Bachelor Thesis

Context Tracking for Question Answering with Aqqu

Anushe Glushik

Examiner: Prof.Dr.Hannah Bast
Adviser: Niklas Schnelle

Albert-Ludwigs-University Freiburg

Faculty of Engineering

Department of Computer Science

Chair for Algorithms and Data Structures

August 23rd, 2019

Writing Period

23. 05. 2019 – 23. 08. 2019

Examiner

Prof.Dr.Hannah Bast

Adviser

Niklas Schnelle

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.

I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date

Signature

Abstract

The goal of this work is to implement a context tracking functionality in the Aqqu question answering system and to study the influence of context tracking on the overall performance of Aqqu. A question answering system with context tracking functionality takes the context into account. To find the correct answer, the system not only processes the current question, but also the history of questions asked before.

To implement context tracking for Aqqu, we developped a web user interface (UI) in the form of a chatbot. Additionally we added to it a data augmentation functionality. We call the Aqqu system with a context tracking functionality conversational Aqqu. We evaluated the conversational version of Aqqu against a newly created conversational version of the original WebQSP questions dataset. For this we transformed and reordered the original one-off questions into conversations about topic entities shared by the original questions.

From the evaluation results it was apparent that the conversational question answering is slightly less accurate, when compared to a single-question answering. Yet taking into account, that it is a far more complicated task, the conversational system still showed a very good performance. Moreover context tracking brings more convenience into the question answering process.

Zusammenfassung

Das Ziel dieser Arbeit ist eine Kontextverfolgungsfunktion im Aqqu- Fragebeantwortungssystem zu implementieren und den Einfluss der Kontextverfolgung auf die Gesamtleistung von Aqqu zu untersuchen. Ein Fragebeantwortungssystem mit Kontextverfolgungsfunktion berücksichtigt den Kontext. Um die richtige Antwort zu finden, verarbeitet das System nicht nur die eine gestellte Frage, sondern auch den Verlauf der zuvor gestellten Fragen.

Um das Kontext-Tracking für Aqqu zu implementieren, haben wir eine Web- Benutzeroberfläche in Form eines Chatbots entwickelt. Es wurde eine zusätzliche Funktionalität zur Datenerweiterung im Chatbot eingeführt. Wir nennen das Aqqu-System mit einer Kontextverfolgungsfunktion Conversational Aqqu. Wir haben Conversational Aqqu mit einer neu erstellten Konversationsversion des ursprünglichen WebQSP-Fragen-Datasets verglichen. Zu diesem Zweck haben wir die ursprünglichen Einzelfragen in Konversationen umgewandelt und nach Themeneinheiten neu angeordnet.

Aus den Bewertungsergebnissen ging hervor, dass die Beantwortung von Konversationsfragen im Vergleich zur Beantwortung von Einzelfragen weniger genau ist. Unter Berücksichtigung der Tatsache, dass es sich um eine weitaus kompliziertere Aufgabe handelt, zeigte das Conversational Aqqu dennoch eine sehr gute Leistung. Darüber hinaus erleichtert das Kontext-Tracking die Beantwortung von Fragen.

Contents

Abstract iii

Zusammenfassung v

1 Introduction 1

2 Related Work 5

3 Aqqu 7

3.1 How Aqqu Question Answering Works 7

3.2 Aqqu Usage 10

4 Context Tracking 13

4.1 Main Approach 14

4.2 Gender Identification 15

4.3 Usage 17

5 Aqqu Chatbot 19

5.1 Question Answering with Aqqu Chatbot 19

5.2 Extending the Dataset with Aqqu Chatbot 23

5.3 Usage 28

6 Datasets for Training and Evaluation 29

6.1 Original Dataset 29

6.2 Conversational Dataset 32

6.3 Basic Conversational Dataset 38
6.4 Extended Conversational Dataset 39

7 Evaluation **41**
7.1 Evaluation Metrics 41
7.2 Averaged Evaluation Metrics 43
7.3 Example of an Evaluation on a Small Dataset 44

8 Experiments **47**
8.1 Experiment 1 47
8.2 Experiment 2 48
8.3 Experiment 3 48
8.4 Analysis of Experiments 1-3 49
8.5 Experiment 4 49
8.6 Experiment 5 50

9 Conclusions **51**

10 Future Work **53**

11 Useful Links **55**

Bibliography **58**

List of Figures

1 An example of question answering in Aqqu Chatbot. 20
2 An example of question answering with additional context-based questions. 20
3 An example of question answering with two main identified entities. . 21
4 No answers found. 22
5 Wrong answer found. 22
6 An example of a separate tracking of male and female genders. . . . 23
7 Each answer candidate can be marked as correct by clicking the gray button next to it. 24
8 Gray buttons turn green after clicking. 25
9 An additional candidate can be chosen to replace the previous answer. 26
10 The answer can be overwritten in the dataset. 27

List of Tables

1 Evaluation on a tiny dataset. 45
2 Average results for tiny dataset. 45
3 Results of experiment 1. 48
4 Results of experiment 2. 48
5 Results of experiment 3. 49
6 Results of experiment 4. 50
7 Results of experiment 5. 50

1 Introduction

Question answering (QA) is a complex task that lies within the fields of information retrieval and natural language processing. Its objective is to build systems that automatically answer questions posed in a natural language. QA systems usually query knowledge bases, where the data is organized into relatively simple structures. However natural language structures have a high variability and ambiguity, and translating them into a structured knowledge base query is a challenging task. Furthermore context is very important in natural language, as the same sentence can have different meanings in different dialogues.

In this project, we focused on analyzing the influence of context on the question answering process, using the system Aqqu [1]. By context we mean the immediate history of previous questions and answers.

The original Aqqu processes questions independently from each other and gives isolated answers. The system does not 'remember' the sequence of questions and does not take notice of the order in which the questions are asked.

However, context is very important in real life. One question is often followed by additional or clarifying questions. A QA system that can track the context and follow a conversation should be more intuitive and convenient to use and may give more reasonable answers.

As an example, let us look at the following dialogue of a user with Aqqu, without taking context into account:

User: Who is Bach?

AqquBot: Johann Sebastian Bach, profession: Composer, Musician, Harpsichordist, Teacher, Violinist, Organist, Violist, Cantor

User: Who was he married to?

AqquBot: World Health Organization, founders: Brock Chisholm

Obviously the reply that the user gets is not what is expected. The system does not handle the ellipsis phenomena in the conversation. To answer all the questions correctly the system requires the information from the conversation history. The ellipsis of the entity *he* refers to *Johann Sebastian Bach*, but the system does not recognize that. It considers the entity *Who* to be an abbreviation of *World Health Organization* and gives the best possible answer for this entity.

In the next example we see a more human-like conversation, produced by the conversational Aqqu system, that has been developed within this project:

User: Who is Bach?

AqquBot: Johann Sebastian Bach, profession: Composer, Musician, Harpsichordist, Teacher, Violinist, Organist, Violist, Cantor

User: Who was he married to?

AqquBot: Johann Sebastian Bach, spouse: Anna Magdalena Bach, Maria Barbara Bach

User: Where was she born?

AqquBot: Anna Magdalena Bach, place of birth: Zeitz

User: Where is it?

AqquBot: Zeitz, containedby: Germany, Saxony-Anhalt

The system understands who is meant by he, she and it, when the user substitutes the noun with a pronoun. This is a more natural way to hold a dialogue.

To make the question answering process more human-like, a simple context tracking functionality was integrated in Aqqu. The explanations of the main approach of context tracking can be found in section 3 – Context Tracking. A short overview of the Conversational Aqqu application is presented in section 4 – Aqqu Chatbot. An important part of the project was to analyze how will the performance of the Aqqu system change after integrating context tracking. For this purpose the WebQSP dataset was restructured into different conversational datasets (section 5 – Evaluation Dataset). In these datasets the questions were sorted by entities and reorganized into conversations. Then the system was trained and evaluated using these datasets (Section 6 and 7 – Evaluation and Experiments). The influence of context tracking is analyzed using the evaluation results in section 8 – Conclusion.

2 Related Work

In this section the current work within the field of Conversational Question Answering is analyzed and compared to the context tracking approach that was implemented in this thesis.

There are several QA datasets, that have a conversational structure. The CoQA dataset [2] contains 127k questions in 8k conversations, each question has an answer and a rationale that supports the answer. The QuAC dataset [3] contains 100k questions in 14k crowdsourced QA dialogs. The answers in the dataset are provided in the form of a text span and the data is collected from Wikipedia texts. The QBLink dataset [4] has about 18k question sequences and each sequence consists of three questions. In these datasets the learning models need to take into account the whole conversation and not just separate questions and answers. Existing approaches to accomplish the conversational QA use a question and a conversation history as an input to produce a correct answer. In these approaches questions are answered on unstructured text data, while Aqqu searches the answers in a structured knowledge base.

In the method that is presented in [5] the system learns from conversational datasets. The model has an additional multi-layer recurrent neural network to capture contextual information from a conversation. For this approach the CoQA dataset was used. The method achieves very good results, but it is only applied to in-domain training and evaluation, and it is not clear how it performs on open-domain with knowledge base.

The Dialog-to-Action approach is described in paper [6]. The authors map conversational natural language questions to a logical form representation and implement a dialog memory management to track the interaction history. The dialog memory management component maintains three types of information: entities, predicates and action sub-sequences. The actions are used to generate logical forms. Each action consists of a semantic category, a function symbol, and a list of arguments. The logical form of the question is created using the conversation history. The questions are answered using a large-scale knowledge base. In the training each question is paired with the correct answer and not paired with the correct logical form. The experiments are conducted with the CSQA dataset [10]. The dataset is based on Wikidata [11] and includes 152K dialogs for training and 16K/28K dialogs for evaluation and testing. The method that is introduced in this thesis is simpler and tracks the history of identified entities without taking into account the predicates and action sub-sequences.

In this work the questions from the WebQSP dataset were restructured into conversations, where each conversation shares one entity. The WebQSP_conversational dataset contains 5k questions in 2281 dialogs. Context tracking for Aqqu is implemented by tracking the history of previous questions and storing previously identified entities from questions and answers. The system does not change the learning model, it extends the set of processing entities and makes the system choose from the entities from both the current and the previous questions. The conversational Aqqu was trained and evaluated with conversational data.

3 Aqqu

3.1 How Aqqu Question Answering Works

Aqqu is a question answering system. It was developed at the Chair for Algorithms and Data Structures from the Department of Computer Science, University of Freiburg by Prof. Dr. Hannah Bast, Elmar Haussmann and Niklas Schnelle [1]. Aqqu is available online and via Telegramm. The system is built on Freebase, but it can be adjusted to any knowledge base. To better understand the approach that has been applied in this project, let us look at how the Aqqu system chooses an answers to a questions from Freebase. Freebase is a data base, containing a very large amount of information. The information can be queried with the SPARQL language. A SPARQL query consists of entities and relations. Aqqu extracts the possible entities and relations from a natural-language query and creates possible matching SPARQL queries. For example for the question "who is the ceo of apple" the SPARQL query is [1]:

select ?name **where** {
Managing_Director job_title.people_with_this_title ?0 .
?0 employment_tenure.company AppleInc .
?0 employment_tenure.person ?name
}

With the SPARQL queries the answer candidates are extracted from the knowledge base. Then the system prunes and sorts the candidates and gives a list of sorted answers as the output. For each question the system goes through the following steps:

- **Entity Matching**. The system splits the question into subsequences and then looks for matching entities or matching entity aliases (from CrossWikis [7]) in the knowledge base. Then a score for each match is computed using the likelihood of the entity to be an alias of the subsequence. Also a match-independent popularity score is computed for each entity. The popularity score is the number of times the entity is mentioned in the ClueWeb12 dataset [8], according to the annotations provided by Google [9].

- **Candidate Generation**. For each entity the system searches for matching relations. Three types of templates are used. A template can consist of entity placeholders e, relation placeholders r, an answer node t and an intermediate object m:

  - a simple template with a single entity and a single relation:
    $<e_1><r_1><\mathrm{t}>$
    An example of the first template for "Who invented the smartphone?":
    <smartphone(entity)><inventor(relation)><Apple Inc.(answer)>

  - a template with one entity, two relations and a mediator object:
    $<e_1><r_1><\mathrm{m}><r_2><\mathrm{t}>$
    An example of the second template for "What company did Steve Jobs work for?"
    <Steve Jobs(entity)><employment(relation)><mediator object>
    <mediator object><company(relation)><Apple Inc.(answer)>

- a template with two entities, two relations and one mediator object:

  $<e_1><r_1><m>$

  $<m><r_2><e_2>$

  $<m><r_3><t>$

  An example of the third template for "What device did Steve Jobs invent for Apple Inc.?"

  <Steve Jobs(entity)><employment(relation)><mediator object>

  <mediator object><company(relation)><Apple Inc.(answer)>

  <mediator object><invention(relation)><smartphone(answer)>

This is obtained via a single SPARQL query for each entity for each relation. A matched template is a query candidate.

- **Relation Matching**. For each candidate from the set of query candidates, the system computes how well the words from the relations match the words from the question. The system considers four types of matches: literal match, derivation match, synonym match and context match. A score for each match is calculated according to the template type.

- **Features Extraction**. The features are extracted for the future candidate pruning and ranking. There are 23 different features that Aqqu uses for learning and evaluation: features, that are based on the entity matching results and relation matching results, features that quantify the result size, the answer-type feature and the n-gram relation matching feature. Also in the current version a Deep Learning based mechanism for matching relations to the question is used. It is based on a convolutional neural network, which is trained on the relation template candidates to judge the similarity between the candidates and the question. The scores from the entity matching and the relation matching are used to calculate these features, as well as parameters such as the number of words in the question, the result size and the n-gram relation.

- **Candidate Pruning**. The candidate pruning can be done either without n-grams (with hard pruning) or with n-grams (with a pruning classifier). We used the second variant with n-grams. A pruning classifier is a logistic regression classifier. The classifier is trained to discard 'obviously' bad candidates. The pruning classifier works with the same 23 features as the ranking classifier. The system returns "no answer" only when the set of candidates is empty.

- **Candidate Ranking**. Two types of ranking were investigated in the Aqqu paper to rank the remaining candidates: pointwise ranking and pairwise ranking. For a pointwise ranking a score is calculated for each candidate and the candidates are sorted according to this score. In the pairwise ranking the ranking problem is transformed to a binary classifier, where each candidate is compared with all the other candidates. The system can be trained with either a linear or a random forest classifier. For this work an Aqqu system with a pairwise ranking and a random forest classifier variant was used. After the ranking all the candidates are sorted according to their relevance.

3.2 Aqqu Usage

Aqqu can be used on a website with a convenient interface:

`http://aqqu.informatik.uni-freiburg.de`

On the website it is possible to either choose a question from a dataset or to type in an arbitrary question. The request with the question is sent to a server after clicking the *Translate and Execute* button. There can be several answers to the same question, and Aqqu returns several candidates. These candidates are ranked from the most probable to the least probable candidate.

Another way to run Aqqu is by making a request in a browser in the following form:

`http://aqqu.informatik.uni-freiburg.de/api?q=who%20played%20dory%20in%20finding%20nemo`

http://aqqu.informatik.uni-freiburg.de/api	specifies the backend that is used
%20	URL encoding for space
?q=who played dory in finding nemo	the executed query

The answer is displayed as JSON, that has the following structure:


```
{"candidates" :
[{"answers": [{"mid" : "m.019xz9", "name":      "Ulm"}, ...],
  "entity_matchess": [{"mid" : "m.0jcx"}, ... ],
  "features" : {"avg_em_popularity" : 14.04, "
     avg_em_surface_score" : 1, "cardinality" : 13, ... },
  "matches_answer_type" : 1.9,
  "pattern" : "ERT",
  "rank_score" : null,
  "relation_matches" : [{"name" : "people.person.place_of_birth
     ", "token_positions" : [4]}, ... ]],
  "root_node" : {"mid" : "m.0jcx"},
  "out_relations" : [{"name" :  "people.person.place_of_birth",
      "target_node" : {"mid" : null, "out_relations" : []}},
     ... ],
  "sparql" : "PREFIX fb: <http://rdf.freebase.com/ns/>\n\
     nSELECT DISTINCT ?0 WHERE {\n fb:m.0jcx fb:people.person.
     place_of_birth ?0 .\n FILTER (?0 != fb:m.0jcx) \n} LIMIT
     300"}, ...],
 "parsed_query" :
{"content_token_positions" : [2, ...],
```

```
  "identified_entities" : [{"entity" : {"mid" : "m.0jcx", "name
     " : "Albert Einstein"}, "perfect_match" : true, "score" :
     1249375, "surface_score" : 0.998050830195, "text_match" :
     false, "token_positions" : [2,3], "types" : ["people.
     person", ...]}, ... ],
  "is_count" : false,
  "target_type" : "('location', 0.7), ('event', 0.6), ('
     conference', 0.6)",
  "tokens" : [{"lemma" : "where", "offset" : 0, "orth" : "where
     ", "tag" : "WRB"}, ... ]},
 "raw_query" :  "where was albert einstein born"
}
```


The information on how to train, build and run the Aqqu system backend can be found under the following link:

https://github.com/ad-freiburg/aqqu

Aqqu uses docker, hence it is possible to create multiple different containers and to train the system with different parameters and datasets in each of these containers.

4 Context Tracking

In natural language, context can often clarify the meaning of a question and simplify the search for an answer. We want to test whether by providing a conversational context we will improve the flexibility, convenience and performance of Aqqu. The Aqqu question answering process includes the following steps:

- Entity Matching
- Candidate Generation
- Relation Matching
- Features Extraction
- Candidate Pruning
- Candidate Ranking

The strategy of the system is to match as many entities as possible, exclude the least relevant answers and score the matches. The main approach to implementing conversation following in this project is to store relevantly matched entities from the previous questions and add these to the set of identified entities of the current question, if the system finds a pronoun in the processed question. Therefore, Aqqu will also take the objects mentioned before into account. The additional entities can be seen as the necessary context for the system to figure out an answer.

Of course the context is not always relevant. Sometimes a user can ask questions about several unrelated subjects. This is why in this approach the system only adds the previous entities to the entity set if there is a pronoun in the current question.

4.1 Main Approach

The main approach for the conversation tracking consists of the following steps:

1. Store the identified entities (ID and name) after the system gets a result.
2. Look for pronouns in further queries.
3. If the processed query contains a pronoun - add the previous entities to the list of matched entities; if it does not – treat the query as usual.

All these steps are done on the client side.

4. On the Aqqu backend side: process the list of identified entities (extended by the context entites or not) and identify the gender for each entity (for gender version).

An example of a resulting url for a question that contains a pronoun could be:

`http://titan.informatik.privat:8090/?q=where%20was%20he%20born&p=m.0jcx,AlbertEinstein`

p= indicates an additional entity, where *m.0jcx* is the entity ID and *Albert Einstein* is the entity name. Also, more than one additional entity can be concatenated to the end of the url:

`http://titan.informatik.privat:8090/?q=where%20was%20he%20born&p=m.0jcx,AlbertEinstein&p=m.05d1y,NikolaTesla`

In this case, both Albert Einstein and Nicola Tesla are added to the identified entities. The system stores both these entities in the query and in the results. This allows the system to also continue a conversation, if the following questions refer to some entity from the previous answers.

For example:

User: Where was Albert Einstein born?
Chatbot answer 1: Albert Einstein, place of birth: Ulm
User: Where is it?
Chatbot answer 1: Albert Einstein, location: Germany, Princeton, Munich, Bern
Chatbot answer 2: Ulm, containedby: Germany, Baden-Württemberg

We built a gender identifier in the system to take the gender of an entity into account.

4.2 Gender Identification

For each entity the system identifies a gender and stores it into a dictionary. The gender identification is done by the Aqqu backend side. The dictionary is stored in the JSON, that is sent as an answer to the chatbot. To identify a gender of an entity the system uses one of two methods:

1. Find a genus in the gender.csv file. The gender.csv file contains around 3.5 million entities. Each entity is marked as Male, Female or has no mark. The system looks an entity up in the file and if the entity is found – its gender will be stored in a gender dictionary.

2. If the entity is not found in the gender.csv file the system tries to guess the gender. To guess the gender we used a gender-guesser python module.

Of course the gender can be wrongly identified. For example the entity *Bahamas* is a country and should be identified as *Neutral*. However the name Bahamas is stored as *Male* in gender.csv. This may lead to the following confusing dialogue:

User: What language is spoken in the Bahamas?
Chatbot: Bahamas, languages spoken: English Language, Bahamas Creole English Language
User: Where is it?
Chatbot: Bahamas Creole English Language, main country: Bahamas

The system does not know that it should be related to the Bahamas.

Despite such mistakes the gender identification with the gender.csv file is more reliable. The gender-guesser does not work for gender-neutral names and can not identify a gender correctly for such entities.

Another example of a wrong gender identification is:

User: Where did Coco Chanel live?
Chatbot: Coco Chanel, location: Paris
User: What country is it a capital of?
Chatbot: Maine-et-Loire, capital: Angers

The system does not find an entity *Paris* in the gender.csv file and uses the gender-guesser. The gender-guesser takes the city name *Paris* for a female name and identifies it as a *Female* entity.

The system stores plural and singular entities together. Therefore the pronouns it and they are treated equally.

4.3 Usage

The conversational tracking functionality is included in the Aqqu backend and can be found under the link:

`https://ad-git.informatik.uni-freiburg.de/ad296/Aqqu`

The system can either process questions without any context, i. e. any additional entities, or process questions with entity-based context. This makes it a backwards compatible addition - as long as the feature is not used, Aqqu works as before.

5 Aqqu Chatbot

5.1 Question Answering with Aqqu Chatbot

A conversational web UI in the form of a chatbot was developed within this project. The Aqqu chatbot was built using Flask. The application takes the text from the Question field and sends a request to Aqqu's backend. Aqqu gives an answer in the form of an API, which then is transformed into a readable answer in the application. The application stores the entities in the cache, in order to process potential future questions with pronouns. The cache is always overwritten when the next question does not have pronouns. The chatbot always gives the answer with the highest rank first. To get another answer, the user can click on the Next answer button. When there are no possible candidates left, the application will report, that it has no alternative answers.

The format of the answer in the Aqqu Chatterbot is: entity name,candidate relation match: all answers (all answers are presented sequentially and comma-separated).

In the answer from Figure 1, Albert Einstein is the entity name, place of birth is the candidate relation match and Ulm is the answer. The additional answers are obtained by clicking the "Next Answer" button.

In Figure 2, the context tracking is shown. The system recognizes that with *he*, the user means Albert Einstein and gives correct answers.

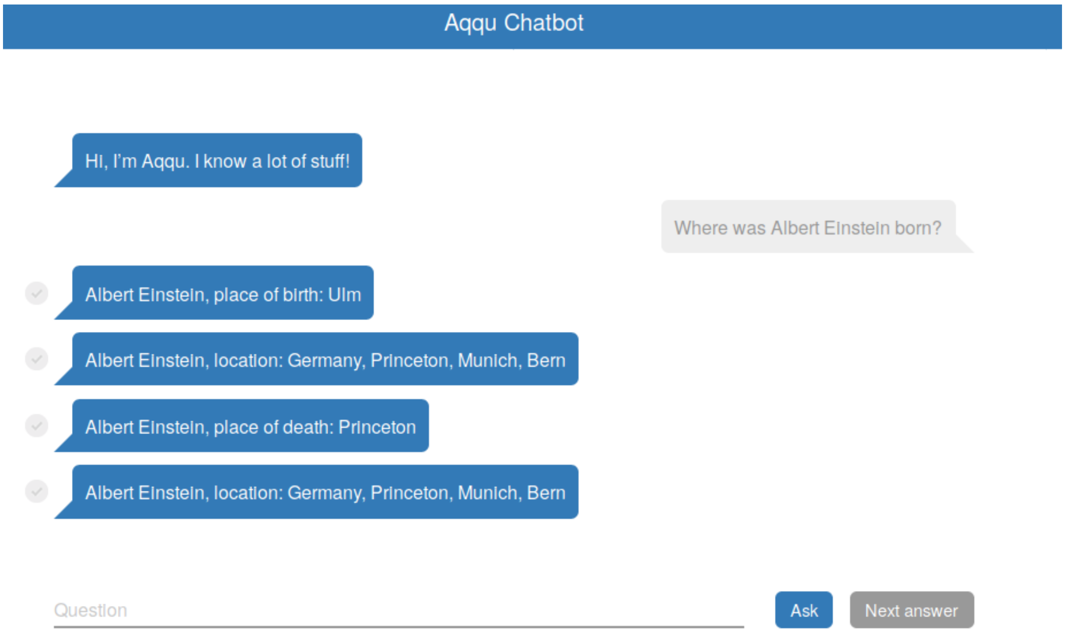

Figure 1: An example of question answering in Aqqu Chatbot.

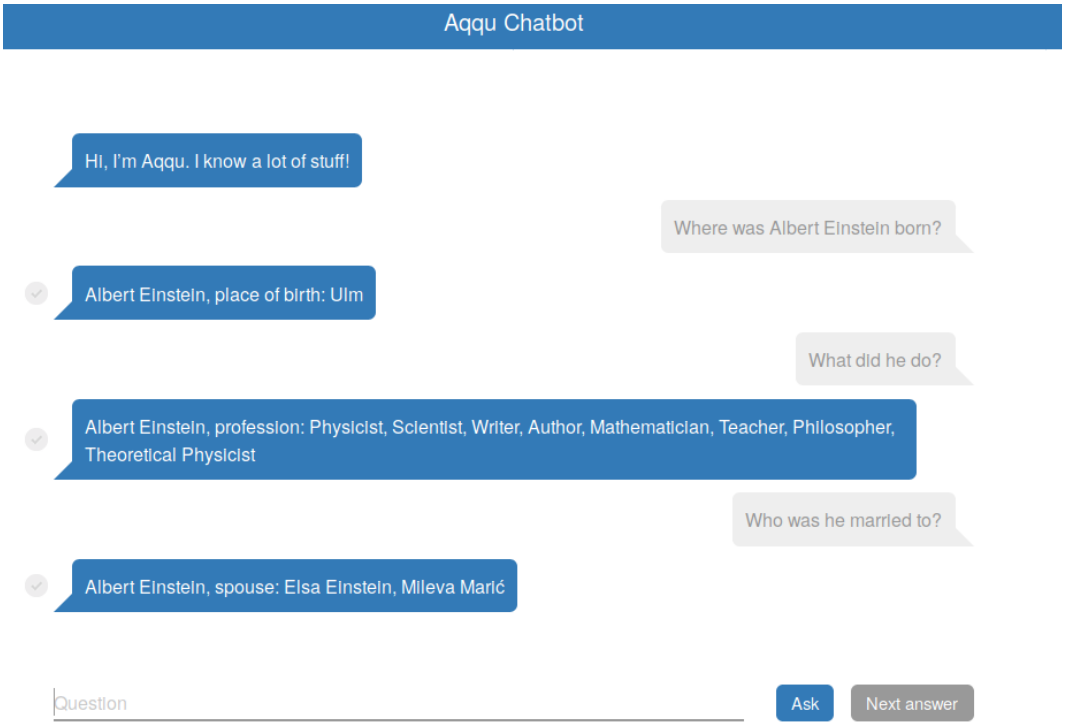

Figure 2: An example of question answering with additional context-based questions.

This behavior can cause some problems for questions where an entity as well as a pronoun are given (Figure 3). For example: 'Who was Isaac Newton and what did he do?' In this case the application will use both Isaac Newton and the previously stored entity. In the following picture it is shown that the system has both identified Issac Newton and Albert Einstein. The answer referring to Isaac Newton got a higher rank – thus it is the first candidate.

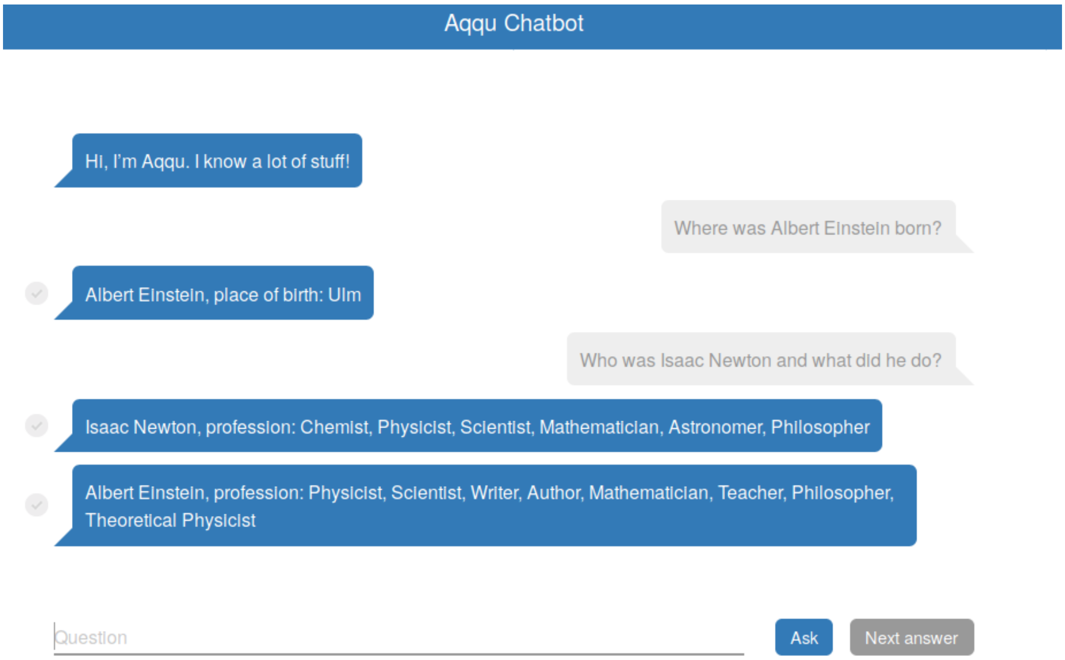

Figure 3: An example of question answering with two main identified entities.

In the Figure 4 the application has not found any suitable candidates.

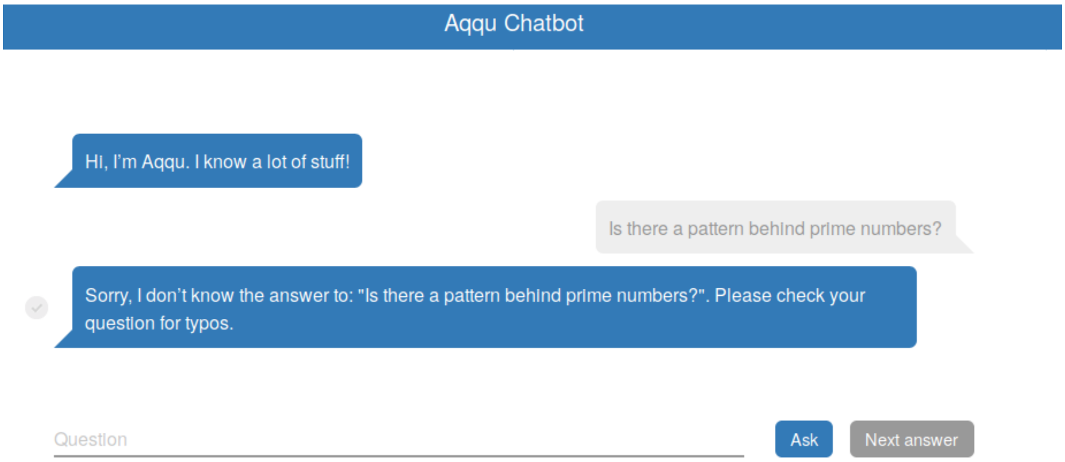

Figure 4: No answers found.

Sometimes it gives wrong answers as well (Figure 5).

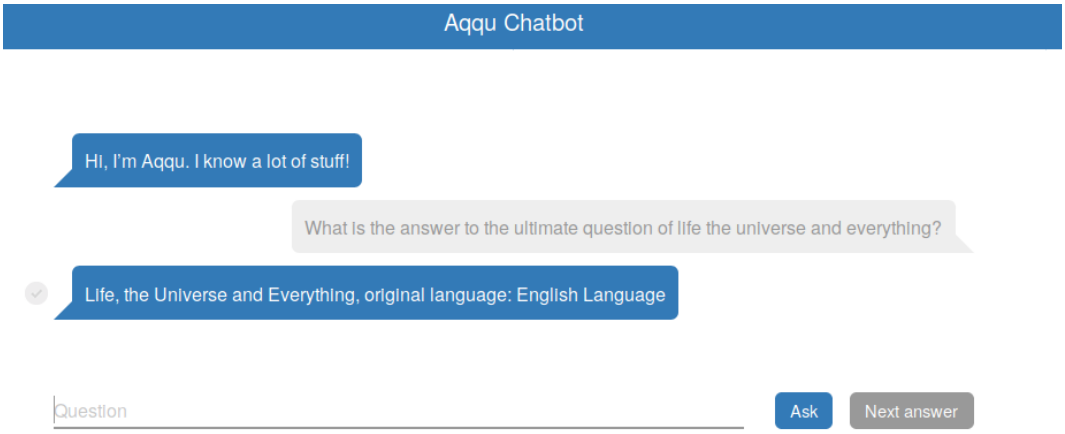

Figure 5: Wrong answer found.

The system tracks female, male and neutral entities separately. This allows a user to ask additional questions to a question that has been asked a few rounds before, if the questions in between have a different gender. This is also more similar to the human way to keep a dialogue.

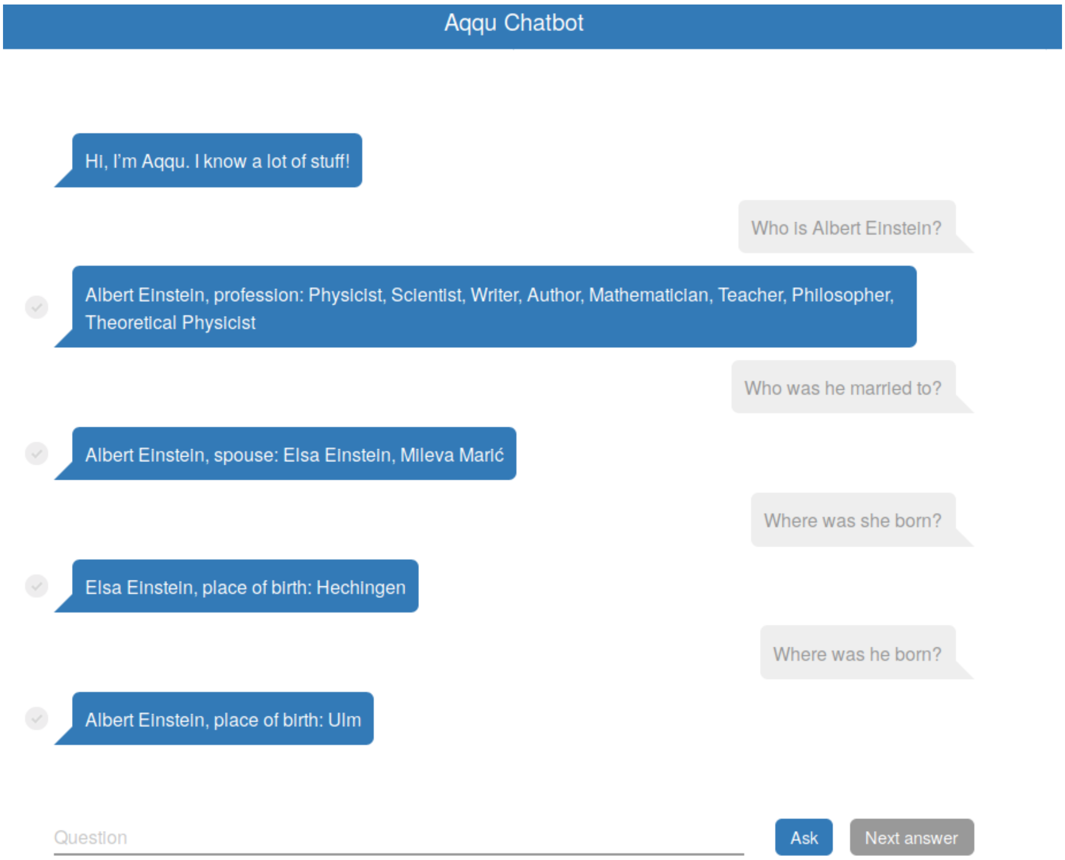

Figure 6: An example of a separate tracking of male and female genders.

5.2 Extending the Dataset with Aqqu Chatbot

Collecting data for question answering systems is a difficult task. It has to be done by humans and requires a lot of work: creating questions, identifying the entities full names and id, finding the right answers to the question, writing a SPARQL query of the question and storing the question and the answer in a special format, that can be later processed by a computer. All that makes the data collection an expensive and tiresome process.

With the Aqqu Chatbot question-answer storing functionality the collecting of data can be done with much less effort.

In the chatbot a user can mark an answer as correct by clicking a gray check sign button right in the answer area. There is such a button on the left side of each answer candidate.

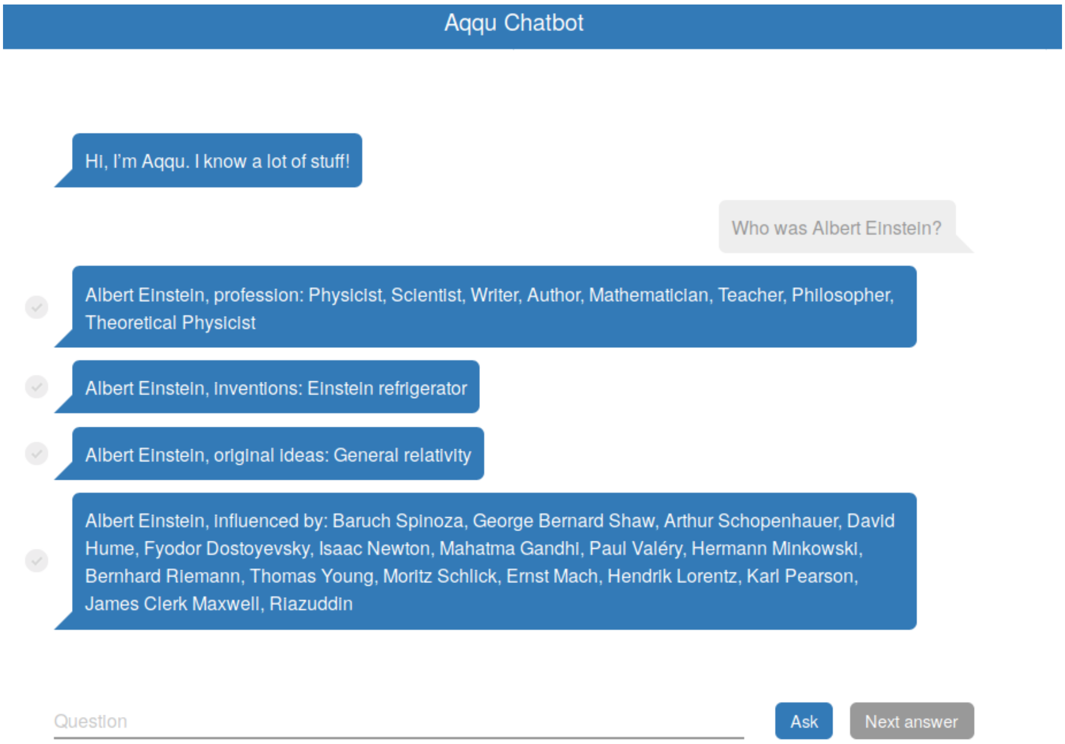

Figure 7: Each answer candidate can be marked as correct by clicking the gray button next to it.

When the user marks a correct answer – the gray button changes to green and all the other gray buttons disappear. Therefore no other candidate can be marked as correct. When the Next Answer button is clicked however, the next candidate can be marked as correct and the previous question-answer will be overwritten in the WebQSP format dataset by the new selection.

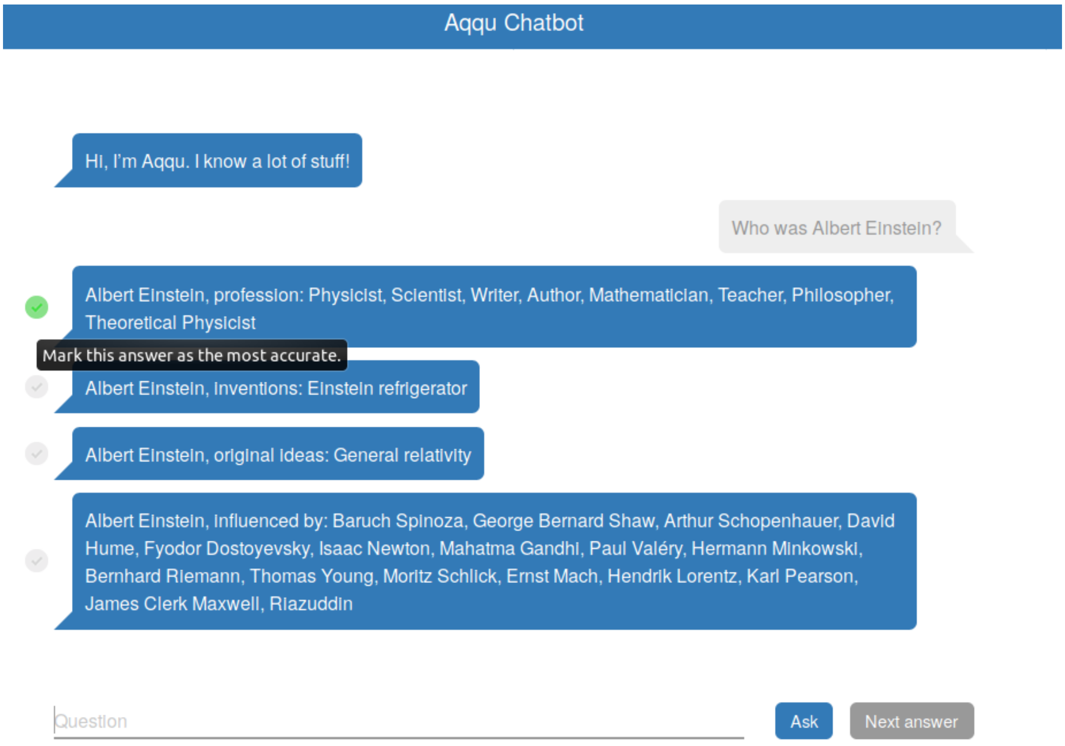

Figure 8: Gray buttons turn green after clicking.

If the user asks the same question again and marks a new correct answer, the old question-answer pair will be overwritten in the dataset (Figures 9-10).

The Aqqu Chatbot dataset is saved on a server, where the Aqqu backend container is stored. For each user there is a separate dataset in the WebQSP format. To identify a user a unique user agent string is used. An example of such a string could be:

`Mozilla/5.0(X11;Ubuntu;Linuxx86_64;rv:68.0)Gecko/20000000Firefox/68.0`

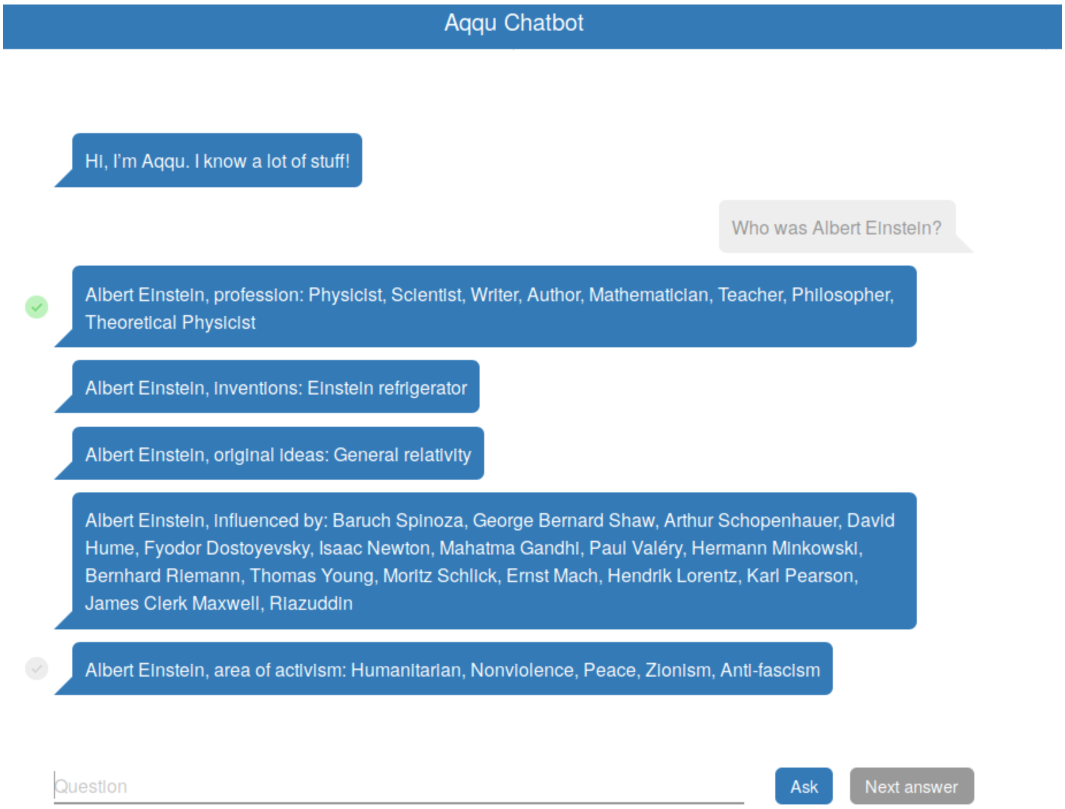

Figure 9: An additional candidate can be chosen to replace the previous answer.

We use different datasets for each user in order to protect the stored data. A user can only overwrite his/her own answers. Different users' answers can not interfere with each other. Some users can give more correct answers than others. It is possible to evaluate the data provided by users of the Aqqu Chatbot and exclude the users with a high error ratio.

With the Aqqu Chatbot dataset extension functionality it is rather easy to augment the existing WebQSP dataset. Training the model on a bigger dataset can improve the question answering of Aqqu in the future.

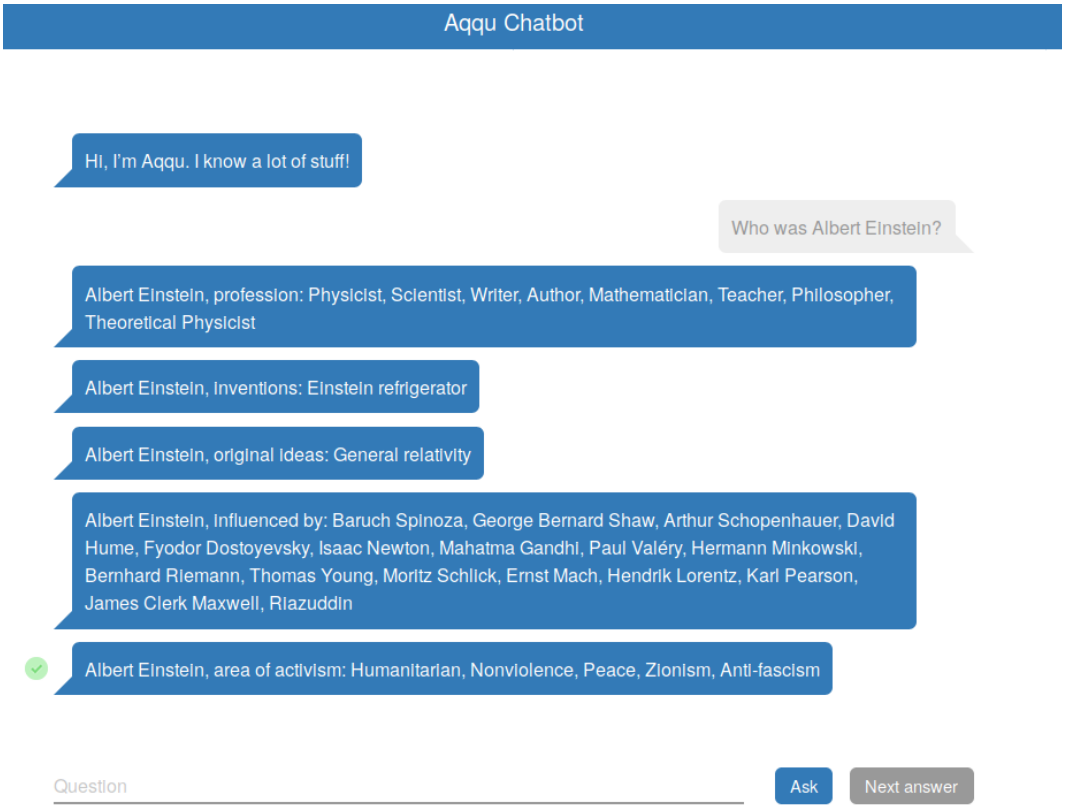

Figure 10: The answer can be overwritten in the dataset.

5.3 Usage

The Aqqu Chatbot code can be found under:

`https://ad-git.informatik.uni-freiburg.de/ad296/AqquChatterbot/tree/Gender`

6 Datasets for Training and Evaluation

6.1 Original Dataset

For the evaluation of the system performance we used the WebQSP dataset. The WebQSP dataset was split into training (70%) and testing (30%) datasets. We chose the same splitting ratio, as the one used for the evaluation of the non-conversational Aqqu [1]. The original dataset consists of a list of questions. Here is an example of a question from the dataset:


```
{
  "Version": "1.0",
  "FreebaseVersion": "2015-08-09",
  "Questions": [
    {
      "QuestionId": "WebQTest-0",
      "RawQuestion": "what does jamaican people speak?",
      "ProcessedQuestion": "what does jamaican people speak",
      "Parses": [
        {
          "ParseId": "WebQTest-0.P0",
          "AnnotatorId": 0,
          "AnnotatorComment": {
            "ParseQuality": "Complete",
            "QuestionQuality": "Good",
            "Confidence": "Normal",
```

```
      "FreeFormComment": "First-round parse verification"
    },
    "Sparql": "PREFIX ns: <http://rdf.freebase.com/ns/>\
       nSELECT DISTINCT ?x\nWHERE {\nFILTER (?x != ns:m
       .03_r3)\nFILTER (!isLiteral(?x) OR lang(?x) = ''
       OR langMatches(lang(?x), 'en'))\nns:m.03_r3 ns:
       location.country.languages_spoken ?x .\n}\n",
    "PotentialTopicEntityMention": "jamaican",
    "TopicEntityName": "Jamaica",
    "TopicEntityMid": "m.03_r3",
    "InferentialChain": [
      "location.country.languages_spoken"
    ],
    "Constraints": [],
    "Time": null,
    "Order": null,
    "Answers": [
      {
        "AnswerType": "Entity",
        "AnswerArgument": "m.01428y",
        "EntityName": "Jamaican English"
      },
      {
        "AnswerType": "Entity",
        "AnswerArgument": "m.04ygk0",
        "EntityName": "Jamaican Creole English Language"
      }
    ]
  },
  {
    "ParseId": "WebQTest-0.P1",
    "AnnotatorId": 0,
    "AnnotatorComment": {
```

```
          "ParseQuality": "Complete",
          "QuestionQuality": "Good",
          "Confidence": "Normal",
          "FreeFormComment": "First-round parse verification"
        },
        "Sparql": "PREFIX ns: <http://rdf.freebase.com/ns/>\
           nSELECT DISTINCT ?x\nWHERE {\nFILTER (?x != ns:m
           .03_r3)\nFILTER (!isLiteral(?x) OR lang(?x) = ''
           OR langMatches(lang(?x), 'en'))\nns:m.03_r3 ns:
           location.country.official_language ?x .\n}\n",
        "PotentialTopicEntityMention": "jamaican",
        "TopicEntityName": "Jamaica",
        "TopicEntityMid": "m.03_r3",
        "InferentialChain": [
          "location.country.official_language"
        ],
        "Constraints": [],
        "Time": null,
        "Order": null,
        "Answers": [
          {
            "AnswerType": "Entity",
            "AnswerArgument": "m.01428y",
            "EntityName": "Jamaican English"
          }
        ]
      }
    ]
  }, ... ]}
```

The dataset consists of a list of questions. A few important parameters of a question in the dataset are:

- **RawQuestion** - not processed query text that may for example include the question mark.

- **ProcessedQuestion** - processed query text, lowercase and without a question mark.

- **Sparql** - the executed SPARQL query.

- **TopicEntityName** - the name of the entity.

- **TopicEntityMid** - the knowledge base ID of the entity.

- **Answers** - the answers for the question. A question can have many correct answers. For example, for a question *What language did ancient Romans write in?* there are two answers: *Greek Language* and *Latin Language*.

- **AnswerType** - the answer type can be either *Entity* or *Value*.

- **EntityName** - if the answer type is not *Entity* the *EntityName*is null.

- **AnswerArgument** - entity ID for answer type *Entity* and value for the type *Value*.

6.2 Conversational Dataset

To evaluate the performance of the system, the questions from the dataset were reshaped into conversations. The script for converting an original WebQSP dataset to a conversational dataset can be found under: `https://ad-git.informatik.uni-freiburg.de/ad/Aqqu/create_data_set.py`

The script gathers the questions into groups according to their entities. In these groups, the first question does not change. In all of the following questions, the entities were replaced with a corresponding pronoun. The entity is defined in the dataset under TopicEntityName. For each entity, the script determines its gender and replaces the entity name with either he, she or it. To find out, which gender the entity belongs to, the script first looks for the entity name in the gender.csv file. If the name is not found, then the gender is guessed using the gender_guesser package. This is an example of a conversation from the conversational WebQSP:

Leading query: *what did galileo do to become famous?*

Second query: *what he was famous for?*

Third query: *what discovery did he make?*

The structure of the conversational dataset is:


```
{
    "FreebaseVersion": "2015-08-09",
    "Conversations": [
        {
            "Questions": [
                {
                    "Parses": [
                        {
                            "Sparql": "PREFIX ns: <http://rdf.
                               freebase.com/ns/>\nSELECT
                               DISTINCT ?x\nWHERE {\nFILTER (?x
                                != ns:m.07_mj3)\nFILTER (!
                               isLiteral(?x) OR lang(?x) = ''
                               OR langMatches(lang(?x), 'en'))\
                               nns:m.07_mj3 ns:base.
                               schemastaging.athlete_extra.
                               salary ?y .\n?y ns:base.
                               schemastaging.athlete_salary.
```

```
    team ?x .\n}\n",
"AnnotatorComment": {
    "FreeFormComment": "First-round
        parse verification",
    "QuestionQuality": "Good",
    "ParseQuality": "Complete",
    "Confidence": "Normal"
},
"Answers": [
    {
        "AnswerArgument": "m.0bl8l
            ",
        "EntityName": "Aston Villa
            F.C.",
        "AnswerType": "Entity"
    }
],
"Time": null,
"AnnotatorId": 0,
"Constraints": [],
"TopicEntityMid": "m.07_mj3",
"TopicEntityName": "Stephen Ireland
    ",
"InferentialChain": [
    "base.schemastaging.
        athlete_extra.salary",
    "base.schemastaging.
        athlete_salary.team"
],
"PotentialTopicEntityMention": "
    stephen ireland",
"Order": null,
"ParseId": "WebQTest-762.P0"
```

```
        }
    ],
    "utterance": "who has stephen ireland
       played for",
    "id": 5,
    "RawQuestion": "who has stephen ireland
       played for?",
    "targetOrigSparql": "PREFIX ns: <http://rdf
       .freebase.com/ns/>\nSELECT DISTINCT ?x\
       nWHERE {\nFILTER (?x != ns:m.07_mj3)\
       nFILTER (!isLiteral(?x) OR lang(?x) = ''
        OR langMatches(lang(?x), 'en'))\nns:m
       .07_mj3 ns:base.schemastaging.
       athlete_extra.salary ?y .\n?y ns:base.
       schemastaging.athlete_salary.team ?x .\n
       }\n",
    "ProcessedQuestion": "who has stephen
       ireland played for",
    "results": [
        "Aston Villa F.C."
    ],
    "QuestionId": "WebQTest-762"
},
{
    "Parses": [
        {
            "Sparql": "PREFIX ns: <http://rdf.
               freebase.com/ns/>\nSELECT
               DISTINCT ?x\nWHERE {\nFILTER (?x
                != ns:m.07_mj3)\nFILTER (!
               isLiteral(?x) OR lang(?x) = ''
               OR langMatches(lang(?x), 'en'))\
               nns:m.07_mj3 ns:sports.
```

```
        pro_athlete.teams ?y .\n?y ns:
        sports.sports_team_roster.team ?
        x .\n}\n",
    "AnnotatorComment": {
        "FreeFormComment": "First-round
            parse verification",
        "QuestionQuality": "Good",
        "ParseQuality": "Complete",
        "Confidence": "Normal"
    },
    "Answers": [
        {
            "AnswerArgument": "m.01634x
              ",
            "EntityName": "Manchester
              City F.C.",
            "AnswerType": "Entity"
        },
        {
            "AnswerArgument": "m.01cwm1
              ",
            "EntityName": "Stoke City F
              .C.",
            "AnswerType": "Entity"
        },
        {
            "AnswerArgument": "m.0bl8l
              ",
            "EntityName": "Aston Villa
              F.C.",
            "AnswerType": "Entity"
        },
        {
```

```
                "AnswerArgument": "m.0fvly
                   ",
                "EntityName": "Newcastle
                   United F.C.",
                "AnswerType": "Entity"
            }
        ],
        "Time": null,
        "AnnotatorId": 0,
        "Constraints": [],
        "TopicEntityMid": "m.07_mj3",
        "TopicEntityName": "Stephen Ireland
           ",
        "InferentialChain": [
            "sports.pro_athlete.teams",
            "sports.sports_team_roster.team
               "
        ],
        "PotentialTopicEntityMention": "
           stephen ireland",
        "Order": null,
        "ParseId": "WebQTest-762.P1"
    }
],
"utterance": "who has he played for",
"id": 6,
"RawQuestion": "who has he played for?",
"targetOrigSparql": "PREFIX ns: <http://rdf
   .freebase.com/ns/>\nSELECT DISTINCT ?x\
   nWHERE {\nFILTER (?x != ns:m.07_mj3)\
   nFILTER (!isLiteral(?x) OR lang(?x) = ''
    OR langMatches(lang(?x), 'en'))\nns:m
   .07_mj3 ns:sports.pro_athlete.teams ?y
```

```
            .\n?y ns:sports.sports_team_roster.team
            ?x .\n}\n",
        "ProcessedQuestion": "who has he played for
            ",
        "results": [
            "Manchester␣City␣F.C.",
            "Stoke␣City␣F.C.",
            "Aston␣Villa␣F.C.",
            "Newcastle␣United␣F.C."
        ],
        "QuestionId": "WebQTest-762"
    }
  ],
  "TopicEntityMid": "m.07_mj3"
}, ... ]}
```


6.3 Basic Conversational Dataset

In the basic case there is one conversation for each entity. Entities that only have one question make a one-question conversation with no additional pronoun-questions. The basic conversational training dataset contains 1720 conversations, the longest one has 27 questions, the average conversation contains 2 questions, and the training dataset has around 3k questions in total. The evaluation set contains 1072 conversations, the longest one has 14 questions, the average conversation contains 1.7 questions, and the conversational training dataset has 1815 questions in total.

6.4 Extended Conversational Dataset

We created an extended version of the training conversational dataset to improve the trained model. The main two ideas to augment the training dataset are:

1. For entities that have more than one query we call the first query (without the entity-pronoun replacement in a conversation) a leading question. If an entity has N questions we create N conversations. In each conversation one of the N questions is a leading question. A conversation always starts with a different question. We get different conversations that consist of the same questions. As an example of a 2-question set {*Where was Albert Einstein born?*, *Who was Albert Einstein married to?*} for an entity Albert Einstein: The first conversation would be:

   - *Where was Albert Einstein born?*
   - *Who was he married to?*

   The second conversation would be:

   - *Who was Albert Einstein married to?*
   - *Where was he born?*

2. For entities with only one question we create one conversation that contains two queries. The first query is the original query, and the second one is the same query but with the entity replaced with a corresponding pronoun. Basically what we get is a conversation where the second question is a pronoun-repetition of the first question. As an example:

   First question: *Where was Albert Einstein born?*
   Second question: *Where was he born?*

The extended training set contains 3453 conversations, the longest one has 27 questions, the average conversation contains 2 questions, and the training dataset has around 3k questions in total. These two approaches to augment the training dataset are relatively easy but efficient. In the evaluation part we will show how training on the extended dataset helped to improve the overall performance.

7 Evaluation

The conversational question answering is a significantly more difficult task than the single question answering. The system has to read the context and apply it to find the right answer. In this part of the work we will evaluate the non-conversational and conversational Aqqu to see how the context tracking functionality influences the overall precision of the question answering process with Aqqu.

To evaluate the system we used the code from

`https://ad-git.informatik.uni-freiburg.de/ad/aqqu-webserver`

as a base and adapted it for the conversational data. The adapted code can be found under:

`https://ad-git.informatik.uni-freiburg.de/ad/aqqu-webserver/tree/conversational_gender`

7.1 Evaluation Metrics

- $q_1, \ldots, q_n$: questions
- $c_1, \ldots, c_i$: the answer candidates
- $g_1, \ldots, g_n$: multiple answers of the gold answer

- $a_1, \ldots, a_n$: the answers from the system for the first candidate

- GA-Size: Gold answer size is the number of ground truth answers ($|g_1, g_2, \ldots, g_n|$).

- BCA-Size: Best candidate answer size is the number of answers of the first candidate ($|a_1, a_2, \ldots, a_n|$)).

- Candidates: The number of all predicted candidates ($|c_1, c_2, \ldots, c_i|$).

- Precision: The precision shows what percentage of the answers from the best candidate are correct.

$$\text{Precison} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

where TP is a true positive, i.e. $TP = |a_k, \ldots, a_m|$, where $a_k, \ldots, a_m$ are correct answers and FP is a false positive, i.e. $FP = |a_l, \ldots, a_p|$, where $a_l, \ldots, a_p$ are false answers.

For example:

Utterance	GA-size	GA	BCA-Size	BCA
who does ronaldinho play for now 2011?	2	"Brazil national football team" "Clube de Regatas do Flamengo"	2	"Clube Atlético Mineiro" "Clube de Regatas do Flamengo"

Only one answer out of two is correct and only one correct answer is found. Therefore TP = 1, FP = 1, Precision = 1/(1+1) = 0.5

- Recall. The recall measures how well the system finds correct answers, i.e. what percentage of correct answers are found.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

where FN is false negative, i.e. $FN = |g_l, \ldots, g_p|$, where $g_l, \ldots, g_p$ are correct answers that were not found by the system. For example:

Utterance	GA-size	GA	BCA-Size	BCA
what state does romney live in?	1	"Massachusetts"	2	"Massachusetts" "Bloomfield Hills"

Only one answer out of two is correct and only one correct answer is found. Therefore TP = 1, FP = 1, Precision = 1/(1+1) = 0.5

- F1: It is the harmonic average of the precision and recall. The best value is 1 (Precision → 1 and Recall → 1) and the worst is 0 (Precision → 0 and Recall → 0).

$$\text{F1} = 2\frac{\text{Precison} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Parse Match: This parameters shows if the candidate relation that gives an answer with the best F1 score is matched perfectly to the ground truth (>0.99 matching).

7.2 Averaged Evaluation Metrics

- Questions: Total number of questions in the evaluation dataset.

- Average Precision: The average precision of all questions.

- Average Recall: The average recall of all questions.

- Average F1: The average F1 across all questions.

$$\text{average F1} = \frac{1}{n}\sum_{i=1}^{n} F1(g_i, a_i)$$

- Accuracy: The percentage of queries answered with the exact gold answer.

$$\text{accuracy} = \frac{1}{n}\sum_{i=1}^{n} I(g_i = a_i)$$

- Parse Accuracy: Average parse match across all questions.

7.3 Example of an Evaluation on a Small Dataset

As an example we have done an evaluation for a *tiny* dataset (*he_ data_ tiny.json*, consists of 14 questions). The dataset has a conversational structure with pronoun-replacement and no gender-identification. The entities are only replaced with *he*.

Lets look at the evaluation results.

ID	Utterance	GA-Size	BCA-Size	Candidates	Precision	Recall	F1	Parse Match
0	what time zone is chicago in right now?	1	1	3	1	1	1	True
1	where to stay there tourist?	1	1	21	1	1	1	True
2	who does ronaldinho play for now 2011?	2	2	21	0.5	0.5	0.5	False
3	what is ella fitzgerald name?	2	2	9	1	1	1	True
4	what state does romney live in?	1	2	10	0.5	1	0.67	False
5	where did his parents come from?	1	1	31	1	1	1	True
6	what university did he graduated from?	1	6	21	0	0	0	False
7	where did he graduated college?	1	6	19	0	0	0	False
8	what colleges did he attend?	5	6	1	0	0	0	False
9	when did he become governor?	1	1	6	0	0	0	False
10	where is his family from?	1	1	8	0.5	1	0.67	False
11	what degrees does he have?	3	6	1	0	0	0	False
12	who does jeremy shockey play for in 2012?	1	1	19	1	1	1	True
13	what does bolivia border?	5	5	10	1	1	1	False

Table 1: Evaluation on a tiny dataset.

Questions	14
Average Precision	0.65
Average Recall	0.82
Average F1	0.689
Accuracy	0.500
Parse Accuracy	0.357

Table 2: Average results for tiny dataset.

8 Experiments

We have conducted five different experiments with the Aqqu system trained on:

1. Original dataset – Experiments 1-3
2. Basic Conversational Dataset – Experiment 4
3. Extended Conversational Dataset – Experiment 5

8.1 Experiment 1

- data with conversational structure and without pronoun replacement
- without gender identification
- not trained with context tracking
- used conversations_WebQSP.json

Training Dataset	Original Dataset
Questions	1815
Average Precision	0.67
Average Recall	0.72
Average F1	0.657
Accuracy	0.478
Parse Accuracy	0.510

Table 3: Results of experiment 1.

8.2 Experiment 2

- data with conversational structure and with pronoun replacement
- without gender identification
- not trained with context tracking
- used he_conversational_WebQSP

Training Dataset	Original Dataset
Questions	1815
Average Precision	0.50
Average Recall	0.55
Average F1	0.495
Accuracy	0.355
Parse Accuracy	0.362

Table 4: Results of experiment 2.

8.3 Experiment 3

- data with conversational structure and with pronoun replacement
- with gender identification

- not trained with context tracking
- used WebQSP_conversation_gender_test.json

Training Dataset	Original Dataset
Questions	1815
Average Precision	0.60
Average Recall	0.65
Average F1	0.586
Accuracy	0.417
Parse Accuracy	0.440

Table 5: Results of experiment 3.

8.4 Analysis of Experiments 1-3

In the evaluation results for the system that was trained on non-conversational data, we can see that the performance of a conversational system is inferior, when compared to a non-conversational one. Also we see that the system with gender identification performs better than the system without it. Therefore we conduct the following experiments with gender identification.

8.5 Experiment 4

- data with conversational structure and with pronoun replacement
- with gender identification
- trained with context tracking on basic conversational dataset
- used WebQSP_conversation_gender_test.json

Training Dataset	Original Dataset
Questions	1815
Average Precision	0.59
Average Recall	0.63
Average F1	0.575
Accuracy	0.417
Parse Accuracy	0.440

Table 6: Results of experiment 4.

8.6 Experiment 5

- data with conversational structure and with pronoun replacement
- with gender identification
- trained with context tracking on extended conversational dataset
- used WebQSP_conversation_gender_test.json

Training Dataset	Original Dataset
Questions	1815
Average Precision	0.60
Average Recall	0.645
Average F1	0.584
Accuracy	0.420
Parse Accuracy	0.425

Table 7: Results of experiment 5.

9 Conclusions

We have added a context tracking functionality into the Aqqu question answering system and called it conversational Aqqu. The context tracking functionality is based on the replacement of some entities in the queries with the corresponding pronouns. We have also developed a web chatbot called Aqqu Chatbot. The Aqqu Chatbot uses the conversational Aqqu as the backend.

We have conducted a set of experiments to evaluate the performance of the conversational Aqqu. In the results of the experiments we can see that the conversational Aqqu performs slightly worse than the original Aqqu system. The F1 score of the original Aqqu is 65.7% and the best conversational Aqqu result has the F1 score of 58.6%. There is an accuracy decrease of 7.1%.

The possible reasons for the errors could be:

- texiit Mistakes during the generation of pronoun questions for conversational datasets which may result in an unnatural grammar in the questions for training and evaluation.

- Mistakes in gender identification during question answering that may cause a wrong context identification, which leads to errors in question answering.

- The context gives the system more entities to process. The more entities – the more difficult it is to prune the incorrect answers and rank the correct ones.

There is more room for errors and the probability of having the correct answer being pruned or getting a low ranking score is higher.

However the conversational question answering is more convenient and intuitive. The loss of precision is a trade-off to achieve a more human-like question answering.

We have also built in a data augmentation functionality in the Aqqu Chatbot. When a user gets an answer for a question in the Aqqu Chatbot, if there is a correct answer among all the answer candidates, the user can mark this answer as correct. The question-answer pair will be saved in the WebQSP format. With this functionality it is very convinient to collect more high-quality data for further training and evaluation of the Aqqu question answering system.

10 Future Work

The conversational question answering with Aqqu can be further improved. It is possible to add a new feature, that indicates whether a candidate is generated from context or from a current question.

Also the data augmentation functionality can be developed to additionaly save the context of a question-answer pair in the WebQSP format. This would further enhance the training.

A good idea could be to collect more data with the Aqqu Chatbot data augmentation functionality and train the system on a bigger dataset.

The Aqqu system only tracks the pronoun-based context. It could be possible to implement other approaches, for example for the system to also process additional single interrogative word questions: "Where?", "When?", etc.

11 Useful Links

Aqqu website:

`http://aqqu.informatik.uni-freiburg.de`

The information how to train, build and run the Aqqu system backend:

`https://ad-git.informatik.uni-freiburg.de/ad/Aqqu`

Aqqu backend with conversational tracking functionality:

`https://ad-git.informatik.uni-freiburg.de/ad296/Aqqu`

The Aqqu chatbot:

`https://ad-git.informatik.uni-freiburg.de/ad296/AqquChatterbot/tree/Gender`

The script for converting a dataset to a conversational dataset:

`https://ad-git.informatik.uni-freiburg.de/ad/Aqqu/create_data_set.py`

The evaluation code:

`https://ad-git.informatik.uni-freiburg.de/ad/aqqu-webserver/tree/conversational_gender`

Bibliography

[1] Hannah Bast, Elmar Haussmann. *More Accurate Question Answering on Freebase.*. Department of Computer ScienceUniversity of Freiburg, 79110 Freiburg, Germany.

[2] Siva Reddy, Danqi Chen, and Christopher D Manning. *Coqa: A conversational question answering challenge.*. arXiv preprint arXiv:1808.07042, 2018.

[3] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. *Quac: Question answering in context.*. arXiv preprint arXiv:1808.07036, 2018.

[4] Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. *A dataset and baselines for sequential open-domain question answering.*. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 1077–1083, 2018.

[5] Chenguang Zhu , Michael Zeng , Xuedong Huang. *SDNet: Contextualized Attention-based Deep Network for Conversational Question Answering.*. Microsoft Speech and Dialogue Research Group, Redmond, WA 98052, USA

[6] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou and Jian Yin. *Dialog-to-Action: Conversational Question Answering Over a Large-Scale Knowledge Base.*. The School of Data and Computer Science, Sun Yat-sen University. Guangdong

Key Laboratory of Big Data Analysis and Processing, Guangzhou, P.R.China
Microsoft Research Asia, Beijing, China

[7] V. I. Spitkovsky and A. X. Chang. *A Cross-LingualDictionary for English Wikipedia Concepts.*. InLREC,pages 3168–3175, 2012.

[8] ClueWeb, 2012. The Lemur Projekt.

[9] E. Gabrilovich, M. Ringgaard, and A. Subramanya. FACC1: Freebase annotation of ClueWeb corpora,Version 1

[10] https://amritasaha1812.github.io/CSQA/

[11] https://www.wikidata.org