Question-to-Question Retrieval for Hallucination-Free Knowledge Access: An Approach for Wikipedia and Wikidata Question Answering

Santhosh Thottingal
santhosh.thottingal@gmail.com

Abstract

This paper introduces an approach to question answering over knowledge bases like Wikipedia and Wikidata by performing "question-to-question" matching and retrieval from a dense vector embedding store. Instead of embedding document content, we generate a comprehensive set of questions for each logical content unit using an instruction-tuned LLM. These questions are vector-embedded and stored, mapping to the corresponding content. Vector embedding of user queries are then matched against this question vector store. The highest similarity score leads to direct retrieval of the associated article content, eliminating the need for answer generation. Our method achieves high cosine similarity (> 0.9) for relevant question pairs, enabling highly precise retrieval. This approach offers several advantages including computational efficiency, rapid response times, and increased scalability. We demonstrate its effectiveness on Wikipedia and Wikidata, including multimedia content through structured fact retrieval from Wikidata, opening up new pathways for multimodal question answering.

1 Introduction

Question answering (QA) is a task that answers factoid questions using a large collection of documents. In the context of Wikipedia, it is to answer questions beyond the keyword based traditional search. The rise of large language models (LLMs) has opened up new possibilities for building more capable QA systems, yet the challenge remains in ensuring their reliability and avoiding the generation of fabricated or hallucinated responses(Gao et al., 2024). As trustable encyclopedic information is the unique value that Wikipedia provides, providing information as accurate as possible is in its mission.

Retrieval-Augmented Generation (RAG) has become a common approach to address this challenge by combining the knowledge retrieval capabilities with the generative power of LLMs (Lewis et al., 2020). A significant limitation in conventional RAG models stems from the typically low semantic similarity between vector embeddings of natural language questions and typical document passages(Karpukhin et al., 2020). This disparity arises primarily from their distinct structural differences: questions are interrogative, while passages are predominantly declarative. For example, when querying with "Where is Eiffel Tower located?", the question's embedding is compared against passages like "Eiffel Tower is located in Paris." Although "Paris" is semantically crucial, the contrasting sentence structures can lead to low cosine similarity scores, often falling within the range of 0.4-0.7, even for relevant content. This "question-to-passage" comparison, exacerbated by lexical differences and the differing focus of questions and passages, hinders high-precision retrieval, particularly for queries seeking specific information. Addressing this challenge requires strategies like question reformulation, passage summarization, or hybrid approaches that combine semantic search with keyword matching.

Furthermore, while LLMs excel at generating human-like text, the generation step itself introduces the risk of hallucination, where LLMs may create plausible but factually incorrect answers. We argue that a retrieval mechanism that can precisely identify the relevant text and thereby removing the generation step to mitigate hallucination is a practical and useful approach in the context of Wikipedia.

This paper introduces a novel approach to knowledge-based question answering that addresses these limitations by performing "question-to-question" retrieval and avoids the generation step. Instead of embedding document content, we employ an instruction-tuned LLM to generate a comprehensive set of questions for each logical content unit within the knowledge base. These
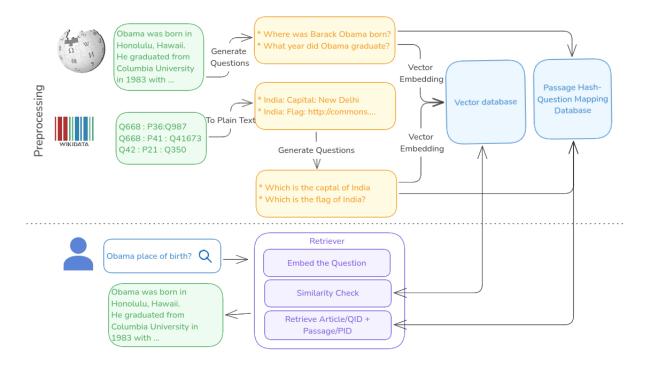
Figure 1: Overall Architecture of Question-Question retrieval and question answering

generated questions are then vector-embedded and stored in a searchable vector store, mapped to the corresponding source document. When a user submits a query, it is also embedded in the same space and a search performed to identify the most similar generated question. For highest matching question, we present the corresponding article and the paragraph with in the article. A similar approach is presented for wikidata where the content is in the form of entity relations, we generate all possible questions and then make the wikidata knowledge graph accessible to a question answer system.

2 Background

The problem of Wikipedia question answering is as follows: Given a factoid question like "Who is the architect of Eiffel Tower?", "How many people died in Chernobil accident?", the retrieval system takes the user to a Wikipedia article's paragraph that answer's the question. Compared to common RAG techniques, there is no LLM based articulation of answer. We assume the question is extractive, in which the answer is available within such a paragraph and does not need to analyze content across multiple paragraphs or articles.

Assume that our collection contains D documents, $d_1, d_2, \cdots, d_D$. We first split each of the Wikipedia article into text passages as the basic retrieval units and get M total passages in our

corpus $C = p_1, p_2, \cdots, p_M$, where each passage p_i can be viewed as a sequence of tokens $w_1^{(i)}, w_2^{(i)}, \cdots, w_{|p_i|}^{(i)}$ Given a question q, the task is to find a p that can answer the question. For the sake of quick and precise mapping between passage and article, assume there is a unique cryptographic hash based on the content of each p that is mapped to a d and saved in a relational database.

A QA system needs to include an efficient retriever component that can select a small set of relevant texts(Chen et al., 2017). Formally speaking, a retriever $R : (q, C) \rightarrow C_\mathcal{F}$ is a function that takes as input a question q and a corpus C and returns a much smaller filter set of texts $C_\mathcal{F} \subset C$, where $|C_\mathcal{F}| = k \ll |C|$. For a fixed k, a retriever can be evaluated in isolation on *top-k* retrieval accuracy, which is the fraction of questions for which $C_\mathcal{F}$ contains a span that answers the question(Karpukhin et al., 2020).

In usual RAG approach, the retrieval process begins with a user's query being converted into a vector representation using a text embedding model. The knowledge base (e.g., a collection of documents or articles) is preprocessed by chunking large bodies of text into smaller, semantically meaningful units such as paragraphs or sentences. These chunks are also embedded into a vector space, often using the same embedding model that was used to embed the user query. The embeddings of these

knowledge base chunks are then stored in a vector database or index to enable fast similarity search. During query time, the embedded user query is compared to each chunk vector and similarity metrics such as cosine similarity are used to identify the most relevant chunks from the knowledge base.

A core problem, as highlighted in the introduction, arises from the inherent structural difference between a user's question and a standard document passage. The closest match is not a plausible answer to our question — instead, it is another question[1]. When a question vector is compared to a passage vector, the comparison relies on shared keywords and latent semantics. However, the semantics of the answer is what defines its usefulness, and comparing it with question embeddings is inherently challenging. This misalignment between the question and passage vector space often leads to low cosine similarity scores even when the retrieved passage contains relevant information. Additionally, the learned embeddings of passages have a degree of freedom that can render arbitrary cosine-similarities(Steck et al., 2024). So, "question-to-passage" vector comparison can make it hard to reliably retrieve the most appropriate content, potentially diminishing the overall performance of the QA system. As illustrated in table 1, for a simple paragraph and questions from it, the usual retrieval score falls below 0.7 and our objective is to maximize that score above 0.9.

Question answering over structured knowledge bases like Wikidata(Vrandečić and Krötzsch, 2014), often involves translating natural language queries into structured queries (e.g., SPARQL) and executing these queries against the KB. Approaches vary from using rule-based systems to more sophisticated neural network models that learn to map natural language to formal query representations(Liu et al., 2024). These techniques heavily rely on the structural aspects of the knowledge base, which is different from our approach. We aim to use the structured data in wikidata as knowledge based for factoid questions without using SPARQL queries.

3 Methodology

We focus our research in this work on improving the retrieval component in open-domain QA. Given a collection of M text passages, the goal of

our retriever is to index all the passages in a low-dimensional and continuous space, such that it can retrieve efficiently the top k passages relevant to the input question for the reader at run-time. Note that M can be very large(e.g., 6 million English Wikipedia articles multiplied by number of paragraphs in each) and k is usually small, such as 1-5. We used two distinct knowledge bases in our experiments: English Wikipedia and Wikidata. The English Wikipedia articles is parsed into a structured format to extract the content of each article. Articles are further divided into logical units, primarily paragraphs. Each paragraph is treated as an independent context unit for which questions are generated. A unique hash is computed for every paragraph which acts as the key to locate it in the original article. This ensures that when a question is retrieved, the associated original context can be found.

The Wikipedia content is primarily in Wikitext markup. It can also be rendered to HTML. But for the purpose of our system, we prepared plain text version of each passage so that it is comprehensible for an LLM. We also removed reference numbers that appear in usual plain text format.

Let $\mathcal{D} = \{d_1, d_2, ..., d_M\}$ represent the knowledge base consisting of M content units.

For each content unit $d_i \in \mathcal{D}$, a set of questions is generated by an LLM, denoted as $Q_i = \{q_{i1}, q_{i2}, ..., q_{in_i}\}$, where n_i is the number of questions generated for d_i. The prompt used for the LLM is given in Appendix **??**. The input to the prompt not only contains the passage text, but contextual information like article title, section titles to resolve coreferences. The prompt also uses few shot prompting to get machine readable output. The concept of document expansion or enrichment by adding queries is common in information retrieval techniques(Nogueira et al., 2019). Here we are not enhancing the document(passage), but uses the questions generated out of it.

Let $\mathcal{E}$ be the embedding function that maps a text input to a vector space, so $\mathcal{E} : \text{Text} \rightarrow \mathbb{R}^d$, where d is the dimensionality of the embedding space. The embedding of a question q_{ij} is denoted as $e_{ij} = \mathcal{E}(q_{ij})$, where $e_{ij} \in \mathbb{R}^d$. The set of all question embeddings is denoted as $\mathcal{V} = \{e_{11}, e_{12}, ..., e_{Mn_M}\}$.

We create an index $\mathcal{I}$ that maps the embeddings e_{ij} to the corresponding content unit d_i via the hash function h, such that $\mathcal{I}(e_{ij}) = h(d_i)$. The hashing function used is SHA-256 with 32 bytes length.

Let q_u be the user's query. The embedding of

[1]Don't use cosine similarity carelessly - Piotr Migdał https://p.migdal.pl/blog/2025/01/dont-use-cosine-similarity

Question	Similarity Score
Where was Barack Obama born?	0.68
Which university did Obama graduate from?	0.71
What year did Obama graduate?	0.70
Where did Obama work as a community organizer?	0.57
Who is the first black president of Harvard Law Review?	0.55
From what years did Obama teach at University of Chicago Law School?	0.75
When was Obama first elected to the Illinois Senate?	0.67
When did Obama run for U.S. Senate?	0.66
Who was Obama's running mate in the 2008 presidential election?	0.63
Who did Obama defeat in the 2008 presidential election?	0.58
Who defeated John McCain in the 2008 presidential election?	0.63
What political party nominated Obama for president?	0.58
Obama birth place	0.64

Table 1: Questions and similarity score for the following passage: *Obama was born in Honolulu, Hawaii. He graduated from Columbia University in 1983 with a Bachelor of Arts degree in political science and later worked as a community organizer in Chicago. In 1988, Obama enrolled in Harvard Law School, where he was the first black president of the Harvard Law Review. He became a civil rights attorney and an academic, teaching constitutional law at the University of Chicago Law School from 1992 to 2004. In 1996, Obama was elected to represent the 13th district in the Illinois Senate, a position he held until 2004, when he successfully ran for the U.S. Senate. In the 2008 presidential election, after a close primary campaign against Hillary Clinton, he was nominated by the Democratic Party for president. Obama selected Joe Biden as his running mate and defeated Republican nominee John McCain.* Embedding model used: text-embedding-004. Dimensions: 798

the user's query is $e_u = \mathcal{E}(q_u)$.

We use cosine similarity $\text{sim}(e_u, e_{ij})$ to compare the user query embedding with each of the generated question embeddings. The best match is chosen using argmax function:

$$j^* = \underset{j}{\text{argmax}} \; \text{sim}(e_u, e_{ij})$$

where j represents the set of all question embeddings in the question vector store.

The content corresponding to the most similar question embedding is retrieved using the hash function:

$$d^* = \mathcal{I}(e_{j^*}) = d_i \; \text{ for some } e_{ij}$$

Let us illustrate this with the same example passage used in table 1. To make it more realistic, let use mimic users queries as incomplete sentences, often missing question words and occasional spelling mistakes. See table 2. As Generated questions are mapped to a passage that an answer it, the effective retrieval similarity for all user queries are same as the Similarity Score. Table 2 illustrates the effectiveness of our approach in handling real world queries and accurately matching the passages.

The context granularity can be refined from paragraph to sentence level through Jaccard similarity matching between the query and individual sentences. When no strong sentence-level match is found, the system defaults to paragraph-level context.

3.1 QA on Wikidata

To incorporate Wikidata, we needed to extract factual information from the knowledge base in a format that is suitable for the question generation task. We wrote a SPARQL query(See Listing 1) that retrieves all subjects, predicates, and objects (triples) associated with each Wikidata entity (QID). The query is designed to be comprehensive in extracting fact triples in a human-readable textual format (e.g., "India: inception: 15 August 1947" or "India: Prime Minister: Narendra Modi (2014-current)"). The conversion of triples to text for querying on structured content was originally discussed in UniK-QA system(Oguz et al., 2022). A unique hash of QID and PID pair is calculated for linking the extracted content with generated questions.

Let Q be a Wikidata item (e.g., Q668 for India), and let $\mathcal{S}_Q$ be the set of statements associated with item Q. Each statement $s \in \mathcal{S}_Q$ can be represented as a tuple $s = (p, v, \Pi)$, where:

- p is the property (PID).

User Query	Most Similar Generated Question	Similarity Score
"Obama's birthplace?"	"Where was Obama born?"	0.91
"France nuclear energy percentage?"	"What percentage of France's electricity is nuclear?"	0.92
"How many people died in chernobyl accident"	"How many people died in chernobyl disaster"	0.97
"How many people died in chernobyl"	"How many people died in chernobyl disaster"	0.97
"Deaths chernobyl accident"	"How many people died in chernobyl disaster"	0.90
"Mayor of paris"	"Who is the current mayor of paris?"	0.93
"longest river in Africa"	"Which is the longest river in Africa?"	0.96
"length of Nile"	"What is the total length of Nile river?"	0.93

Table 2: Example of Question-to-Question Retrieval with Cosine Similarity Scores. Note the typos and omission of question words in user query to mimic real world search scenario

- v is the value.

- Π is the set of qualifiers (optional).

Let $\mathcal{T}$ be the function that maps a statement s to a textual representation t.

$$\mathcal{T} : s \rightarrow t$$

The function $\mathcal{T}$ involves:

- Label retrieval for QIDs and PIDs.

- Formatting of dates and times into human-readable text.

- Concatenating subject, predicate, object and qualifiers.

- For simple cases, t will be in format of "label(Q): label(p): value".

- For statements with qualifiers, the textual representation will be: "label(Q): label(p): value (label(q_1): v_1, label(q_2): v_2, ...)", where q_i is a qualifier and v_i is the value of qualifier.

The set of all text triples generated from a given QID is denoted as $T_Q = \{t_1, t_2, ..., t_n\}$, where n is the number of statements for item Q.

For each textual triple $t_i \in T_Q$, we generate a set of questions using an LLM, denoted by $Q_i = \{q_{i1}, q_{i2}, ..., q_{in_i}\}$. This process can be represented as:

$$Q_i = LLM(t_i)$$

Where n_i is the number of questions generated for t_i. The prompt used for the LLM is given in Appendix **??**

The same embedding function $\mathcal{E}$ is used:

$$\mathcal{E} : \text{Text} \rightarrow \mathbb{R}^d$$

The embedding of a generated question q_{ij} is denoted as $e_{ij} = \mathcal{E}(q_{ij})$. The set of all question embeddings is denoted as $\mathcal{V} = \{e_{11}, e_{12}, ..., e_{Mn_M}\}$, where M is the total number of text triples across all items and n_i is the number of questions for each triple.

We create an index $\mathcal{I}$ that maps the embeddings e_{ij} to the corresponding source triple t_i via the hash function h, such that $\mathcal{I}(e_{ij}) = h(t_i)$.

The rest of the steps are the same as the general approach:

- Let q_u be the user's query, the embedding is $e_u = \mathcal{E}(q_u)$.

- Similarity is calculated as:

$$j^* = \underset{j}{\operatorname{argmax}} \operatorname{sim}(e_u, e_{ij})$$

- The corresponding triple is retrieved:

$$t^* = \mathcal{I}(e_{j^*}) = t_i \text{ for some } e_{ij}$$

Let's consider a few example Wikidata triplets related to "India" in the Subject: Predicate: Object format:

Figure 2: Screenshot from a prototype showing the answer presentation using our approach for the question "length of Nile". Upon entering the query, page navigates to Nile article, and scrolls to this paragraph and then highlights it.

- Subject: Q668 (India)

- Predicate: P571 (inception)

- Object: 1947-08-15T00:00:00Z

The text representation for this is "India: Inception: 15 August 1947". Similarly, we can have textual representations like:

- India: Inception: 15 August 1947

- India: Prime Minister: Narendra Modi (2014-current)

- India: Life expectancy: 62 (1999)

- India: Flag: `https://commons.wikimedia.org/wiki/File:Flag_of_India.svg`

- India: Capital: New Delhi

Here are some example questions an LLM might generate for those Wikidata triplets:

- When was India founded?

- When did India become independent?

- Who is the current prime minister of India?

- Who was the prime minister of India in 2020?

- What was the life expectancy in India in 1999?

- What is the average life span in India around the year 1999?

- Show me the flag of India.

- What does the flag of India look like?

- What is the capital of India?

- Where is the capital of India located?

See C for example comparison and similarity score.

Multimodal content (images, videos, 3D models) becomes searchable through question-based matching of their associated metadata. For instance, Wikidata triples like "Q243:P4896:[filename]" can be transformed into text ("Eiffel Tower: 3D Model: [filename]") and indexed with questions like "What is the 3d model of Eiffel Tower?"—enabling matches with user queries such as "Show me Eiffel tower 3d model." Similarly, Q140:P51:[audio file name] facilitates answers to queries like "How does the lion roar?"

4 Discussion

4.1 Practical Considerations

The increased vector store size—approximately ten-fold due to question-based indexing rather than passage-based—remains manageable given modern vector databases' capability to efficiently search billions of records. While Wikipedia's frequent updates necessitate question regeneration, hash-based tracking enables selective re-indexing of modified passages only. The experimental implementation indexed under 1,000 Wikipedia articles, utilizing llama-3.1-8b-instruct-awq for question generation and baai/bge-small-en-v1.5 (384-dimensional vectors) for embeddings, with processing conducted as a background task.

4.2 Advantages

Our approach offers several distinct advantages over traditional RAG systems:

- Hallucination-Free Responses: The most important advantage is that the response is hallucination-free, since we directly retrieve the original content instead of relying on generated answers.

- High Precision Retrieval: Comparing questions with questions leads to highly accurate results with very high cosine similarity scores.

- Fast Retrieval Time: Due to the absence of LLM calls during inference time, query time is significantly lower than that of RAG based systems. This also implies significant cost savings.

- Multimodality: Retrieval of images, audio and other forms of data are handled seamlessly because we retrieve facts as structured text and treat them uniformly with other textual context.

4.3 Limitations

The system focuses on direct fact retrieval (extractive). Answering more complex questions that require multi-hop reasoning, aggregation, or synthesis might require further advancements. For questions that need more than just one paragraph or fact, the system might fall short. While the diversity of generated questions is high, it might be possible to improve question generation coverage to improve system performance for complex questions. LLMs are efficient only in a small set of languages compared to 300+ languages where Wikipedia exist. We have not evaluated the effectiveness of this approach in languages other than English.

5 Conclusion

In this paper, we introduced a novel question-to-question retrieval approach for open-domain question answering over Wikipedia and Wikidata, which achieves high precision and eliminates the risk of hallucination by avoiding the traditional generation step. Our approach leverages an instruction-tuned LLM to generate comprehensive sets of questions for each content unit in the knowledge base, which are then embedded and indexed for efficient retrieval. This method leads to cosine similarity scores consistently above 0.9 for relevant question

pairs, demonstrating a high degree of alignment between user queries and the indexed content. By directly retrieving the original text and structured data based on the best matching generated questions, our system offers a fast, scalable, and reliable alternative to conventional RAG pipelines, with the added benefit of supporting pseudo-multimodal question answering.

Acknowledgments

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Shicheng Liu, Sina J. Semnani, Harold Triedman, Jialiang Xu, Isaac Dan Zhao, and Monica S. Lam. 2024. Spinach: Sparql-based information navigation for challenging real-world questions. *Preprint*, arXiv:2407.11417.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *Preprint*, arXiv:1904.08375.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022.

UniK-QA: Unified representations of structured and unstructured knowledge for open-domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1535–1546, Seattle, United States. Association for Computational Linguistics.

Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. 2024. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 887–890. ACM.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

A LLM Prompts

A.1 LLM prompt to generate all possible questions for a given passage

You are an expert question generator tasked with creating simple, natural-language questions from a given Wikipedia article that people might typically search on the internet based on a provided text passage.

The input will include:

- Article Title

- Section Title

- Paragraph Text

Guidelines for Question Generation

Use the article and section titles to resolve any ambiguous references in the text Create questions that can be directly answered by the text Prioritize who, what, where, when, and how questions Ensure questions are simple, clear and concise mimicking common search engine query patterns Avoid yes/no questions unless the answer is explicitly stated in the text Avoid generating questions that require external knowledge not present in the text Avoid generating speculative or opinion-based questions Avoid very long questions that may be difficult to understand Include questions about:

- Key concepts

- Specific details

- Important processes

- Significant events or characteristics

- Dates, places, and people

- Questions that can be answered by current subject

Output Format

- Provide a bullet list of questions

- Each question should be a single, complete interrogative sentence

Example Processing

Input:

Article Title: Barack Obama Section Title: Early Life and Education

Obama was born in Honolulu, Hawaii. He graduated from Columbia University in 1983 with a Bachelor of Arts degree in political science and later worked as a community organizer in Chicago. In 1988, Obama enrolled in Harvard Law School, where he was the first black president of the Harvard Law Review. He became a civil rights attorney and an academic, teaching constitutional law at the University of Chicago Law School from 1992 to 2004. In 1996, Obama was elected to represent the 13th district in the Illinois Senate, a position he held until 2004, when he successfully ran for the U.S. Senate. In the 2008 presidential election, after a close primary campaign against Hillary Clinton, he was nominated by the Democratic Party for president. Obama selected Joe Biden as his running mate and defeated Republican nominee John McCain.

Expected Output:

- Where was Barack Obama born?

- What state was Obama born in?

- Which university did Obama graduate from?

- What year did Obama graduate?

- What was Obama's major in college?

- What did Obama do after graduating from Columbia?

- Where did Obama work as a community organizer?

- When did Obama enroll in Harvard Law School?

- Who is the first black president of Harvard Law Review?

- From what years did Obama teach at University of Chicago Law School?

- When was Obama first elected to the Illinois Senate?

- When did Obama run for U.S. Senate?

- Who did Obama compete against in the Democratic primary?

- Who was Obama's running mate in the 2008 presidential election?

- Who did Obama defeat in the 2008 presidential election?

- Who defeated John McCain in the 2008 presidential election?

- What political party nominated Obama for president?

A.2 LLM prompt to generate all possible questions for a Wikidata statement

You are a specialized question generation system. Your task is to convert knowledge triplets into natural questions. Each triplet follows the format:

[Subject] : [Predicate] : [Object]

Guidelines for question generation:

Transform the predicate into an appropriate question word:

- "founded by" -> "Who founded"

- "located in" -> "Where is"

- "born in" -> "When was"

- "invented" -> "What did"

- "composed" -> "Who composed"

Question formation rules:

- Start with the appropriate question word (Who, What, Where, When, How)

- Place the subject appropriately in the question

- End all questions with a question mark

- Maintain proper grammatical structure

- Preserve proper nouns and capitalization

- Remove the predicate's passive voice if present ("founded by" → "Who founded")

Handle special cases:

- Multiple objects: Generate separate questions for each object

- Complex predicates: Break down into simpler components

- Dates: Use "When" for temporal relations

- Locations: Use "Where" for spatial relations

Examples:

Input: "San Francisco : founded by : José Joaquín Moraga, Francisco Palóu"
Output:

- Who founded San Francisco?

Input: "Eiffel Tower : located in : Paris"
Output:

- Where is the Eiffel Tower?

Input: "JavaScript : created by : Brendan Eich"
Output:

- Who created JavaScript?

Input: "Theory of Relativity : developed in : 1905"
Output:

- When was the Theory of Relativity developed?

Always ensure questions are:

- Grammatically correct

- Natural sounding

- Unambiguous

- Focused on a single piece of information

- Answerable using the information in the original triplet

B SPARQL Query for getting all statements for a given Qitem

C Example question-question matching for Wikidata

User Query	Most Similar Generated Question	Cosine Similarity Score
"When was India established?"	"When was India founded?"	0.97
"Who leads India now?"	"Who is the current prime minister of India?"	0.94
"What was life expectancy in India back then?"	"What was the life expectancy in India in 1999?"	0.92
"Show Indian flag"	"Show me the flag of India."	0.95
"India's capital city"	"What is the capital of India?"	0.96
"Tell me the time of Indian independence"	"When did India become independent?"	0.95
"Who is the PM of India now"	"Who is the current prime minister of India?"	0.96
"What is the flag of India like?"	"What does the flag of India look like?"	0.92
"Where is the capital of India located"	"Where is the capital of India located?"	0.97
"Average life span of India?"	"What is the average life span in India around the year 1999?"	0.93

Table 3: Example of Wikidata Question-to-Question Retrieval with Cosine Similarity Scores

Listing 1: SPARQL Query for getting all statements for a given Qitem

```
SELECT
  ?property
  ?propertyLabel
  ?statementValue
  ?statementValueLabel
  ?statementValueImage
  ?qualifierProperty
  ?qualifierPropertyLabel
  ?qualifierValue
  ?qualifierValueLabel
  ?unitOfMeasure
  ?unitOfMeasureLabel
  ?statementRankLabel
WHERE {
  VALUES ?item {wd:\${qid}}

  # Main statement pattern
  ?item ?propertyPredicate ?statement .
  ?statement ?statementPropertyPredicate ?statementValue .

  # Property and statement property predicates
  ?property wikibase:claim ?propertyPredicate .
  ?property wikibase:statementProperty ?statementPropertyPredicate .

  # Rank of the statement
  ?statement wikibase:rank ?statementRank .
  BIND(
    IF(?statementRank = wikibase:NormalRank, "Normal",
      IF(?statementRank = wikibase:PreferredRank, "Preferred",
        IF(?statementRank = wikibase:DeprecatedRank, "Deprecated", "Unknown")
      )
    ) AS ?statementRankLabel
  )

 # Optional image
  OPTIONAL {
    ?statementValue wdt:P18 ?statementValueImage .
  }

  # Optional qualifiers
  OPTIONAL {
    ?statement ?qualifierPredicate ?qualifierValue .
    ?qualifierProperty wikibase:qualifier ?qualifierPredicate .
  }

  # Optional unit of measure for quantities
  OPTIONAL {
    ?statement ?statementValueNodePredicate ?valueNode .
    ?valueNode wikibase:quantityUnit ?unitOfMeasure .
  }

   # Labels for properties, values, qualifiers, and units
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "\${language},␣en"  .
    ?property rdfs:label ?propertyLabel .
    ?statementValue rdfs:label ?statementValueLabel .
    ?qualifierProperty rdfs:label ?qualifierPropertyLabel .
    ?qualifierValue rdfs:label ?qualifierValueLabel .
    ?unitOfMeasure rdfs:label ?unitOfMeasureLabel .
  }
}
ORDER BY ?property ?statementValue ?qualifierProperty ?qualifierValue
```