Language Technology QA System - Final Report

Group 3
Martijn Prikken and Sijbren van Vaals and Sander Beyen
m.prikken.1@student.rug.nl
s.j.van.vaals@student.rug.nl
s.beyen@student.rug.nl

Abstract

We have implemented an end-to-end Question Answering system designed to address user information needs related to video games by leveraging the rich structured data available in Wikidata. Our system incorporates various NLP approaches to extract entities and properties from user questions and combines them to enable the construction of SPARQL queries for answering those questions.
We evaluated our system on a test set of 50 held-out questions and measured an overall accuracy of 65%. The model performs well on questions of the form 'Who/what is the X of Y?' including alternative phrasings as well as Yes/No questions. However, there is room for improvement for question types involving 'when' and phrases where the primary property of interest is expressed as a verb.

All of our code can be found on our repository: https://github.com/sijbrenvv/TT

1 Introduction

Translating a user's natural language question into a machine-understandable query is a challenging task. The primary obstacle is converting the user's information needs into a format suitable for evaluation using standard Semantic Web query processing and inferencing techniques (Usbeck et al., 2017).

Our project draws inspiration from the field of Question Answering over Linked Data, which is considered a scientific 'shared task'. This means multiple groups have developed systems that are evaluated on new, unknown data allowing for a comprehensive comparison of various approaches and their respective results.

The general structure of standard Question Answering systems tends to follow a defined pattern. First, the user formulates a question in natural language. Subsequently, the system analyzes and processes the request and retrieves the answer within a text, knowledge graph, or language model, typically in the form of a name, date, number, or list.

Wikipedia serves as a popular knowledge graph, with two prominent sources: DBpedia and Wikidata (Vrandečić and Krötzsch, 2014). DBpedia is a database of Wikipedia pages, while Wikidata contains essential facts extracted from a shared database, which are used as building blocks for creating Wikipedia pages. In our approach, we utilize Wikidata as a key resource, as outlined in section 3.1.

The process of querying a knowledge graph is facilitated through Open Linked Data and the Resource Description Framework (RDF). Linked Data serves as a methodology for publishing data on the web and establishing connections between different data sources. RDF acts as the language of Open Linked Data and knowledge graphs (Berners-Lee et al., 2001). RDF data is structured as triplets ⟨Item, Property, Value⟩, and can be queried with SPARQL, using the SPARQL endpoint, a web service that enables us to access RDF data.

For our project, we have built an end-to-end Question Answering system for video/computer games. The system's primary objective is to transform natural language questions into SPARQL queries, enabling the retrieval of relevant information from Wikidata. The performance of our system has been evaluated on unseen test data.

2 Data

As means of preparation, we were given an outline of what types of questions our system is expected to be able to handle. This includes, but is not limited to, questions of the following types:
1. What/Who is/are the X of Y?
2. Other questions with a single answer
3. List-questions
4. Yes/No questions
5. Count Questions
6. Questions that require access to qualified statements

After development, we evaluated the performance of our QA system on test data, comprised of 50 questions related to video games. Following the system's generation of answers to these questions, we assigned scores of 0 to incorrect answers, 0.5 to partially correct answers, and scores of 1 to completely correct answers. This method allowed us to accurately quantify the effectiveness of our system.

3 Methodology

3.1 Approach

As mentioned before, our QA system leverages the Wikidata database to facilitate the process of answering questions. Our program parses the input question and extracts the information needed for building the query by identifying the relevant entities and properties. This query is then sent to Wikidata. Finally, the response is processed and transformed into the answer to the user's question.

Our approach consists of a few key components. Firstly, we query all of the properties used in Wikidata and store them in two separate lists with their corresponding labels. In this process, we do omit properties that contain the string 'ID', since these will never be used. Next, we use the msmarco-distilbert-cos-v5 (Reimers and Gurevych, 2019) sentence transformer model. Even though the model is trained for semantic search tasks, we observed consistent and accurate results in retrieving the correct properties. Additionally, we incorporate the "spacy en_core_web_trf" model provided by the spaCy library (Honnibal and Montani, 2017). It is worth noting that these preparations are completed before any questions are asked to the system.

Upon receiving a question, our application completes the parsing process using the spaCy parser. To identify entities within the question, we use spaCy's entity recognition capabilities. In case this yields no results, we use capital letters. In the next section, we will delve deeper into the specific techniques we employ.

We identify properties relevant to the given question, by using the same sentence transformer mentioned earlier to encode the question. Afterward, we calculate the dot product between the query and all of the labels. Then, the system tries the 10 most similar labels.

In order to identify yes/no questions, we use a list of words that serve as indicators for that type of question. We identify the entity and check if there is a word in the question that is used on the page of that entity.

4 Architecture

4.1 Entities

We make use of the spaCy model to identify entities. However, if this yields no results we have an alternative strategy to look for patterns that could indicate the presence of an entity. The system searches for words that start with a capital letter, excluding the first word of the question. Additionally, in case there is a word that starts with a capital letter that is followed by the word 'of' and the subsequent word is also capitalized, the program takes the whole sequence of words as the entity. By applying these rules our application is able to extract entities such as "God of War" and "League of Legends".

Once we have identified the entities, we send a request to the Wikidata database. This request retrieves all possible Q-values related to the identified entities. Next, we iterate through the list of returned entities and attempt to find an answer to the question. We systematically try each entity until a suitable answer is found. This process allows us to explore different possibilities and increase the chances of obtaining a correct response.

4.2 Properties

To identify suitable properties for the given question, we leverage the pre-stored list of property labels and their corresponding identifiers that we obtained during the initialization phase. This list allows us to quickly access the relevant properties without the need for additional requests to the wikidata database.

By iterating through these selected property labels and combining them with the identified entities, we generate a set of candidate queries. Each candidate query represents a combination of an entity and a property that we can use to retrieve potential answers from Wikidata.

This approach enables us to efficiently explore different combinations of entities and properties, increasing the chances of finding the correct answer to the question while minimizing the number of external requests to the Wikidata database.

4.3 Yes/No questions

In order to recognize Yes/No questions we use a predefined list of words that commonly indicate this question type: ["Is", "Was", "Does", "Can", "Are", 'Did']. If the question begins with any of these words, we classify it as a Yes/No question. The system follows a similar process for identifying the entity as in a normal question. However, in the case of Yes/No questions, it does not specifically identify a property. If no entity is found, the system returns "no" as the answer. If an entity is found, the system constructs the following SPARQL query to retrieve all values associated with that entity from wikidata:

SELECT ?answerLabel WHERE { wd:' + ID1_1 + ' ?property ?answer SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }}

Here, ID_1 represents the entity found in the question. The query retrieves all values for the entity, and the system iterates through each value to check if any of them, except for the entity itself, are mentioned in the question. If a match is found, the system answers "yes"; otherwise, it answers "no".

While this approach generally performs well on normal question formats and handles cases where the property is not explicitly mentioned, it has a couple of limitations. Firstly, it only considers one entity, potentially missing out on the correct entity if it is not the first one in the list. Secondly, it may incorrectly answer "yes" for unconventional question structures. Examples include questions like "Is Minecraft's genre Notch?" or "Is Elden Ring's country of origin Simplified Chinese?"

4.4 Producing the answer for non-Yes/No questions

When our system encounters a question beginning with the phrase "How many", we collect all the potential answers and store them in a list. For instance, if the question is "How many composers does Undertale have?" and the system identifies "Toby Fox" as the answer, it adds "Toby Fox" to the list. Finally, we return the length of this list as the answer. In this case, the answer would be 1.

However, it's important to note that we exclude certain types of questions that follow the pattern "How many" followed by specific words like "followers," "units," or "copies." For these types of questions, the system should return the answer obtained through a separate request rather than counting the number of answers.

For question types other than "How many", our system retrieves the values returned by the request sent to Wikidata without any further modifications. However, we apply two specific editing rules to ensure the answers are presented in a more readable format.

1. Multiple Answers: If the system receives multiple answers, we separate them with a comma to provide a clear distinction between each answer. This formatting helps improve the readability and organization of the answer for the user.

2. Date Formatting: When the answer obtained from Wikidata contains a date, we remove the "00:00:00Z" part that is included by default. This adjustment eliminates unnecessary details related to the time portion of the date, making the answer more concise.

5 Results and evaluation

We can quantitatively evaluate our system's performance by measuring the coverage, which is the percentage of questions answered, as well as the accuracy of the output.

Our system found a non-null answer to 47 out of 50 questions, giving us a coverage percentage of 94%. Additionally, according to our own evaluation, it scored a total of 32.5 points on the test set, giving us an overall accuracy percentage of 65%.

To gather a more nuanced understanding of our model's strengths and weaknesses, we divided the questions into 6 different types and looked at accuracy by category:

Question Type	Accuracy
Who is/was/are the X of Y?	93%
Alternative phrasings	71%
Yes/No?	66%
How Many?	50%
When?	33%
Property is expressed as a verb	20%

Table 1: Questions sorted by type and accuracy percentage.

For the qualitative analysis, we evaluate our program's performance based on the different question types by aiming to understand the underlying reasons why certain questions were or were not answered correctly. This way we can identify both the strengths and the limitations of our implementation, giving us insight into areas for improvement.

5.1 Who is the X of Y? (93% accuracy)

First, we will take a look at the question types that our system performed well on, ordered by accuracy percentage. It makes sense for the model to perform best on 'Who is the X of Y' questions since they are the most basic form of questions we covered during the lectures and assignments. Nearly all of these questions had relatively unambiguous structures which allowed our system to derive the correct property with comfortably high probabilistic margins as well as find the proper corresponding entity.

Only the question "Who is the main enemy of Mario?" was answered incorrectly, since it provided the entire list of Mario's enemies not distinguishing one as more important than the others. The output for this question is shown in figure 1 in the appendix.

This could be fixed by making our system recognize that usage of the word 'main' indicates a user's desire for a singular answer and if applicable make use of qualified statements. As a backup strategy, we could also make the system pick just the first entry of the list of property items in the Wikidata, which in this specific case would have granted the correct answer as well.

5.2 Alternative phrasings (71% accuracy)

Second, in the category of questions that the system handles well are questions of the form "Who is the X of Y" with alternative phrasings, such as putting the verb after the intended property or starting the sentence with the word 'on'. It returned the correct answer for 5 out of 7 of these questions. Our model was not caught off guard by slight changes in word ordering or syntactic structure.

However, an interesting mistake can be found in our system's processing of the question "Who are four of the main characters of 'The Witcher 3: Wild Hunt' game?". As can be seen in figure 2, our program did not manage to identify the correct entity. It detected the word 'four' instead of 'The Witcher 3: Wild Hunt'.

This is a result of us merely relying on spaCy for our entity recognition. As proven by our system it performs quite well but of course, there are exceptions and we can improve the performance of our model by taking precautions to handle these rare situations. We could implement either an entity ruler or perform an additional, alternative entity check in case our program returns 'null' as output.

5.3 Yes/No questions (66% accuracy)

The last category of questions for which our model performed above chance is Yes/No questions, with an accuracy of 6 out of 9. Our system was prepared for the types of phrasings that could be found in the test set but it did not manage to find the correct properties in all 3 of its mistakes and, as can be seen in figure 3, even returned the incorrect entity for the question "Can FIFA 20 be played on Nintendo Switch?"

5.4 "How Many?" questions (50% accuracy)

Another interesting category of the test set that our system scored 50% accuracy on are questions of the type 'How many?' since they contain a various range of errors. As shown in figure 4, the question "How many awards has Destiny 2 (2017) received?" was answered incorrectly because of property recognition, with the correct 'award received' property being assigned a lower score than 'nominated for' by a minuscule probabilistic margin of 0.008!

The system also struggled to detect correct entities in Mario-related questions because, firstly, there are almost 50 different Mario games as well as movies, artists, and historical figures with the same name.

Furthermore, question number 43 was quite interesting: “How many parts does the Ratchet & Clank have?” We did not understand why the question seems to have a spelling error in it. Our system returned ’null’ to this question because it detected ‘the Ratchet & Clank’ as the entity instead of “Ratchet & Clank”. We tested the output for alternate phrasings such as ‘How many parts does Ratchet & Clank have?” and “How many parts does the Ratchet & Clank series have”? For both of these alternatives, it did find the correct entity.

Lastly, it also answered the question “How many social media followers did Just Dance have in 2021?” incorrectly despite detecting the correct entity as well as the right property. This error is a direct result of the way we coded the system to deal with "how many" type questions. Our system was not prepared for the bigram ’How many’ to be followed by ’social media followers’, only ’followers’ and therefore incorrectly returned the length of the list instead.

5.5 "When?" questions (33% accuracy)

A category of questions that our system was not prepared for, was phrases starting with the word ‘When’. In our output for the question “When was Electronic Arts founded?” we found the answer “Trip Hawkins”. At first glance, this seems odd but he is in fact the founder of EA. Our program was not able to connect the words ‘when’ and ‘founded’ to the right property, namely ‘inception’.

We also found a rather intriguing anomaly when reviewing the output of our system for the second time which can be seen in figure 5. Initially, our system had incorrectly answered “Wimbledon” to the question “When was Lara Croft born”, but after querying the question manually it did return the correct answer, the 14th of February 1968, making correct use of the property ‘date of birth’ instead of ‘place of birth’. We suspect this error is a result of the error message the Wikidata request returns because too many requests are sent. Instead, it assigned the next highest similarity as an answer, which is ‘place of birth’. In essence, this error only occurs when you feed the system a long list of questions, like the test set.

Figure 6 showcases the only question containing the word ‘when’ that our model answered correctly: "Elden Ring, when was the game published?”. It is worthwhile to note that the assigned property probability margin between ‘publication date’ and ‘publisher’ was 0.15 and therefore quite steep, allowing the system to select the right property with significant certainty.

5.6 Property is expressed as a verb (20% accuracy)

Last and also worst, is the output for questions where the property is expressed as a verb. Our system failed to assign the correct property on 4 out of 5 occasions and also could not find an entity for the game FIFA 19.

6 Conclusion

In this project, we aimed to develop an end-to-end Question Answering system for video/computer games, making use of the vast amount of structured data available in Wikidata. Through our methodology and evaluation, we believe we have made significant progress in achieving this objective.

We employed a wide range of techniques and tools to extract entities and properties from user questions, construct SPARQL queries, and retrieve relevant information from Wikidata. Our approach utilized the msmarco- distilbert-cos-v5 sentence transformer for properties, the spaCy transformer pipeline for entities, and the querying capabilities of the Wikidata SPARQL endpoint.

During the evaluation process, we measured the performance of our QA system on a held-out test set consisting of 50 diverse questions related to video games. We used a scoring system that assigned scores of 0, 0.5, and 1 to incorrect answers, partially correct answers, and completely correct answers, respectively. The results showed an overall accuracy rate of 65%, indicating good effectiveness.

We identified several strengths of our methods for entity and property recognition in regards to answering questions of the form ’Who is the X of Y?’ including alternative phrasings as well as Yes/No questions. However, our evaluation also revealed the limitations of our approach. Our program encountered challenges in identifying properties that

were not explicitly mentioned in the question, primarily when they were expressed as a verb, or when the word 'when' was present in the sentence.

Moving forward there are several avenues for improvement. We could enhance our entity recognition through an entity ruler and experiment with integrating LLMs into our property classification pipeline. Moreover, we could also broaden the scope of our system to either other domains or by expanding the number of question types the program is able to handle.

In conclusion, our system demonstrates potential in effectively addressing users' information requirements regarding video games, offering promising results and contributing to advancements in the field of Question Answering systems.

7 Accountability

The development of our QA system involved a collaborative effort, with Martijn primarily responsible for writing the code, Sijbren handling the testing phase, and Sander taking the lead in report writing and documentation.

References

Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The semantic web. *Scientific american*, 284(5):34–43.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th open challenge on question answering over linked data (QALD-7). In *Semantic Web Evaluation Challenge*, pages 59–69. Springer International Publishing.

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

A Appendix

Figure 1: Output for "Who is the main enemy of Mario?". Correct recognition of entity and property but the answer is still incorrect because of improper formatting.

```
print(qa("Who is the main enemy of Mario?"))
```

```
Property probability:  [('enemy', 0.554635763168335), ('antagonist of', 0.38473016023635864), ('killed by', 0.299832105636
5967), ('game mode', 0.2854388654232025), ('conflict', 0.28515899181365967), ('opponent during disputation', 0.27042791247
36786), ('game artist', 0.24296510219573975), ('number of dislikes', 0.24269931018352509), ('antagonist muscle', 0.2416617
2742843628), ('in opposition to', 0.2371070235967636)]

All possible Entities:  [{'id': 'Q12379', 'title': 'Q12379', 'pageid': 13960, 'display': {'label': {'value': 'Mario', 'lan
guage': 'en'}, 'description': {'value': 'fictional character in the Mario video game franchise', 'language': 'en'}}, 'repo
sitory': 'wikidata', 'url': '//www.wikidata.org/wiki/Q12379', 'concepturi': 'http://www.wikidata.org/entity/Q12379', 'labe
l': 'Mario', 'description': 'fictional character in the Mario video game franchise', 'match': {'type': 'label', 'language
': 'en', 'text': 'Mario'}}, {'id': 'Q3362622', 'title': 'Q3362622', 'pageid': 3204847, 'display': {'label': {'value': 'Mar
io', 'language': 'en'}, 'description': {'value': 'male given name', 'language': 'en'}}, 'repository': 'wikidata', 'url':
'//www.wikidata.org/wiki/Q3362622', 'concepturi': 'http://www.wikidata.org/entity/Q3362622', 'label': 'Mario', 'descriptio
n': 'male given name', 'match': {'type': 'label', 'language': 'en', 'text': 'Mario'}}, {'id': 'Q4803535', 'title': 'Q48035
35', 'pageid': 4590001, 'display': {'label': {'value': 'Mario', 'language': 'en'}, 'description': {'value': 'media franchi
se', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q4803535', 'concepturi': 'http://www.wi
kidata.org/entity/Q4803535', 'label': 'Mario', 'description': 'media franchise', 'match': {'type': 'label', 'language': 'e
n', 'text': 'Mario'}}, {'id': 'Q177131', 'title': 'Q177131', 'pageid': 176761, 'display': {'label': {'value': 'Mario', 'la
nguage': 'en'}, 'description': {'value': 'American R&B singer (born 1986)', 'language': 'en'}}, 'repository': 'wikidata',
'url': '//www.wikidata.org/wiki/Q177131', 'concepturi': 'http://www.wikidata.org/entity/Q177131', 'label': 'Mario', 'descr
iption': 'American R&B singer (born 1986)', 'match': {'type': 'label', 'language': 'en', 'text': 'Mario'}}, {'id': 'Q20097
7', 'title': 'Q200977', 'pageid': 197675, 'display': {'label': {'value': 'Moses the Black', 'language': 'en'}, 'descriptio
n': {'value': 'monk, priest and martyr in Egypt', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org
/wiki/Q200977', 'concepturi': 'http://www.wikidata.org/entity/Q200977', 'label': 'Moses the Black', 'description': 'monk,
priest and martyr in Egypt', 'match': {'type': 'alias', 'language': 'en', 'text': 'mario'}, 'aliases': ['mario']}, {'id':
'Q1227379', 'title': 'Q1227379', 'pageid': 1168705, 'display': {'label': {'value': 'Mario Nuzzi', 'language': 'en'}, 'desc
ription': {'value': 'Italian painter (1603-1673)', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.or
g/wiki/Q1227379', 'concepturi': 'http://www.wikidata.org/entity/Q1227379', 'label': 'Mario Nuzzi', 'description': 'Italian
painter (1603-1673)', 'match': {'type': 'label', 'language': 'en', 'text': 'Mario Nuzzi'}}, {'id': 'Q979474', 'title': 'Q9
79474', 'pageid': 929170, 'display': {'label': {'value': 'Marco Pino', 'language': 'en'}, 'description': {'value': 'Italia
n painter (1521-1583)', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q979474', 'conceptur
i': 'http://www.wikidata.org/entity/Q979474', 'label': 'Marco Pino', 'description': 'Italian painter (1521-1583)', 'match
': {'type': 'alias', 'language': 'en', 'text': 'Mario'}, 'aliases': ['Mario']}]

Found entity:  Mario

Answer found:
Wario, Hammer Bro., Bowser, Goomba, Chain Chomp, Kamek, Bullet Bill, Doctor Eggman, King Bob-omb, Koopa Troopa, Bowser J
r., Wart, Dark Bowser, Tatanga, Larry Koopa, Ludwig von Koopa, Boom Boom
```

Relevant section: 5.1

Figure 2: Output for "Who are four of the main characters of "The Witcher 3: Wild Hunt' game?". This example showcases the limitations of the spaCy entity recognizer. This issue could be addressed through the implementation of an entity ruler.

```
print(qa("Who are four of the main characters of the 'The Witcher 3: Wild Hunt' game?"))
```

```
Property probability:  [('list of characters', 0.3311046361923218), ('characters', 0.26262155175209045), ('name of the cha
racter role', 0.2377474457025528), ('Mertens-Pack 3 Number', 0.22878704965114594), ('character role', 0.2158176749944687),
('RubyGems gem', 0.21070407330989838), ('ordeal by', 0.20927521586418152), ('game mode', 0.2087160348892212), ('collection
creator', 0.20764055848121643), ('narrative role', 0.20588484406471252)]

All possible Entities:  [{'id': 'Q202', 'title': 'Q202', 'pageid': 344, 'display': {'label': {'value': '4', 'language': 'e
n'}, 'description': {'value': 'natural number', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org/w
iki/Q202', 'concepturi': 'http://www.wikidata.org/entity/Q202', 'label': '4', 'description': 'natural number', 'match': {'
type': 'alias', 'language': 'en', 'text': 'four'}, 'aliases': ['four']}, {'id': 'Q462471', 'title': 'Q462471', 'pageid': 4
36205, 'display': {'label': {'value': 'power forward', 'language': 'en'}, 'description': {'value': 'position in the sport
of basketball', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q462471', 'concepturi': 'htt
p://www.wikidata.org/entity/Q462471', 'label': 'power forward', 'description': 'position in the sport of basketball', 'mat
ch': {'type': 'alias', 'language': 'en', 'text': 'four'}, 'aliases': ['four']}, {'id': 'Q925222', 'title': 'Q925222', 'pag
eid': 875615, 'display': {'label': {'value': 'Four', 'language': 'en'}, 'description': {'value': 'commune in Isère, France
', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q925222', 'concepturi': 'http://www.wikid
ata.org/entity/Q925222', 'label': 'Four', 'description': 'commune in Isère, France', 'match': {'type': 'label', 'language
': 'en', 'text': 'Four'}}, {'id': 'Q18003476', 'title': 'Q18003476', 'pageid': 19538539, 'display': {'label': {'value': 'F
our', 'language': 'en'}, 'description': {'value': '2014 studio album by One Direction', 'language': 'en'}}, 'repository':
'wikidata', 'url': '//www.wikidata.org/wiki/Q18003476', 'concepturi': 'http://www.wikidata.org/entity/Q18003476', 'label':
'Four', 'description': '2014 studio album by One Direction', 'match': {'type': 'label', 'language': 'en', 'text': 'Four
'}}, {'id': 'Q4031568', 'title': 'Q4031568', 'pageid': 3842440, 'display': {'label': {'value': '4', 'language': 'en'}, 'de
scription': {'value': '2004 film by Ilya Khrzhanovsky', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikida
ta.org/wiki/Q4031568', 'concepturi': 'http://www.wikidata.org/entity/Q4031568', 'label': '4', 'description': '2004 film by
Ilya Khrzhanovsky', 'match': {'type': 'alias', 'language': 'en', 'text': 'Four'}, 'aliases': ['Four']}, {'id': 'Q3413449',
'title': 'Q3413449', 'pageid': 3252338, 'display': {'label': {'value': 'four', 'language': 'en'}, 'description': {'value':
'playing card', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q3413449', 'concepturi': 'ht
tp://www.wikidata.org/entity/Q3413449', 'label': 'four', 'description': 'playing card', 'match': {'type': 'label', 'langua
ge': 'en', 'text': 'four'}}, {'id': 'Q1703333', 'title': 'Q1703333', 'pageid': 1636622, 'display': {'label': {'value': 'Fo
ur', 'language': 'en'}, 'description': {'value': 'fourth Bloc Party album', 'language': 'en'}}, 'repository': 'wikidata',
'url': '//www.wikidata.org/wiki/Q1703333', 'concepturi': 'http://www.wikidata.org/entity/Q1703333', 'label': 'Four', 'desc
ription': 'fourth Bloc Party album', 'match': {'type': 'label', 'language': 'en', 'text': 'Four'}}]

Found entity:  four

Answer found:
None
```

Relevant section: 5.2

Figure 3: Output for "Can FIFA 20 be played on Nintendo Switch?". This example shows both incorrect property and entity recognition at the same time.

```
print(qa("Can FIFA 20 be played on Nintendo Switch?"))
```

```
Property probability:  [('FIFA country code', 0.3443134129047394), ('surface played on', 0.33018267154693604), ('Discord S
tore game SKU', 0.32079583406448364), ('game mode', 0.3182792365550995), ('Mexican video game rating category', 0.29368132
35282898), ('exact match', 0.28766709566116333), ('ground level 360 degree view URL', 0.2853683531284332), ('playing hand
', 0.2782096266746521), ('Portable Game Notation', 0.27443215250968933), ('Professional shogi player number', 0.2737616896
6293335)]

All possible Entities:  [{'id': 'Q19610114', 'title': 'Q19610114', 'pageid': 21209510, 'display': {'label': {'value': 'Nin
tendo Switch', 'language': 'en'}, 'description': {'value': 'hybrid video game console', 'language': 'en'}}, 'repository':
'wikidata', 'url': '//www.wikidata.org/wiki/Q19610114', 'concepturi': 'http://www.wikidata.org/entity/Q19610114', 'label':
'Nintendo Switch', 'description': 'hybrid video game console', 'match': {'type': 'label', 'language': 'en', 'text': 'Ninte
ndo Switch'}}, {'id': 'Q65458682', 'title': 'Q65458682', 'pageid': 65092891, 'display': {'label': {'value': 'Nintendo Swit
ch Lite', 'language': 'en'}, 'description': {'value': 'handheld game console developed by Nintendo', 'language': 'en'}}, '
repository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q65458682', 'concepturi': 'http://www.wikidata.org/entity/Q654586
82', 'label': 'Nintendo Switch Lite', 'description': 'handheld game console developed by Nintendo', 'match': {'type': 'lab
el', 'language': 'en', 'text': 'Nintendo Switch Lite'}}, {'id': 'Q110875040', 'title': 'Q110875040', 'pageid': 105896976,
'display': {'label': {'value': 'Nintendo Switch Sports', 'language': 'en'}, 'description': {'value': '2022 sports simulati
on video game developed by Nintendo', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q11087
5040', 'concepturi': 'http://www.wikidata.org/entity/Q110875040', 'label': 'Nintendo Switch Sports', 'description': '2022
sports simulation video game developed by Nintendo', 'match': {'type': 'label', 'language': 'en', 'text': 'Nintendo Switch
Sports'}}, {'id': 'Q30943865', 'title': 'Q30943865', 'pageid': 32553138, 'display': {'label': {'value': 'Nintendo Switch O
nline', 'language': 'en'}, 'description': {'value': 'suite of online services for the Nintendo Switch', 'language': 'en
'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q30943865', 'concepturi': 'http://www.wikidata.org/entity/Q
30943865', 'label': 'Nintendo Switch Online', 'description': 'suite of online services for the Nintendo Switch', 'match':
{'type': 'label', 'language': 'en', 'text': 'Nintendo Switch Online'}}, {'id': 'Q30917688', 'title': 'Q30917688', 'pageid
': 32527002, 'display': {'label': {'value': 'Nintendo Switch Pro Controller', 'language': 'en'}, 'description': {'value':
'alternative controller for the Nintendo Switch', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org
/wiki/Q30917688', 'concepturi': 'http://www.wikidata.org/entity/Q30917688', 'label': 'Nintendo Switch Pro Controller', 'de
scription': 'alternative controller for the Nintendo Switch', 'match': {'type': 'label', 'language': 'en', 'text': 'Ninten
do Switch Pro Controller'}}, {'id': 'Q28869610', 'title': 'Q28869610', 'pageid': 30530557, 'display': {'label': {'value':
'Nintendo Switch system software', 'language': 'en'}, 'description': {'value': 'operating system', 'language': 'en'}}, 're
pository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q28869610', 'concepturi': 'http://www.wikidata.org/entity/Q28869610
', 'label': 'Nintendo Switch system software', 'description': 'operating system', 'match': {'type': 'label', 'language': '
en', 'text': 'Nintendo Switch system software'}}, {'id': 'Q55218324', 'title': 'Q55218324', 'pageid': 55266241, 'display':
{'label': {'value': 'Nintendo Switch game card', 'language': 'en'}, 'description': {'value': 'game card format used by Nin
tendo Switch', 'language': 'en'}}, 'repository': 'wikidata', 'url': '//www.wikidata.org/wiki/Q55218324', 'concepturi': 'ht
tp://www.wikidata.org/entity/Q55218324', 'label': 'Nintendo Switch game card', 'description': 'game card format used by Ni
ntendo Switch', 'match': {'type': 'label', 'language': 'en', 'text': 'Nintendo Switch game card'}}]

Found entity:  Nintendo Switch

Answer found:
no
```

Relevant section: 5.3

Figure 4: Output for "How many awards has Destiny 2 (2017) received?". This example shows incorrect property assignment through an astronomically small probabilistic margin between the 2 best options.

```
print(qa("How many awards has Destiny 2 (2017) received?"))
```

```
Property probability:  [('nominated for', 0.41557973623275757), ('award received', 0.40785086154937744), ('nominated by',
0.37071362137794495), ('winner', 0.3561249375343323), ('nominee', 0.3310772776603699), ('category for recipients of this a
ward', 0.3051200807094574), ('release of', 0.2916550040245056), ('votes received', 0.29086315631866455), ('has goal', 0.28
758540749549866), ('trophy awarded', 0.2742938995361328)]
```

Relevant section: 5.4

Figure 5: Output for "When was Lara Croft born?". We believe this example may show a limitation of the Wikidata request architecture since manually entering the question in our QA system returns a different output when compared to the test set.

```
print(qa("When was Lara Croft born?"))
```

```
Property probability:  [('date of birth', 0.39147305488586426), ('birthday', 0.35719698667526245), ('place of birth', 0.34
4503223896026 6), ('date of baptism in early childhood', 0.3037388026714325), ('birth name', 0.29374855756759644), ('first
appearance', 0.27443790435791016), ('earliest date', 0.25959154963493347), ('place of origin (Switzerland)', 0.25308698415
756226), ('inHerit Place Number', 0.2508789300918579), ('date of incorporation', 0.24481675028800964)]

Found entity:  Lara Croft

Answer found:
1968-02-14
```

Relevant section: 5.5

Figure 6: Output for "Elden Ring, when was the game published?". Example of the only question containing the word 'when' that our model answered correctly.

```
print(qa("Elden Ring, when was the game published?"))

Property probability:  [('publication date', 0.41499388217926025), ('year of taxon publication', 0.3396685719490051), ('pu
blished in', 0.33782488107681274), ('earliest date', 0.29632869362831116), ('author of afterword', 0.2794544696807861 3),
('publisher', 0.26471659541130066), ('place of publication', 0.2601791024208069), ('public domain date', 0.260175347328186
04), ('earliest end date', 0.25998827815055847), ('NSDAP membership number (1925-1945)', 0.256812930107116 7)]

Found entity:  Elden Ring

Answer found:
2022-02-25
```

Relevant section: 5.5

Question Type	Accuracy	Example of a mistake
Who is/was/are the X of Y?	93%	Who is the main enemy of Mario?
Alternative phrasings	71%	Who are four of the main characters of the 'The Witcher 3: Wild Hunt' game?
Yes/No?	66%	Can FIFA 20 be played on Nintendo Switch?
How Many?	50%	How many parts does the Ratchet & Clank have?
When?	33%	When was Electronic Arts founded?
Property is expressed as a verb	20%	Which game followed FIFA 19?

Table 2: Questions sorted by type and accuracy percentage.