Quiz Question Generation with Knowledge Graphs: A Pilot Study

Tomoe Mizutani
Yale University
tomoe.mizutani@yale.edu

Angus Fong
Yale University
hoching.fong@yale.edu

Abstract

In this work, we address the recent task of automatically generating quiz questions about real-world entities using Google Knowledge Graph and Wikidata. Our framework combines the unique strengths of the Google Graph Search API in matching relevant entities to a query field and Wikidata in graphically organizing information about real-world entities. Following an efficient, non-neural framework, it takes a user selected topic and mines large knowledge bases then generates a natural language question with multiple choices. The quality of questions is evaluated along multiple scales including readability, naturalness, difficulty etc. This line of work has implications for leveraging AI to improve education, and we encourage future work to further explore how best to use large-scale knowledge graphs to create study tools.

1 Introduction

Quiz questions are integral to learning. For example, medical students often consolidate their knowledge of anatomy and physiology by using ”quizbooks” that contain multiple choice questions and the answers, usually hidden. On the other hand, trivia games like Jeopardy are fun because they quiz players on interesting and often oddly specific questions about real-world entities.

With the right queries, answers to these quiz questions can be found in large-scale knowledge graphs such as Google Knowledge Graph[1] and Wikidata [2], which form the basis of modern search engines. In light of this, we took it upon ourselves to investigate whether we can build an automated learning tool that mines these public knowledge graphs to generate multiple-choice questions about user-selected topics, as well as correct and incorrect choices.

2 Related Work

2.1 Knowledge Graphs

A knowledge graph is a graph-based database that consists of triples representing relationship between entities in form (head entity, relation, tail entity). Google Knowledge Graph, officially introduced in 2012 upon integrating preexisting collaborative knowledge base called Freebase [3], consists of over 570 million entities and 18 billion facts. Google Knowledge Graph Search API gives user partial access to Google Knowledge Graph, allowing users to retrieve entities and their properties.

Wikidata is a collaborative knowledge base hosted by the Wikimedia Foundation. It serves as open data source for Wikipedia products such as Wikipedia and for public use. In WIkidata, each entity is an item with descriptions and properties.

The major difference between Google Knowledge Graph and Wikidata is that the former models relationships between entities, whereas the latter creates profiles for existing entities. Google Knowledge Graph focuses on building knowledge on how entities in the world interact with one another, where Wikidata serves to provide all relevant information on one entity.

2.2 Quiz Generation

Seyler and colleagues (2016) were the first to address this question. They used the Yago2 knowledge graph to come up with Jeopardy!-style ques-

[1] https://developers.google.com/knowledge-graph/
[2] https://www.wikidata.org/wiki/Wikidata:Main_Page
[3] https://developers.google.com/freebase/

tions regarding a specific user-defined topic. Simple one-hop pattern matching then consisted of a triple linking an entity, a property, and its value. Next, a simple, non-neural template matching framework was used to convert the logical form into a natural language question. They also trained a logistic classifier to quantify the difficulty of the questions generated compared to historical data from Jeopardy!.

Some of the more recent models include neural approaches in question generation (Indurthi et. al., 2017). In this model, they generated Question-Answer pairs from Freebase by first extracting a set of keywords from the graph and then using an RNN to generate a natural language question. Improvement in automatic question generation models can benefit downstream NLP tasks, such as the training of better QA systems.

More recently, Elsahar et. al., 2018 presented a neural model for question generation from knowledge base triples in a ZeroShot setup, leveraging triples occurrences in the natural language corpus in an encoder-decoder, and a novel part-of-speech copy action mechanism for natural language conversion. Benchmark and human evaluation on two dimensions show that their model sets a new state-ofthe-art for zero-shot question generation.

3 Methods

The workflow of our model uses a confluence of Google Knowledge Graph and the Wikidata Knowledge Base, thus combining their unique strengths and properties for an optimal user experience.

3.1 Topic / entity scoping

The user is first prompted to enter either a specific entity (eg. ”Taylor Swift”, or ”Dragomir R. Radev”) or a general topic (eg. ”American popstar” or ”Natural Language Processing”).

3.2 Entity selection with Google Graph Search API

A core functionality of the Google Graph Search API is to return a list of matching entities, as sorted by relevance. This forms the basis in Google’s recommendation systems and automatic question answering services (see Fig. 1). Here, we simply call Google’s Knowledge Graph Search API (https://developers.google.com/knowledge-graph/) on a entity with a user-defined type according to ”https://schema.org/”, a convention for organizing structured data in online knowledge graphs. This returns a large list of properties, ranked by relevance. The relevance scores in turn defined the probability distribution over which to repeatedly sample a real-world entity forming the subject of the question.

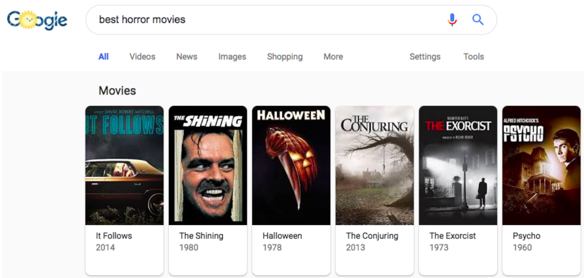

Figure 1: Google’s recommendation engine relies on its Knowledge Graph, which includes types and properties of various real-world entities.

3.3 Property selection with SPARQL WikiData Query Service

After selecting an entity using the Google Knowledge Graph, we used the SPARQL Wikidata Query Service (https://query.wikidata.org/) to retrieve its known properties. Fig. 2 shows an example enumeration of an entity and its properties. In its current state, the algorithm selects properties randomly. We note that future work can improve this by 1) learning an additional model to select nontrivial properties (eg. ”instance of”, ”given name” etc. are uninteresting) and 2) incorporating feedback, i.e. learning what the user should be quizzed on based on past performance.

3.4 Distractor Generation with WikiData

To generate n distractors for a multiple choice question, we first searched Wikidata to obtain the value labels for neighboring entities. (Eg. Taylor Lautner is in the returned list when querying ”Taylor Swift”). We note that this assumption, that alternative entities returned by the Google Graph Search API form competitors, is not always valid. If this search does not yield sufficient distractors, we next search Wikidata for all possible value labels under that property (eg. ”Pop music”, ”Classical music”, etc. under ”Genre”.)

3.5 Natural language question generation

Upon selecting query entity and property, we converted them into a fill-in-the-blank style natural

prop_label	prop_val
instance of	human
country of citizenship	United States of America
languages spoken, written or signed	English
instrument	Violín
instrument	guitar
work location	Nashville
occupation	actor
genre	pop music
instrument	ukulele
genre	country music
voice type	mezzo-soprano
field of work	musical composition
place of birth	Reading
instrument	banjo
discography	Taylor Swift discography
occupation	singer-songwriter
nominated for	Grammy Award for Album of the Year
award received	MTV Video Music Award for Best Female Video
record label	Big Machine Records

Figure 2: Given entity ”Taylor Swift”, we use SPARQL to query Wikidata for a list of associated properties.

language question. The property values used in Wikidata were inconsistent, in that some were predicate such as ”educated at”, while others were descriptors such as ”place of birth”. The key to generating grammatically correct sentences required the distinction between the two. One simple but powerful method to do this was to observe the last part of speech for each property. Predicate-form properties ended with a preposition. Using the Stanford NLP Part of Speech Tagger [1], we were able to detect predicate-form properties. Finally, we generated a natural language question which intent was easily comprehensible by the users.

4 Results

With our model, we were able to generate questions on a wide range of topics including people, animals, organizations, works of art/literature/music, etc (refer to Table 1 for an examples of generated questions).

4.1 Human Evaluation

Firstly, automatic evaluations such as BLEU-score and METEOR were deemed unsuitable for our task at hand. It is tedious to generate references

[1] `https://nlp.stanford.edu/software/tagger.shtml`

Topic	Generated Questions
computer scientist	Steve Gibson is educated at ___ [’1. Princeton University’, **’2. University of California, Berkeley’**, ’3. Massachusetts Institute of Technology’, ’4. Courant Institute of Mathematical Sciences’]
whales	The parent taxon of Whale shark is ___ [’1. Eschrichtius’, ’2. Orcinus’, **’3. Rhincodon’**, ’4. Balaenidae’]
London	The award received of Royal Society is ___ [’1. Spellemann Award for choir record of the year’, **’2. Princess of Asturias Award for Communications and Humanities’**, ’3. Coast Guard City’, ’4. Gramophone Award for Lifetime Achievement’]

Table 1: Example questions generated using our model.

for each question; plus, our fill-in-the-blank style questions are, strictly speaking, ungrammatical.

Instead, following Elsahar et. al., we conduct experiments to investigate the quality of our generated questions along two measures, and compare it with their published results:

(a) Trivialness: users were asked to indicate whether the generated question contains the given predicate in the fact or not, either directly or implicitly.

(b) Naturalness: the comprehensibility and readability of the generated questions. Each user was asked to rate each generated question using a scale from 1 to 5, where: (5) perfectly clear and natural, (3) artificial but understandable, and (1) completely not understandable.

7 users were invited to participate in a trial of this application with fully informed consent. To control variability, we asked each user to select a topic, or entity, with the ”Person” identifier. Results are shown in Table 2, and compared with Elsahar et. al.

Strikingly, our simpler model outperforms previous work in generating natural, readable questions, while being competitive in making ques-

Model	Trivialness (% "trivial")	Naturalness (mean)
Encoder-Decoder (Elsahar et. al.)	**6**	3.14
No Copy (Elsahar et. al.)	**6**	2.72
Types context (Elsahar et. al.)	37	3.21
All contexts (Elsahar et. al.)	46	2.61
Our model	23	**3.94**

Table 2: Performance metrics as compared to previous model.

tions nontrivial and sufficiently challenging.

5 Future Work

As it stands, our pilot study has several limitations. First, the Google Graph Search API does not return all relevant instances, since results are different from client-end Google querying. (Eg. "Yale Professors" only returns one person). This warrants further investigation. Secondly, properties are often uninteresting, too arcane, or too trivial. Learning a further model to make questions interesting and accessible would be rewarding. Finally, temporal and spatial reasoning can be incorporated to quiz users on relationships between entities without explicitly referring to a property. This greatly expands the range of questions in the search space.

6 References

Sathish Indurthi, Dinesh Raghu, Mitesh M. Khapra. Generating Natural Language Question-Answer Pairs from a Knowledge Graph Using a RNN Based Question Generation Model. `http://aclweb.org/anthology/E17-1036`

Dominic Seyler, Mohamed Yahya, Klaus Berberich. Knowledge Questions from Knowledge Graphs. `http://aclweb.org/anthology/E17-1036`

Hady Elsahar, Christophe Gravier, Frederique Laforest.Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types. `http://aclweb.org/anthology/N18-1020`