Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Drawing Questions from Wikidata

Bachelor Thesis

Fabian Bissig

fbissig@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Philipp Brandes, Laura Peer
Prof. Dr. Roger Wattenhofer

October 22, 2015

Acknowledgements

I am grateful for the participation in the evaluation and the provided feedback by the Distributed Computing Group and my family and friends.

Abstract

Quiz apps for mobile devices have recently been gaining in popularity. Questions for those are created manually. We introduce *Wikidata Quiz*, an application that accesses the structured data set of knowledge base Wikidata. We construct a graph by querying multiple Wikidata items originating from any chosen topic. The structure of the resulting graph is used to generate relevant questions with answer options. We evaluate the algorithm to learn if automatically generated questions are a viable replacement.

Contents

Chapter 1

Introduction

Quiz apps are popular among owners of mobile devices for improving general knowledge. What is more, they offer social media features such as playing against a friend of yours. The quiz questions are commonly created beforehand by humans, which guarantees a certain quality standard, but limits the number of questions and puts them at risk of becoming outdated.

Automated question generation is desirable for the following reasons:

- wider area of topics that can be covered
- finding outdated questions does not require examining them all
- effortless internationalisation through use of global knowledge covering all parts of the world and localised labels
- access to complementary descriptions and images of objects appearing in questions whose display improves learning effect

While Wikipedia's great source of knowledge is a huge benefit for humankind, its unstructured nature renders it very difficult for parsing by machines. This issue is addressed by Wikidata [1], a collaboratively edited knowledge base, launched in 2012. Knowledge bases contain facts about the world, such as gender and date of birth of individuals. Wikidata's initial purpose was to serve other Wikimedia projects [2], first and foremost by centralising interlanguage links connecting Wikipedias of different languages. The data is available under a free licence and can be downloaded. Wikidata's support by the Wikimedia Foundation led Google to shut down Freebase [3], the main structured knowledge base before Wikidata [4] that launched in 2007.

Knowledge bases may play a much more vital role in the near future because of the emergence of intelligent personal assistants such as Apple's Siri, Google's Google Now and Microsoft's Cortana, which could feed off of structured information such as Wikidata.

In our project we develop an application that can be utilised to generate quiz questions with answer options. We introduce an algorithm to find relevant questions to a topic the user chooses. We subsequently analyse the perceived quality of those questions judged by users.

1.1 Related Work

QuizUp [5] is a mobile trivia game that achieved remarkable success. Its questions are tailored to young people and its highlights are the ability to play against friends and the time pressure when answering questions.

Automatic question generation works well for lexical databases. Brown et al. [6] automatically generate questions to assess users' vocabulary. They make use of WordNet, a lexical database that groups English words into sets of synonyms. This allows their system to provide users with texts to read targeted to their individual reading levels.

Knowledge bases play a vital part in finding answers to questions posed by humans in a natural language. This is what the computer science discipline Question Answering (QA) is concerned about. Yao & Van Durme [7] have outlined how to extract answers from Freebase. They parse a question such as "what is the name of Albert Einstein's sister" and search for the answer in Freebase.

A considerable challenge is to generate questions about the contents of an English story. Kunichika et al. [8] realise adaptive Question and Answer, so that learners of different understanding states are given suitable questions. Heilman & Smith [9] go a similar route but they overgenerate questions, then rate them.

1.2 Outline

We start by explaining important terms and concepts in Chapter 2. In Chapter 3 we offer a detailed look on how we use the info on Wikidata to generate questions and answers. We provide an overview of the technologies used in Chapter 4. In Chapter 5 we explain how we evaluate the application and present the type of questions that work best. Chapter 6 summarises the findings and suggests future work.

Chapter 2

Background

2.1 Semantic Web

The World Wide Web revolutionised the way how we access information. It removed the restrictions caused by costly physical storage and the lengthy transfer of physical mail. Quick distribution of information was facilitated. However, the Web is concerned with the structure of data – not its meaning. The Semantic Web, on the other hand, connects objects, i.e. it is used to represent interactions and relationships between objects. It can be seen as the next evolutionary step of the World Wide Web.

Relationships in the Semantic Web are stored in form of a triplestore composed of subject–predicate–object. When a website claims “Lisa Simpson was born in Springfield.”, it would have the following HTML markup:

```
<div vocab="http://schema.org/" typeof="Person"
    href="https://www.wikidata.org/wiki/Q5846">
  <span property="name">Lisa Simpson</span> was born in
  <span property="birthPlace" typeof="Place"
      href="https://www.wikidata.org/wiki/Q151076">
    <span property="name">Springfield</span>.
  </span>
</div>
```

This can be written in triples (each terminated by a `.`):

```
<https://www.wikidata.org/wiki/Q5846>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://schema.org/Person> .
<https://www.wikidata.org/wiki/Q5846> <http://schema.org/name>
    "Lisa Simpson" .
<https://www.wikidata.org/wiki/Q5846> <http://schema.org/birthPlace>
    <https://www.wikidata.org/wiki/Q151076> .
<https://www.wikidata.org/wiki/Q151076> <http://schema.org/itemtype>
    <http://schema.org/Place> .
```

```
<https://www.wikidata.org/wiki/Q151076> <http://schema.org/name>
    "Springfield" .
```

The HTML markup illustrates how the information could be embedded on a website. It can be scraped by a machine and transformed into triples. Each triple then represents an edge in the graph.

Wikidata's role in the Semantic Web is to be one source of information like all others, available for getting referenced.

2.2 Wikidata

Wikidata stores a multitude of *items* with an *identifier*. An item consists of *statements*, which are comprised of a *claim* and an optional *reference* supporting that claim. A *claim* consists of a *property*, a *value* and optional *qualifiers*. *Values* can be either other items or a variable of some complex type (e.g. integers and dates of designated precision, strings, URLs, geographical coordinates).

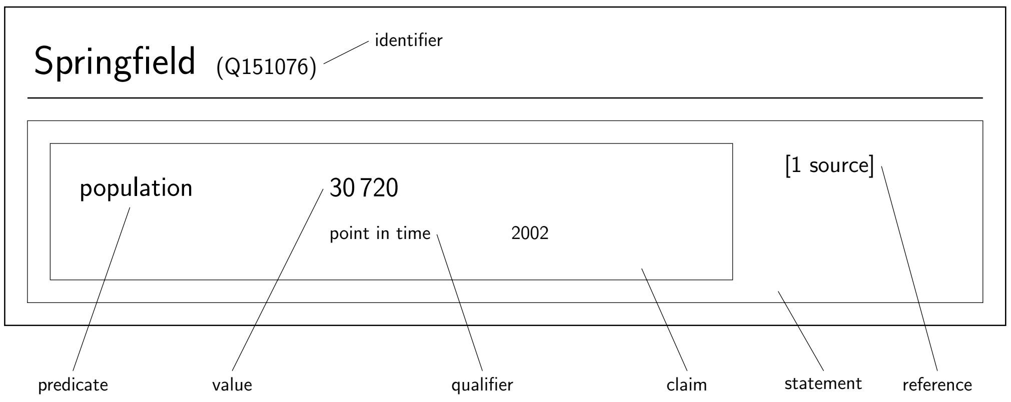

Figure 2.1: Exemplary Wikidata item

For example, we consider the Wikidata item for the fictional town Springfield, identified by Q151076 (see figure 2.1). It contains a statement that consists of a claim and a reference (collapsed to [1 source] in the figure) to an episode where the claim was made. The claim consists of a property population (identified by P1082) and a value 30 720. This claim also has a qualifier point in time with value 2002.

Chapter 3

Generation of Questions and Answers

There are lots of items in Wikidata, connected by claims. Posing a question with the use of Wikidata is straight forward: As demonstrated in question 3.1, displaying an item and the property of a claim as well as multiple possible values (with at least one value correct) suffices.

Q:	The Simpsons: *production company*
1.	20th Century Fox
2.	Dan Castellaneta
3.	Maggie Simpson
4.	Sitcom

Question 3.1: Trivial answer

However, there are two main issues that have to be addressed here:

1. How to find relevant questions that are somehow connected to a topic the user requests questions about and are relatively rich in variety
2. How to find possible answers for the user to choose from, with them being somehow related to the type of question, but only one answer possibility being correct

To start with, the user selects a *topic*, i.e. a Wikidata item. We place this topic as the *origin* vertex of our directed graph.

This chapter examines how we generate a *graph* consisting of *vertices* and *edges* starting from a single Wikidata item and how to find relevant questions and possible answers to the questions. For that, we make use of the claims that connect Wikidata items.

3.1 Graph Generation

As we are constructing a graph based on a structured but sometimes partially incomplete data set, we need to make an assumption. We assume that well connected vertices in a graph built around a primary topic express particular

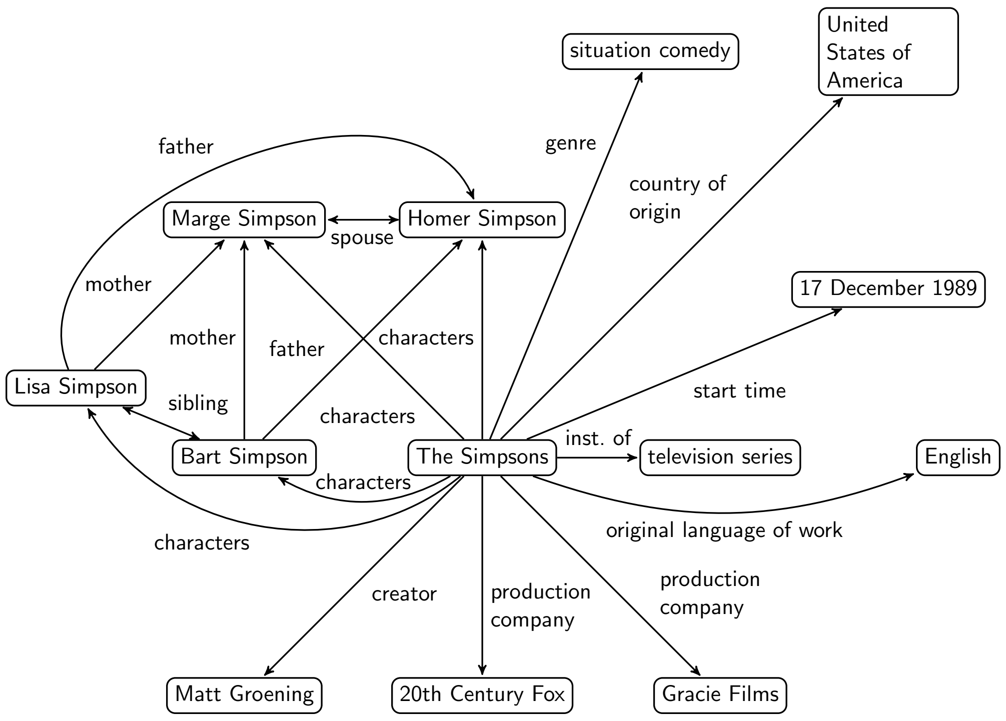

Figure 3.1: Illustrative graph for The Simpsons

relevance to the topic. This can be justified by the fact that all vertices are connected with the origin with maximum distance 2.

The graph is constructed using linked data stored in Wikidata. It has the form of subject–predicate–object, corresponding to a single claim. This results in a graph as shown in figure 3.1. There are two functions available to the graph generation algorithm: one for receiving *outgoing claims*, a set of predicate–object pairs for a subject (e.g. *mother*: Marge Simpson for Lisa Simpson) and one for receiving *incoming claims*, a set of subject–predicate pairs for an object (e.g. The Simpsons: *country of origin* for United States of America).

We perform a breadth-first search on the linked data stored in Wikidata. The procedure is driven by a queue that contains the Wikidata items to be processed. When processing an item, a new vertex and an edge that connects the two is added to the graph for each Wikidata item that is connected by a Wikidata property to the original Wikidata item. Additionally, each new Wikidata item is added to the queue for future processing. For instance, the Wikidata item The Simpsons contains a claim with property creator and value Matt Groening. This means that the graph gets a node for the creator Matt Groening and an edge labelled creator connected to The Simpsons.

When added, each vertex in the graph is assigned a number (the *level*), which denotes the distance to the origin. The level increases by one for each hop from the origin. The procedure stops at a certain level so as not to run indefinitely. It is important to choose the optimal maximum distance. Its choice is influenced by two factors:

1. keep waiting time low
2. stay on topic

Testing showed that the maximum distance to the origin necessary for good questions is 2. Rising that level leads to questions too irrelevant to the chosen topic, while still taking a long time to generate.

3.1.1 Outgoing Claims

Outgoing claims are predominantly useful because they are relevant to the Wikidata item. They state some fact that is directly concerned with the item. This allows them to be considered for the most part without severe performance impact.

Wikidata's data model allows claims to have qualifiers, which are additional property–value pairs. Qualifiers further describe or refine the claim in a statement. Qualifiers are not modelled in the graph. However, each encountered Wikidata item in a qualifier is added to the queue. For example, The Simpsons contains a claim with the property voice actor and the value Dan Castellaneta. Additional qualifiers with property role list characters Homer Simpson, Krusty the Clown and Mayor Quimby. The graph is not modified for those qualifiers. However, the characters are added to the queue, as they form important items that would get lost if ignored or not referenced in another way.

3.1.2 Incoming Claims

There can be an enormous amount of incoming claims. Large countries get referenced for every piece of art (each book, movie, TV series, etc.)

Incoming claims must only be considered cautiously because they introduce mostly distant and potentially irrelevant items. In addition, their consideration causes a massive performance impact.

Additional measures are taken to reduce the number of Wikidata items that get introduced through incoming claims. Only up to a specific number of claims with the same property (e.g. actor) are kept. In a movie with dozens of actors, only a subset thereof is kept. This means that we have to determine which ones to keep and which ones to discard by assigning a number called *importance* to each Wikidata item. Finding important items is done by determining their degree of connectivity within Wikidata. This procedure also relies on the assumption that items with more claims are more relevant.

Testing revealed that considering incoming claims could only be done for the origin. Finding incoming claims and especially sorting them by importance takes up to 10 seconds for each Wikidata item. Finding them for an entire level except for the origin would drastically delay graph generation. Either way, if the algorithm deems that there are enough outgoing claims for the origin already, incoming claims will not be considered at all.

3.1.3 Pruning the Graph

The finished graph with vertices with maximum distance 2 to the origin is often large and contains many leaves. Such a graph is suitable for finding answer options. However, this graph has a bias towards vertices with many leaves, complicating question generation. Additionally, the graph is unnecessarily large for finding relevant questions, which always involve well connected vertices. That is why a duplicate of the graph is pruned for finding relevant questions.

To prune the graph, we use a simple heuristic approach. We iterate over all vertices, calculating their respective degree, removing all vertices with a degree lower than a specified threshold.

3.2 Extraction of Questions

The graph consists of a multitude of vertices all representing a single Wikidata item (such as a human, an administrative territorial entity, an architectural structure, a movie, etc.). Each edge in the graph is a Wikidata property and represents a potential question (e.g. mother connecting Bart Simpson to Marge Simpson). We will use the terms *subject* and *property* for the question and *answers*, divided into *solution* and *answer options*, for the correct and wrong answers provided. Note that the subject as well as the answers are Wikidata items, while the property is a Wikidata property.

However, simply posing questions based on random edges in the graph would likely lead to irrelevant ones getting asked for the most part. Taking into consideration vertices' degrees allows us to find edges evolving around well connected vertices. On the assumption that well connected vertices in the graph are particularly relevant to the topic, this allows us to find the most interesting questions for a user to answer.

We face two challenges concerning the edges, our prospective questions, but also the vertices, our prospective answer options:

1. How to weigh them such that more relevant ones are prioritised
2. How to vary them such that we get great variety with little repetition

3.2.1 Weighting and Varying of Edges (Questions)

Based on the assumption, we iterate over all edges and assign them a *weight* that is calculated by adding together the outdegrees of the two vertices connected by the edge. We sort the edges by weight in descending order, preferring edges with high weights for prospective questions. A vital observation was to not take into account the indegree. As an example, the indegree is very high for countries as those are referenced by many vertices. This would then lead to a bias pushing questions with that country as the answer to the top.

Without variation, clusters build up. For instance, questions about a country involve the property contains administrative territorial entity a lot because those administrative territorial entites are well connected among themselves in the graph. To counteract that, the weights of edges with the same property are increased depending on whether they are first, second, third and so on to appear.

Determining the order of questions that get asked based on the graph's connections leads to monotonous questions appearing in clusters. For example, most Wikidata items representing humans contain a property country of citizenship. Those then often point to the same country (e.g. to the United States of America for actors of a movie from Hollywood). Because of the apparent significance of such a country, demonstrated by many incoming claims, such questions would accumulate. This is tackled by limiting the number of questions with the same property or solution. Additionally, the position of questions in the result set is adjusted such that similar ones do not cluster together.

3.3 Extraction of Answer Options

The simplest solution to obtaining answer options is grabbing random vertices from the graph and verifying they are not a correct answer. For example, question 3.2 Marge Simpson: *sister* with solution Patty Bouvier may offer sitcom, which is a vertex in the graph but makes no sense in this context. As we only want one correct answer, it must not offer Selma Bouvier as an answer option, because they both are sisters of Marge Simpson, indicated by an edge sister connecting Marge Simpson to Selma Bouvier.

Q:	Marge Simpson: *sister*
1.	Patty Bouvier
2.	sitcom
3.	~~Selma Bouvier~~
4.	Jacqueline Bouvier

Question 3.2: Answer options of different validity and quality

It is evident that answer options should at least be of the same instance (e.g. human) as the solution to the question to prevent the user from being able to rule out all answer options because they simply do not apply to the question. For instance, proposing names of movies or locations when actually asking for a relative of a cartoon character would make it too easy for the user. Again, we continue with the assumption that vertices with high outdegrees are more relevant. We justify this by the fact that all vertices in the graph are connected to the origin with maximum distance 2.

3.3.1 Getting Items of Same Instance

Wikidata by itself does not have an object model that requires Wikidata editors to classify items. Ontologies with classes, subclasses and instances are ideal for organising knowledge bases. However, there is a single property instance of that established itself as the way to achieve exactly that. We follow instance of to another Wikidata item (e.g. film) to get relevant answer options.

We consider the vertices connected by an instance of edge as *instances* and *classes*. They are only Wikidata items too, though. As shown in figure 3.2, fictional human will have an incoming edge for each Simpsons character. Likewise, human has an incoming edge for each voice actor. By following the edge instance of to the class, we can choose any of the other incoming edges with the same label instance of to find a suitable answer option.

We notice that Wikidata items can be instances of multiple classes. For example, a member of the Simpsons family might be a fictional human, fictional character, animated character and television character. A country might be an instance of a country, sovereign state, federation, etc. When trying to find answer options, we use this to our advantage by taking the union of instances of all those classes.

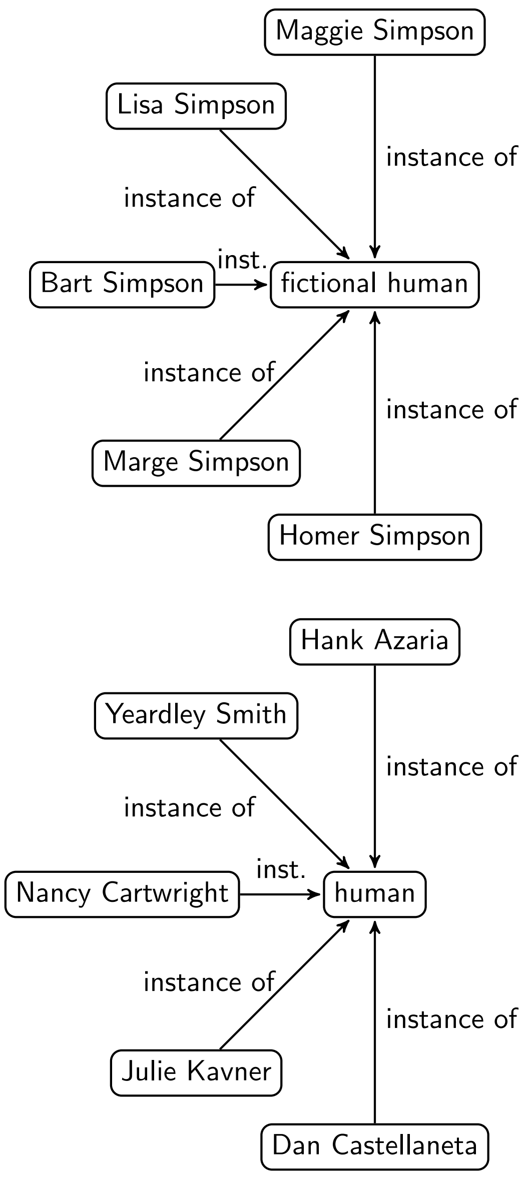

Figure 3.2: Classes of various instances

3.3.2 Weighting and Varying of Vertices (Answers)

Based on the assumption, we sort all instances of one or multiple classes by outdegree in descending order. Similar to the weights of the edges, we observed that the indegrees should not be considered. Vertices with high indegrees tend to be the same for different topics (e.g. the two genders or large countries).

As our questions offer four answers with one of them being the solution, we need up to three answer options. If we always take the three vertices with highest outdegree, answer options become highly repetitive. We use a queue from which we can dequeue as desired to get varying answer options.

We investigated further approaches to eliminate irrelevant answer options. For example, we wanted questions that have a human as the solution to only offer humans with the same gender. To achieve that, we follow the edge sex or gender to either male or female. In the queue containing the answer options we check the gender of each item and skip it if it does not match.

Chapter 4

Implementation

The application is written in Python using the Flask web framework. A graph database stores the dataset of Wikidata for efficient access. A Python software package is used for the creation and handling of the graph.

4.1 Wikibase RDF Query

Wikibase RDF Query is a software package that combines the graph database Blazegraph with tools to import a dump of Wikidata. The dataset of Wikidata is expressed in the Resource Description Framework (RDF) data model. It consists of a long list of triples in the form of subject–predicate–object. This dataset is imported into Blazegraph, which optimises it for faster access.

The necessary data is requested by SPARQL queries. Because of the fact that multiple extensive queries have to be answered in quick succession, the graph database Blazegraph is running locally on the same server as the website. Blazegraph accesses a file stored on an SSD with a 7 GB compressed Wikidata dump inflated to over 50 GB with indexes.

4.2 Python Web Application (Flask)

Flask is a simple web framework for Python. We chose Python because of its simplicity when writing code that handles data. We chose to write a web application with Flask because it allows universal access from all sorts of mobile devices.

Communication with the graph database Blazegraph is achieved by using the Python package SPARQLWrapper. It forwards SPARQL queries to the graph database and returns the requested data in the form of nested Python dictionaries, which can easily be worked with.

4.3 NetworkX

NetworkX offers efficient implementations for querying and modifying a graph. It is used for creating the directed graph that contains the prospective questions and answers.

For debugging and curiosity reasons, it must be possible to inspect the graph visually by rendering it with a suitable layouting algorithm. The pruned version of the graph is used for that purpose, as it is more compact. Graphviz is a package of tools for drawing graphs. It has multiple layouting algorithms, of which dot is used. It draws directed graphs as hierarchies, which leads to neatly arranged graphs after moderate rendering times.

4.4 Redis

Redis is a data structure server. It maps keys to types of values. Redis was chosen to supply a way of caching both generated questions and labels.

Keys are given an expiration time such that questions and labels are regenerated periodically. As Wikidata gets updated regularly, this ensures that stale data does not remain in the cache for too long.

Cached questions are stored in terms of a Redis list, which provides both constant time push and pop operations. A user successively answering questions only triggers pop commands on Redis until the user changes topic.

Redis is also used to cache labels in various languages. If a required label is not cached, it is requested from the local Wikidata dump. For languages not included in the dump, labels can be requested from the official Wikidata server, increasing waiting time, though.

CHAPTER 5

Results

5.1 Public Evaluation of Questions

To evaluate the generated questions and answers, we invited participants to choose a topic and let our backend generate questions. The generated questions are cached, and on request one question after the other is shown to the participant. Participants are prompted to rate single questions on a scale from 1 to 5 (worst to best). At any time, participants can choose to change the topic or to reset questions for the current topic.

Of all the participants, 18 rated at least 10 questions. Altogether, 1262 questions were rated in the course of five releases. The three versions 0.1, 0.4 and 0.5 have received enough ratings to be of sufficient statistical significance.

Q:	1973 World Rally Championship for Manufacturers: *followed by*
1.	1976 World Rally Championship for Manufacturers
2.	1975 World Rally Championship for Manufacturers
3.	1974 World Rally Championship for Manufacturers
4.	1973 World Rally Championship for Manufacturers

Question 5.1: Trivial question referring to next such event

Public evaluation has uncovered a few smaller bugs in the beginning. Badly rated questions helped us detect ones that are trivial to solve, do not make sense or are illogical. Trivial questions include ones about events asking for the next/previous such event, with the answer options only differing in the year (see question 5.1). When asking about relatives of a person, participants requested the answer options to be of the same gender to remain logical (see question 5.2).

Q:	Alexis Arquette: *sister*
1.	Rosanna Arquette
2.	Paul Calderón
3.	Julia Sweeney
4.	Frank Whaley

Question 5.2: Illogical gender

Participants ranked poorly questions that are related to Wikipedia's structure with categories and templates (see question 5.3). While those are used for

Q:	Sri Lanka: *category for films shot at this location*
1.	Category:Republics
2.	Category:North Central Province, Sri Lanka
3.	Category:Films shot in Sri Lanka
4.	Category:Island countries

Question 5.3: Related to Wikipedia's categorisation function

Wikipedia internal purposes (to link to the corresponding Wikipedia article), they do not make sense to the participant and should thus be eliminated. Likewise, questions involving Wikidata's class structure (with edges like instance of and subclass of) were also eliminated.

The evaluation also uncovered questions with certain properties that were undesirable in the opinion of the participants. This includes properties sister city, said to be the same as, forename and surname (see question 5.4).

Q:	Dave: *said to be the same as*
1.	Davy
2.	Dave
3.	David
4.	Dafydd

Question 5.4: Undesirable question

Participants praised the ability of Wikidata Quiz to generate a whole range of questions about movies and TV series. They were also impressed by how it maps entire family trees, originating from a single person, to a whole array of questions.

Q:	Clark County: *located in the administrative territorial entity*
1.	California
2.	Idaho
3.	Nevada
4.	Arizona

Question 5.5: Only question for Las Vegas

The evaluation showed that the quality of questions is primarily dependent on the completeness of Wikidata's dataset. It revealed that Wikidata is still highly incomplete in certain areas. Most cities we looked into are well covered with a notable exception being Las Vegas (see question 5.5). Coverage is good for all topics surrounding pop culture while still sometimes being incomplete. As an example, TV series exist but Wikidata items for single episodes are mostly still lacking. But in time we expect the quality of questions to rise with improvements in Wikidata's contents.

In general, the application sparked the participants' interest and curiosity in further digging into the large dataset of Wikidata. We found that the work of contributors to Wikidata can be interactively evaluated by Wikidata Quiz.

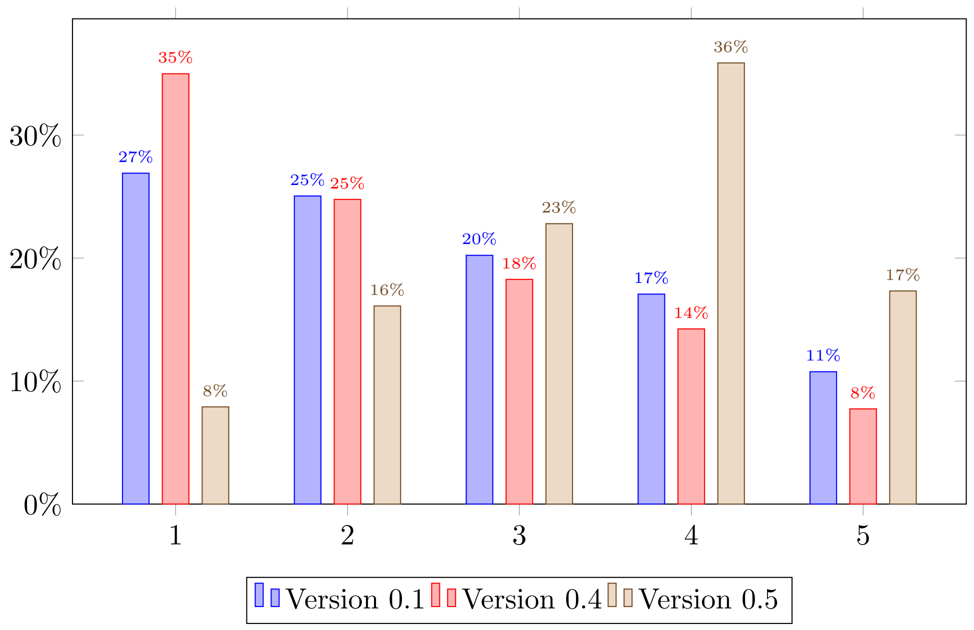

Figure 5.1: Ratings of 1191 questions from 1 to 5 (worst to best) for three versions

Figure 5.1 shows how 1191 generated questions were rated for the three statistically significant versions 0.1, 0.4 and 0.5. Initially, participants noticed many trivial questions and reported them, as can be seen from the low ratings. Following versions were under even higher scrutiny, with participants reporting further trivial and also illogical questions.

For the last version, participants reported mostly good questions, as is evident from the higher ratings. Questions with low ratings are ones on topics with few Wikidata claims or are caused by wrong information in Wikidata. Additionally, participants also praised the mixed order that was established by shuffling the generated questions in the last version.

5.2 Runtime

Runtime is a great challenge for an application accessing a large knowledge base. The graph is constructed gradually by executing consecutive queries on the graph database. Each query takes several tens of milliseconds. The query for getting incoming claims, i.e. items that reference the current item, even takes several seconds. Depending on subject (e.g. large countries that are referenced a lot) it takes even considerably longer. All these milliseconds accumulate to multiple seconds for large graphs.

To give an idea on how long participants approximately waited, we determined the average waiting time. For that, we regenerated all questions for the 86 topics participants chose and that were logged. This resulted in an average waiting time of 3.3 seconds. Note that labels were cached in the Redis instance.

We provide an overview of topics and their respective generation time below:

Switzerland	17.3 s
Volkswagen	2.3 s
mathematics	6.3 s
Sun	9.0 s
The Simpsons	4.5 s
Albert Einstein	7.4 s
Roger Federer	4.3 s
Linus Torvalds	3.3 s
Super Nintendo Entertainment System	0.4 s

Table 5.6: Generation times for popular Wikidata items in seconds

Chapter 6

Conclusion and Future Work

We introduced an algorithm that allows for automated generation of questions with answers. The algorithm uses a graph that is constructed using structured data from knowledge base Wikidata. We set up a website where users can give feedback on generated questions. Analysis shows that participants found good questions and they saw improvements in the algorithm over time. We found that the incomplete status of Wikidata negatively impacts the quality of the generated graph. We found limits in the types of questions that are suitable for generating from knowledge bases. Based on the positive feedback especially on questions regarding pop culture, we conclude that automated question generation provides benefits for quiz applications in that questions no longer become outdated.

Wikidata is expected to considerably grow over the following years. It has the potential to reduce time spent on the repetitive task of keeping information up to date in general. We see Wikidata Quiz as only one application to depend on structured data. In the future, with knowledge bases becoming more reliable, we predict more reliance on such datasets in other domains as well.

We believe that future work should invest in finding new question types. Good questions can involve asking for values like integers and dates, or requesting users to compare images.

The dataset is so large that even research in graph databases that are optimised for question generation might be of interest. This could include preprocessing the data to find well connected items. Different approaches to finding the major vertices and edges in a subgraph may also be investigated.

Bibliography

[1] Wikimedia Foundation: Wikidata. https://www.wikidata.org/ Accessed: 2015-10-17.

[2] Vrandečić, D.: Wikidata: A new platform for collaborative data collection. In: Proceedings of the 21st international conference companion on World Wide Web, ACM (2012) 1063–1064

[3] Metaweb Technologies: Freebase. https://www.freebase.com/ Accessed: 2015-10-17.

[4] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM (2008) 1247–1250

[5] Plain Vanilla Games: QuizUp. https://www.quizup.com/ Accessed: 2015-10-17.

[6] Brown, J.C., Frishkoff, G.A., Eskenazi, M.: Automatic question generation for vocabulary assessment. In: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, Association for Computational Linguistics (2005) 819–826

[7] Yao, X., Van Durme, B.: Information extraction over structured data: Question answering with Freebase. In: Proceedings of ACL. (2014)

[8] Kunichika, H., Katayama, T., Hirashima, T., Takeuchi, A.: Automated question generation methods for intelligent English learning systems and its evaluation. In: Proc. of ICCE. (2004)

[9] Heilman, M., Smith, N.A.: Good question! statistical ranking for question generation. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics (2010) 609–617