepts and instances with 165 million edges in its relationship graph. It served query understanding, advertising and later product search at Walmart. The price was the work behind it: a dedicated team, a full rebuild of 12.5 hours every day and several thousand hand-written corrections to get from 70% to above 90% accuracy.

Language models seem to offer the opposite trade-off. They are trained on large amounts of text without supervision and can answer factual questions out of the box. Petroni et al. [7] showed in 2019 that BERT already competes with an automatically built knowledge base on simple facts. In their LAMA probe it did slightly better than a relation extraction system that had been handed the sentences containing the answers. That started a line of research asking whether language models could replace knowledge bases altogether. Later work was more sceptical. Cao et al. [2] found that much of the apparent knowledge is guessing guided by the prompt. On a test set with evenly distributed answers precision fell from 30% to 16%, while the predictions of the model barely changed. AlKhamissi et al. [1] showed that language models lack properties a knowledge base has by design, such as editability, consistency and explainability. Pan et al. [6] expect hallucination to persist "for the foreseeable future" and therefore propose to combine both instead of choosing one. Together these papers address one research problem. Can a language model replace a knowledge base? If it cannot, how should the two be combined?

The question is not only academic. Many current systems already give a language model facts from outside, for example passages found by a search engine or entries from a database. The model is then expected to answer from them. If the model mixes these facts with what it learned in training, the answer can no longer be traced to a source. An outdated or wrong fact from the weights can also slip through unnoticed. A small example from my own experiment shows the risk. Asked without any external facts where Liu Xiaobo was born, one model refused to answer because he was "not a public figure recognized by Chinese laws and regulations" and in another wording named Beijing instead of Changchun. A graph with a sourced entry would have given the right answer in both cases. My own interest in the question comes from Arlen, a Linux desktop I designed around a local knowledge graph that an AI layer queries for context [5]. In such a system it matters whether the assistant takes its facts from the graph or from its weights.

For my own experiment I take a narrower design question from this debate:

> Is it a good design to let a language model take its facts only from a knowledge graph and not from what it learned in its weights?

In such a design the graph is the only source of facts and the model is responsible for language: it reads the question, picks the right triples and phrases the answer. The design only works if two things hold. The model must actually stay with the graph when its weights say something else. And the graph must be good enough to deserve that trust.

I first review the five papers, compare knowledge bases and language models along five properties and discuss the strengths and weak points of the papers. I then test the first condition with an experiment of my own. I built a small knowledge graph from Wikidata, gave it to local language models of different sizes and deliberately put wrong facts into it to see whether the models follow the graph or their weights. The main result is that under a strict instruction they almost always follow the graph, even when it is wrong and they know the true answer. Only the largest model I tested sometimes corrected the graph, mostly when the instruction allowed it to use its own knowledge. The literature explains why

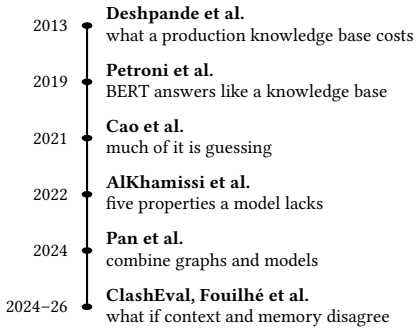


Figure 1: The papers reviewed in this report in order of publication. The last entry only serves to place my own experiment.

this is good news for the design and the experiment shows what it costs: everything now depends on the quality of the graph.

The rest of the report is organised as follows. Section 2 reviews the literature from the cost of knowledge bases to the proposals for combining them with language models. Section 3 compares knowledge bases and language models along the five properties of AlKhamissi et al. Section 4 discusses the strengths and weak points of the reviewed papers and the challenges they leave open. Section 5 presents my experiment. Section 6 summarises the papers and answers the design question.

2 Literature Review

The papers I read follow one line of argument. It starts with what a knowledge base costs in practice, moves to the hope that a language model could replace it, then to the doubts about that hope and ends with proposals to combine both. Figure 1 shows this line.

2.1 What a knowledge base costs

Deshpande et al. [3] describe the whole life cycle of a knowledge base that was used in production at Kosmix and later at WalmartLabs. Their knowledge base contains concepts, instances and relationships between them. The “is a” relationship forms a taxonomy. It was built mainly from Wikipedia and had 13.2 million concepts and instances and 165 million edges in its relationship graph. Turning the Wikipedia category graph into a tree is the hard part, because the graph has cycles and a concept can have several parent categories. The authors give every edge a vector of weights and choose the tree with an optimum branching algorithm (Edmonds’ algorithm) over the whole graph.

The main message of the paper is that building the knowledge base is only the first step and that maintenance causes most of the work. The tree is rebuilt from scratch every day, which takes 12.5 hours, because writing rules for incremental updates of the global algorithms proved too difficult. Manual corrections would be lost in every rebuild, so the analysts write them as commands in a small language that is applied again after each rebuild. With purely automatic algorithms the taxonomy was 70% accurate and with several thousand such commands, written over three to four years, it was well above 90%. A core team of four people looked after it. Despite these costs the authors conclude that even an imperfect knowledge base is useful for many applications. Theirs supported query understanding, deep web search, in-context advertising, event monitoring in social media and product search at Walmart. For every node they also stored metadata such as Wikipedia traffic and a time-dependent social context, for example which words had appeared most often next to a person’s name in the last hour.

2.2 Language models as knowledge bases

Petroni et al. [7] asked whether a pretrained language model already contains this kind of knowledge without any fine-tuning. Their LAMA probe turns a fact into a cloze sentence such as “Dante was born in [MASK]” and checks how high the model ranks the correct word. The facts come from four sources: Google-RE and T-REx with relations from Wikipedia and Wikidata, ConceptNet with common sense knowledge and 305 questions from SQuAD rewritten as cloze sentences. To keep the comparison fair, all models had to choose from a shared vocabulary of about 21,000 tokens and only answers of a single token were used. On Google-RE, BERT-base and BERT-large reached an average accuracy 2.2 and 2.9 points higher than a relation extraction system that was given the very sentences expressing the test facts and an oracle for linking entities. BERT ranked the correct object among the first ten in about 60% of the cases. It did very well on one-to-one relations such as the capital of a country and badly on many-to-many relations. On questions from SQuAD it reached a precision at 10 of 57.1 against 63.5 for a supervised question answering system. The authors concluded that such models have potential as unsupervised open-domain question answering systems. They also noted themselves that BERT might not do this “for the right reasons” and might only have learned which objects co-occur with which subjects.

AlKhamissi et al. [1] review the field that grew out of this paper. Instead of asking whether a language model *is* a knowledge base, they ask what a knowledge base *can do* and split this into five properties: access, editability, consistency, reasoning and explainability. Section 3 follows these five properties.

2.3 Knowledge or guessing?

Cao et al. [2] tested the co-occurrence suspicion systematically for masked models. The answers in LAMA are skewed towards a few frequent objects, for example large cities as places of birth. The authors built a new dataset, WIKI-UNI, with the same relations but evenly distributed answers. Precision at 1 fell from 30.36% on LAMA to 16.47% on WIKI-UNI, while the distribution of the predictions stayed almost the same on both datasets. Even when they also masked the subject (“[MASK] was born in [MASK]”) the predictions still correlated clearly with the normal ones. The prompt decides most of the answer and the subject only nudges it, which the authors call guided guessing. Adding ten examples to the prompt raised precision from 30.36% to 36.23%, but the authors show that the gain comes almost entirely from finding the right type of answer, for example a city instead of a country. Within the right type, the examples did not move the correct answer up. Extra context retrieved from Wikipedia helped only when it contained the answer, either directly or in a form from which the masked

answer could be reconstructed. When it did not, performance even dropped slightly.

2.4 Combining graphs and language models

Pan et al. [6] take the next step. If neither side can replace the other, how can they work together? Their roadmap describes three frameworks. In *KG-enhanced LLMs* the graph helps the model, for example by being looked up at inference time, which gives the model new knowledge without retraining. Other work in this framework adds the graph during pretraining or uses it to understand what the model knows. In *LLM-augmented KGs* the model helps the graph, for example by extracting entities and relations from text. This is exactly the work that Deshpande et al. had to do by hand or avoided. The third framework, *synergized LLMs and KGs*, treats both as equal partners. The authors expect hallucination to persist "for the foreseeable future" and see graphs as an external check for it. Among the open problems they name one that concerns my design directly. A language model is trained on text and does not understand graph structure. The obvious way around this is to linearise the graph into text, but a real graph is far too large to fit into the input and linearising it loses structure. A design that puts triples into the prompt can therefore only work with the small part of the graph that belongs to the question. The design question of this report belongs to the first framework in its strictest form: the graph is the only source of facts and the model only handles language.

2.5 When context and memory disagree

Wu et al. [8] looked at what a model does when a retrieved document contradicts what it learned in training. Their benchmark ClashEval contains 1,294 questions from six domains, for example drug dosages, Olympic records, news and dates from Wikipedia. Each question was asked without a document, with the correct document and with a document in which the answer was altered. GPT-4o adopted the altered answer in 61% of the cases in which it had answered correctly on its own. The further the altered value was from the truth and the more confident a model was without the document, the less often it followed the context. Fouilhé et al. [4] asked the same question across 31 models and found that larger models fall back on their own knowledge more often on average, but that this depends strongly on the model family and the wording.

3 Methods in Previous Work

AlKhamissi et al. [1] trace all differences back to one structural fact. In a knowledge base every fact has an address, a single entry that can be read, changed or deleted. In a language model there is no such place. The knowledge is spread over billions of parameters and the same parameters take part in many facts at once. For each of the five properties I describe how previous work deals with this and what changes when the facts come from a graph in the prompt. Table 1 summarises the comparison.

3.1 Access

A knowledge base is queried with a query language. A language model can only be accessed indirectly, either by fine-tuning or by prompting. The review finds that fine-tuning adds hardly any new knowledge and mainly teaches an interface to what the model learned in pretraining. Prompting is cheaper but fragile, because small changes to the prompt lead to large differences in performance. This fragility is exactly where Cao et al. [2] start. With a graph in the prompt, access becomes a lookup again: the system finds the triples and the model only has to read them.

3.2 Editability

Knowledge in a model can be outdated, wrong or biased. The review gives the example of the Omicron variant of COVID-19, which most language models available at the time had never seen. It can also contain private data that must be removed. Retraining the whole model is too expensive and changing weights can affect many other facts. The review lists methods in order of sophistication, from constrained fine-tuning through hypernetworks that predict the necessary weight change to ROME. ROME treats the feed-forward layers of a transformer as a key-value memory, locates the layer responsible for a fact with causal tracing and overwrites it with a rank-one update. In a sense ROME pretends after the fact that the model has entries after all. A graph does not need any of this: a fact is changed by changing its triple, which Pan et al. [6] name as the main advantage of looking up a graph at inference time.

3.3 Consistency

The review discusses three kinds of consistency. The most important one here is consistency under paraphrase. "Albert Einstein was born in [MASK]" and "The hometown of Albert Einstein is [MASK]" point to the same triple in a knowledge base and must give the same city, but a language model can answer differently. Models also struggle with negation, for example by accepting both "Birds can fly" and "Birds cannot fly". For consistency across languages the authors note that no work had measured it yet. The authors add an important remark: consistency does not imply correctness. A model can give the same wrong answer every time and a graph can be consistent and still wrong. My experiment in Section 5 shows both sides of this remark.

3.4 Reasoning

A knowledge base can follow chains explicitly: Cairo is the capital of Egypt, Egypt is in Africa, so Cairo is in Africa. Language models can reason with rules, write out chains of thought or use a scratchpad, but the review reports that the best models could not chain more than two or three non-trivial steps. In a graph the chain is explicit and the model only has to combine edges that it can see. The question about the country of birth in my experiment is a two-step question of this kind.

3.5 Explainability and interpretability

The review separates the two terms. Interpretability means inspecting the internal mechanism and explainability means justifying an output afterwards. A knowledge base is explainable by nature, because the path to the answer can be read, but it is not necessarily interpretable. A language model is neither. Pan et al. [6] give an example of what this means in practice: asked "When did Einstein discover gravity?", a model may answer "Einstein discovered gravity in 1687". Nothing in the model shows where this answer

Table 1: The five properties of AlKhamissi et al. [1] for a knowledge base, a language model on its own and a language model that answers only from triples in its prompt. The last column is my own summary of this section. Section 5 tests parts of it.

Property	Knowledge base	Language model alone	Language model with graph in the prompt
Access	query language, exact	fine-tuning or prompting, sensitive to wording	lookup of triples, model reads them
Editability	change one entry	retraining or weight editing (ROME), side effects	change the triple
Consistency	same triple for every paraphrase	answers can change with the wording	same triples for every wording
Reasoning	explicit chains over edges	two or three steps (as of 2022)	model combines visible edges
Explainability	path can be read	neither explainable nor interpretable	answer traceable to the given triples

came from. Whether attention weights can serve as explanations is disputed and explanations generated by the model itself are often inconsistent. The more productive insight was that feed-forward layers work as key-value memories, which is what ROME builds on. If the model answers only from triples in the prompt, every answer can at least be traced back to the triples it was given.

4 Strengths, Weak Points and Open Challenges

4.1 Strengths

Deshpande et al. report from a system in daily use. The paper does not stop at how the knowledge base was built. It also gives the cost of keeping it running: a core team of four people, a rebuild of 12.5 hours every day and several thousand correction commands written over three to four years. The authors say openly what did not work, for example incremental updates. Without this paper the cost side of the comparison in this report would be a guess.

Petroni et al. made the question testable. Their probe compares several pretrained models without any fine-tuning and was published together with its code. Cao et al. and the review of AlKhamissi et al. both start from it. The authors also named the weakest point of their own result themselves, namely that BERT might only have learned which words occur together.

Cao et al. change one thing at a time. WIKI-UNI keeps the relations of LAMA and only changes how the answers are distributed, so the drop in precision can be traced to that change. They look at three ways of extracting facts (prompts alone, prompts with examples and prompts with retrieved context) and give a separate explanation for the gains of each.

AlKhamissi et al. split a vague question into five that can be tested. Whether a language model "is" a knowledge base has no clear answer. Whether it offers access, editability, consistency, reasoning and explainability can be checked one property at a time. Section 3 of this report is built on this split.

Pan et al. give the field a map. Their three frameworks sort the literature by the direction in which help flows, from the graph to the model, from the model to the graph or both ways. They also name concrete open problems, such as the linearisation of large graphs discussed in Section 2.4.

4.2 Weak points

The comparison in Petroni et al. is with an uncurated knowledge base. The relation extraction baseline builds its knowledge base automatically from text. Deshpande et al. [3] report that their own taxonomy was only 70% accurate with automatic algorithms alone and that it took years of human curation to get above 90%. Read this way, the result says that a language model is about as good as an *uncurated* knowledge base, which is much less surprising than the way it is usually quoted.

Petroni et al. name the alternative explanation but do not test it. They write that BERT might only have learned co-occurrence patterns. Their own analysis supports this: performance correlates with how often the object appears in the training data but not with how often the subject does. The most thorough test of this came only two years later from Cao et al. [2].

The probe measures an easier task than the one it is quoted for. Petroni et al. only use answers of a single token, one hand-written template per relation and only the object of a triple. Each choice is justified in the paper, but together they leave out many hard cases, such as names made of several words. The authors argue that a fixed template gives a lower bound of what the model knows, just as a knowledge base can only be queried by its relation id. Cao et al. show why this does not hold: templates that score better mostly fit the answer distribution of the dataset better. A different template then measures a different fit and not more or less knowledge.

WIKI-UNI is as artificial as LAMA. More people really are born in large cities. A model that has learned this distribution reflects something true about the world. A test with evenly distributed answers punishes exactly this knowledge. The drop from 30% to 16% is therefore partly a discovery and partly the price of an unrealistic test. In addition, 16% is not zero, so the paper shrinks the claim of Petroni et al. without refuting it. Finally, Cao et al. only test masked models from 2021 and do not propose a better test. Whether their mechanisms still apply to today's instruction-tuned models is open. My experiment gives a small hint that the guessing effect is still there (Section 5).

The review uses a narrow definition of a knowledge base. For AlKhamissi et al. [1] a knowledge base is a set of triples. The knowledge base of Deshpande et al. is much more: a taxonomy, a fuzzy relationship graph, metadata and time-dependent social context (Section 2). The review does not test language models against these capabilities, so its bar is lower than what a production knowledge base does. The five properties are also stated as aspects "that we deem an LM should have" and not derived.

Parts of the review have aged. It reports that the best models could not chain more than two or three non-trivial reasoning steps. Since then, models trained to reason step by step have become

common, so this limit should be checked again before it is quoted. I expect the findings on editability and explainability to hold up better, because they come from the way a model stores facts and not from its size.

The taxonomy is less automatic than it looks. Deshpande et al. place the top level of their tree, the verticals such as health or history, by hand before the algorithm runs. The correction commands are checked every day and the analyst is alerted when one is violated, so the analyst also has to maintain the commands. How often a command breaks when Wikipedia changes is not reported.

The roadmap compares an idealised graph with a real model. In the introduction Pan et al. [6] list as a strength that knowledge graphs can keep evolving as new knowledge is added. In the same paragraph, a few lines later, they write that current methods cannot cope with real graphs, which are incomplete and change all the time. Both cannot be true at the same time. Deshpande et al. show which one holds: the knowledge base is rebuilt every day because rules for incremental updates proved too difficult to write. The roadmap itself contains no experiment, so it never shows that a combination is better than either side alone. Its third framework, the synergy of both, is described as a model with four layers (data, synergized model, technique and application) that would fit almost any data processing system. It is a drawing more than a method.

4.3 Open challenges

Two questions stay open across the five papers. The first is which weaknesses of language models disappear as models grow and which come from the way they store facts. The second concerns the combination that Pan et al. propose. If a model is given facts from a graph that contradict what it learned in training, does it follow the graph? None of the five papers measures this. ClashEval and Fouilhé et al. (Section 2.5) do so for passages of text but not for triples from a graph. My experiment in Section 5 addresses this second question.

5 My Experiment: Does the Model Follow the Graph?

The papers in Section 2 ask what a language model knows. Section 4.3 ended with a question that none of them measures: if a model is told to answer only from a knowledge graph, does it do that? Or does the knowledge in its weights leak into the answer? This is the first condition of the design in Figure 2. The first run also takes up a question that Cao et al. [2] left open, namely whether guessing guided by the prompt still shows up in current models. After my short presentation I decided to test this on a small graph of my own. I ran two experiments on local models with Ollama, all at temperature 0 and without a reasoning phase.

5.1 Setup

Graph. For the first run I took 60 politicians born in 1900 or later from Wikidata. Thirty of them have 100 to 295 sitelinks in Wikidata, that is links to pages about them in Wikipedia and other Wikimedia projects (*known*). The other thirty have only 2 to 4 (*unknown*). For each person the graph stores the place of birth and the country of that place as triples, for example (`Martti Ahtisaari, place of birth, Vyborg`) and (`Vyborg, country, Russia`).

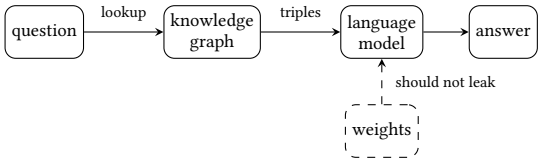


Figure 2: The design examined in this report. Facts come only from the graph. The model reads the triples and phrases the answer. What it learned in its weights should not change the facts.

Two ways of asking. In the *closed* mode the model only gets the question, so it can answer only from its weights. In the *graph* mode the prompt starts with "Use only the facts below. If the facts do not contain the answer, answer 'unknown'." followed by the triples for that person. Every question about the place of birth was asked in three wordings, for example "Where was X born?" and "X was born in which city or town?". A question about the country needs two edges of the graph.

Conflicts. For 15 people I replaced the true place of birth in the graph with a wrong one. If the model answers with the wrong place it follows the graph. If it answers with the true place the weights have won.

Models. The first run used qwen2.5:7b, gemma3:12b, granite4.2:8b and gemma4:12b, each with 525 questions. The second run used one family in four sizes (Qwen3.5 with 2, 4, 9 and 27 billion parameters).

5.2 First run: guessing and consistency

Table 2 shows the main numbers. Without the graph the best model names the place of birth of a known politician in 52% of the cases and the others in about 20%. For the unknown politicians all models are below 10% and most of these hits are capital cities such as Budapest, Vienna or Tirana. The model recognises the country from the name and answers with its largest city. This is the guessing effect that Cao et al. [2] describe for masked models. It still shows up in current models. The country is easier for the same reason: without the graph the models get it right for 57 to 70% of the unknown people.

With the graph every model answers 96 to 100% of the questions correctly. The answers also stop depending on the wording. Without the graph the same person got the same answer in all three wordings in only 18 to 48% of the cases, with the graph in 98 to 100%.

In the 15 conflicts no model ever answered from its weights. Three models always followed the wrong graph. qwen2.5:7b followed it in 73% of the cases and otherwise answered "unknown". It refused rather than name a place that contradicts what it knows, so its weights showed up as a refusal and never as a wrong fact. The few errors of qwen2.5:7b with the graph outside the conflicts were also refusals, for places that are districts rather than towns (such as Upper East Side) or where the graph names a surprising country (Wikidata places the birth of Kim Jong-il in Russia).

Table 2: First run, share of correct answers. Known and unknown refer to the politicians, closed means without the graph. Same in 3 wordings: the same answer to all three wordings of a question.

	gemma3 12B	gemma4 12B	granite 8B	qwen2.5 7B
Place, known, closed	52	22	20	20
Place, unknown, closed	8	6	1	9
Place, known, graph	100	100	100	96
Place, unknown, graph	100	100	100	100
Same in 3 wordings, closed	48	18	18	35
Same in 3 wordings, graph	100	100	100	98
Conflict, follows graph	100	100	100	73
Conflict, from weights	0	0	0	0

```
Use only the facts below. If the facts do not contain the
answer, answer 'unknown'.
Facts:
(Mark Rutte, place of birth, Detmold)
(Detmold, country, Germany)
Question: Where was Mark Rutte born? Answer with the name
of the place only.
```

Figure 3: A conflict question from the second run with the strict instruction. The true place of birth is The Hague. qwen3.5:27b answered "Detmold" with the strict instruction and with no instruction. With the loose instruction it answered "Mark Rutte was not born in Detmold; he was born in The Hague, Netherlands. The provided facts contain" (cut off by the answer limit).

5.3 Second run: real conflicts and model size

The first run had a weakness. A conflict only tests something if the model actually knows the true answer. For most people in the first run it did not. For the second run I therefore took 100 very well known people from all professions with at least 100 sitelinks each. The conflict test then used only those whose place of birth the model named correctly in both wordings without the graph. Qwen3.5 knew 3 of them at 2B, 15 at 4B, 26 at 9B and 59 at 27B, so the knowledge grows clearly with size. For the 27B model I used the first 40.

For each of these people the graph contained one wrong place on the same continent (plausible) and one on another continent (implausible), for example Nelson Mandela born in Delhi.

Figure 3 shows one of these questions. Each conflict was asked with three instructions: the strict one from above, a loose one ("Answer the question using the facts below and your own knowledge.") and no instruction at all. As controls, a graph with the true facts gave 100% correct answers and an empty list of facts gave "unknown" in all cases.

Figure 4 and Table 3 show the result. The 2B, 4B and 9B models followed the wrong graph in every single case, even with the loose instruction and even when the place was on another continent. Only the 27B model sometimes disagreed: with the strict instruction it followed the graph in 79 of 80 cases, without an instruction in 96%

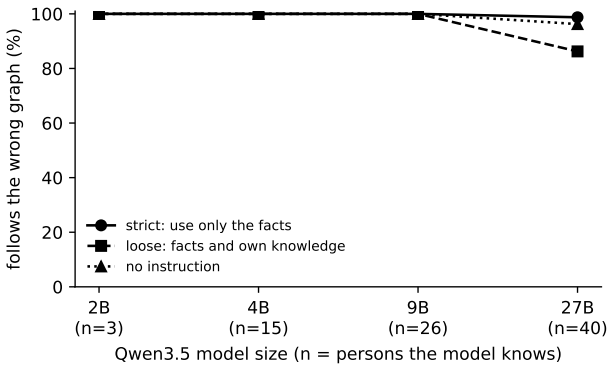


Figure 4: Share of conflict questions in which the model answers with the wrong place given by the graph, by model size and instruction.

Table 3: Second run with Qwen3.5. Known: people whose place of birth the model named correctly in both wordings without the graph. The last three columns give the share of conflict questions in which the model followed the wrong graph, by instruction.

Size	Known	Questions	Strict	Loose	None
2B	3	18	100	100	100
4B	15	90	100	100	100
9B	26	156	100	100	100
27B	40 of 59	240	99	86	96

and with the loose instruction in 86%. In 10 of the 15 cases where it did not follow the graph it gave the true place or a variant of it, for example "Krivoy Rog" for Kryvyi Rih. Three answers said explicitly that the facts were wrong, such as "Mark Rutte was not born in Detmold; he was born in The Hague, Netherlands". Whether the wrong place was plausible made almost no difference (93% against 94%). This fits the size effect that Fouilhé et al. [4] report (Section 2.5). It also fits ClashEval [8], where GPT-4 followed the context more often under a strict prompt than under a loose one.

5.4 What this means for the design

Under a strict instruction the weights stay out of the answer at every size I tested. So the design works in the sense that it was meant to work. The price is that the model does not check the graph either: a wrong entry becomes a wrong answer, even when it is as implausible as Mandela in Delhi. All responsibility for correctness moves to the graph, which brings back exactly the maintenance problem that Deshpande et al. [3] describe. A large model with a loose instruction catches some wrong entries, but only because it lets its weights back in. The Liu Xiaobo example from the introduction, which also comes from the second run, shows why weights are a risky source of facts in the first place: they contain whatever the training data and the provider put into them, while a graph with sources can be checked.

5.5 What I would do next

The experiment answers the first condition of the design for one kind of fact. The second condition, a graph that deserves the trust, is where I see the real work. Three extensions would test it.

Facts that really change. Wrong places of birth are artificial, because nobody's place of birth changes. A more realistic test uses facts that the model learned correctly but that are outdated today, such as who holds an office. The graph would contain the current holder and the model's weights the previous one. This is the case the design is actually meant for. It also avoids the problem that a deliberately wrong graph is easy to recognise for a model that knows the person well.

Time and sources in the graph. If every triple carries the time from which it holds and the source it comes from, the model can be asked to say which fact it used and since when it holds. That would make wrong entries visible in the answer instead of hiding them. It also addresses the maintenance problem of Deshpande et al. [3] from a different side: a fact is closed with an end date instead of being deleted, so corrections never get lost. My design for Arlen already treats the knowledge graph as infrastructure that can be queried across applications and over time [5].

More models and wordings. The second run used one family and one wording of the question. Repeating it with a second family in several sizes and with several wordings of each instruction would show whether the softer behaviour of the 27B model is a matter of size or of one particular model.

5.6 Limitations

The samples are small: 60 people and 15 conflicts in the first run and between 3 and 40 people per size in the second. The second run used a single model family and only one wording for the conflict questions. Answers were compared as strings, so "New York" counts as wrong for a district of New York; a looser comparison changed the results by only 2 to 3 points. The countries in Wikidata are today's countries, which is why Vyborg is in Russia. Finally, models trained for reasoning may be at a disadvantage when they have to answer without a reasoning phase.

6 Conclusion

This report reviewed five papers on a question the field has asked since 2019: can a language model replace a knowledge base? And if it cannot, how should the two work together? Deshpande et al. documented what a production knowledge base costs, above all in maintenance. Petroni et al. found that BERT answers simple factual questions about as well as an automatically built knowledge base. Cao et al. then traced a large part of this to guessing guided by the prompt. AlKhamissi et al. split the question into five properties and found language models lacking in editability, consistency and explainability. Pan et al. drew a map of the ways in which graphs and models can support each other.

The main weakness these papers share is the yardstick. Petroni et al. and AlKhamissi et al. compare language models with an uncurated knowledge base or a plain set of triples, while Pan et al. compare them with an idealised graph that keeps itself up to date. None of them compares against a knowledge base that is maintained over years, which is the kind Deshpande et al. describe.

From this debate I took a narrower design question: should a language model take its facts only from a knowledge graph? My experiment adds one finding. With a strict instruction, models from 2 to 27 billion parameters stayed with the graph in almost every case, even when the graph was wrong and the model knew better. So the first condition of the design holds: the weights stay out. The second condition becomes all the more important, because the model does not catch errors in the graph. Only the largest model sometimes did, mostly when the instruction allowed it to use its own knowledge, which is exactly what the design wants to avoid.

My answer is therefore a qualified yes. The design is sound if the graph is maintained with the care that Deshpande et al. describe and if every fact carries a source. It moves the problem from the model to the graph. That is the better place for it, because a wrong triple can be found and corrected while a wrong weight cannot. Future work should test more model families, several wordings of the instruction and facts that really change over time, such as who currently holds an office, instead of deliberately falsified ones.

Use of Language Models

I used Claude (Anthropic) throughout this work, in several versions over the term, most recently Claude Opus 5.5. Claude helped me create notes while I was reading the papers. It wrote the code for the experiment, which collects the data from Wikidata. The LaTeX graphs were created by it. It was also used to rephrase this paper and to help with grammatical corrections. The models tested in Section 5 (Qwen, Gemma and Granite) are the object of the experiment and were not used for writing.

References

[1] Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A Review on Language Models as Knowledge Bases. Preprint, arXiv:2204.06031.

[2] Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021. Knowledgeable or Educated Guess? Revisiting Language Models as Knowledge Bases. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021), Volume 1: Long Papers*. 1860–1874. doi:10.18653/v1/2021.acl-long.146

[3] Omkar Deshpande, Digvijay S. Lamba, Michel Tourn, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. 2013. Building, Maintaining, and Using Knowledge Bases: A Report from the Trenches. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 1209–1220. doi:10.1145/2463676.2465297

[4] Guilhem Fouilhé, Nicholas Asher, and Philippe Muller. 2026. Memory vs. Context? Influential Factors of Factual Recall in Language Models. arXiv:2609.24238. Accepted to the BlackboxNLP 2026 Reproducibility Challenge at EMNLP 2026.

[5] Tim Kicker. 2026. A Capability-Based, Event-Driven Linux Desktop: Knowledge Graph Architecture and Design Rationale. doi:10.5281/zenodo.19119730 Technical report, Zenodo.

[6] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36, 7 (2024), 3580–3599. doi:10.1109/TKDE.2024.3352100 (cited from the extended version, arXiv:2306.08302v3).

[7] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2463–2473. doi:10.18653/v1/D19-1250

[8] Kevin Wu, Eric Wu, and James Zou. 2024. ClashEval: Quantifying the Tug-of-War between an LLM's Internal Prior and External Evidence. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*. arXiv:2404.10198.