COALA – A Rule-Based Approach to Answer Type Prediction

Nadine Steinmetz and Kai-Uwe Sattler

Technische Universität Ilmenau, Germany
firstname.lastname@tu-ilmenau.de

Abstract. For answering a question correctly, the previous detection of the answer type is essential. Especially in the field of Question Answering (QA) over knowledge bases, answers might be of many different types as natural language is ambiguous and a question might lead to different relevant queries. For semantic knowledge bases data types (such as date, string, or number) as well as all ontology classes (such as athlete, championship, or television show) have to be taken into account. Therefore, the previous detection of the answer type is a helpful sub-task for QA systems, but also a complex classification problem. We present our rule-based approach *COntext Aware anaLysis of Answer types* (COALA). Our approach is based on the extraction of several question features and the context aware disambiguation to retrieve the correct answer type. COALA has been developed in the course of the SMART task challenge and we evaluated our approach based on over 21,000 questions.

Keywords: Question Answering · Answer Type Detection · Semantic Web

1 Introduction

Answer type prediction (ATP) is an essential subtask for automatic question answering (QA). It can either be considered a classification task or an analytical problem. For predicting the answer type using a classifier, a large dataset for annotation and obviously human annotators are required. Consequentially, the classifier is rather determined to a specific version of the ontology and the knowledge base (KB). If ontology classes, properties, or knowledge facts are removed or added, the classifier requires to be retrained utilizing additionally annotated source data. In contrast to that, an analytical approach is rather flexible. The prediction task mostly requires mapping processes to the underlying KB. In case of changes to the ontology or the KB, the data source is adapted while the mapping processes and the analysis in general stay the same. Changes in the underlying ontology might be rare for well established knowledge bases. But especially for newly created KBs and ontologies, changes are common

and trained approaches might not take into account these changes. In contrast, changes within the KB in terms of new entities and facts are quite frequent. Here, an analytical approach is able to adapt to the changes on the contrary to pre-trained classifiers.

With this paper, we introduce our analytical approach to detect the answer type of an input question utilizing rules. We analyzed a large amount of questions provided with the training dataset for the SMART task challenge aligned with ISWC 2020 [5]. We extracted question features to be able to find the correct references for the detection of the answer type. Following this approach, we defined a set of rules and developed a plain but efficient application which is able to detect the answer types for the majority of questions wrt. the SMART datasets.

The remainder of the paper is organized as follows: Section 2 covers some related approaches on ATP and QA. We describe our approach in Section 3. Based on the training and the test dataset provided by the SMART task challenge, we evaluated our approach and present the results in Section 4. We also discuss flaws and errors in that section and give an outlook on improvements and future work in Section 5.

2 Related Work

Our approach – along with ATP approaches in general – contributes to a higher accuracy of QA systems. More specifically, the answer and answer types are part of a semantic knowledge graph. For this purpose, several approaches have been developed and evaluated, especially in the course of the QALD challenge[1]. Developers of QA systems face several challenges to overcome. Unger et al. provided a fundamental publication on Semantic QA systems [8]. And Höffner et al. presented a survey on the challenges of QA systems [3]. The prediction of answer types is a subtask within the procedure of transforming a natural language question to a formal query and ranking the results. Therefore, hereinafter we will examine related work with a focus on the answer type prediction instead of the complete QA system. Mostly, previous work deals with a very limited amount of types to be predicted. For instance, Murdock et al. presented their approach on typing answer candidates in the course of the development of IBM Watson[2]. In consequence, the authors did not restrict the answers to a specific amount of types, because "nearly any word can be used as a type" [6]. Bast et al. presented an approach on question answering for Freebase [2]. For their system, they included a simple answer type classification based on whether the question asks for a person, a place or a date.

This procedure is similar to the answer type check we included for our QA system HYDRA [7]. In addition to the types utilized by Bast et al., HYDRA checks for the answer type *number*, when the question starts with *how many* or *how much*.

[1] http://qald.aksw.org/
[2] https://www.ibm.com/watson

For QA on a knowledge graph, the amount of different types depends on the number of classes defined in the underlying ontology and additionally the types utilized as range for the ontology properties, such numbers, dates etc. Therefore, the typing of expected answers requires an intelligent but efficient approach.

Ziegler et al. presented an approach for efficiency-aware answer type prediction for compositional questions [9]. The idea is to decompose a complex question and retrieve the most specific types for all sub-questions. Similar to our approach, they use CoreNLP POS tagging to find references for types in the questions. But, there is no differentiation between different patterns as well as multiple references from which some might not be constructive and rather be misleading.

For our approach, we examined a large amount of questions to identify the position and pattern of references to the expected answer type of a question. Although we analyze the context of the question to disambiguate answer type candidates, we also keep in mind the efficiency of the subtask within the complete QA pipeline.

3 Method

Our analytical approach includes several separate processing steps. The first step is the detection of the question type. We describe this step in detail in Section 3.1. According to the detected question type, the question is analyzed for further characteristics. These subsequent analysis steps are specified in Section 3.2. In general, the result of the question analysis is the answer category and type(s). Our overall approach is depicted in Figure 1 and further described in detail in the next sections.

3.1 Question Type Detection

For the SMART task challenge, a set of answer types respectively categories have been predefined:

- boolean
- literal – number
- literal – date
- literal – string
- resource – set of ontology classes

These answer types can be deduced from different question types. Therefore, we identified different question types in the first step. Naturally, we first analyzed the first words of the questions in the training dataset (the numbers in the brackets depict the amount of occurrence in the training set of the challenge having types of the DBpedia ontology):

- Most frequently, questions start with *which* or *what* ($\approx$ 7,514 – 42%)

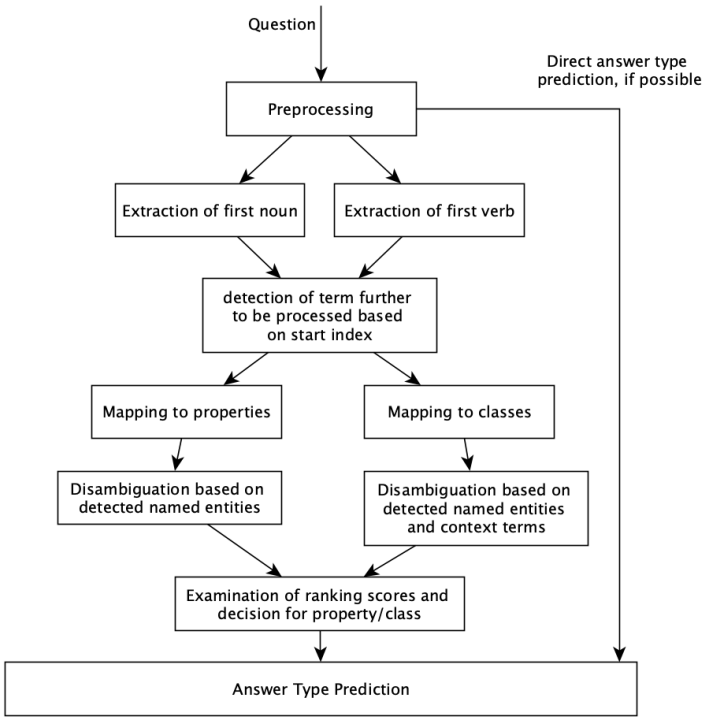

Fig. 1. Overview of our approach

- Questions starting with an auxiliary verb, such as *is* or *did*, are second most frequent ($\approx 2,841$ – 16%)
- There are questions that begin with a main verb and the questions are more a request, such as *list*, *give*, *show* ($\approx$ 1,283 – 7%)
- Questions starting with *who*, *whose* or *whom* are also very frequent ($\approx$ 2,189 – 12%)
- There are several hundred questions either starting with *when* ($\approx$ 1,146 – 6%), *where*($\approx$ 389 – 2%), and *how much*, or *how many*($\approx$ 845 – 5%).
- In addition to the main question types above, a considerable high amount of questions in the dataset start with an entity, as in *Paris is the capital of which country?* ($\approx$ 1,200 – 7%).
- Naturally, there is a relatively low number of further questions with miscellaneous beginnings.

Obviously, many answer types can be deduced easily from the first words of a question.

Questions starting with an auxiliary verb, such as *Was Albert Einstein a vegetarian?* require a `boolean` as answer type. Within the dataset, out of 2,799 questions having `boolean` as answer type, 2,764 start with an auxiliary verb.

Questions starting with *when*, such as *When did Tycho Brahe start working in Uraniborg?*, require a date as answer type. The dataset contains approx. 1,486 questions assigned with the answer type `date` of which 1,146 start with the word *when*.

Questions starting with *how much* or *how many*, such as *How many organizations work for Environmentalism?*, obviously require the answer type to be `number`. The dataset contains 1,633 questions assigned with the answer type `number` and 770 of them start with the words *how much* or *how many*.

Consequently, we deduce the answer types of these question types directly from the first words of the sentence.

For questions starting with *who* or *where*, the general answer type can be anticipated from the first words with `dbo:Person` and `dbo:Place` respectively. But, in order to detect the more specific type, we perform an elaborate analysis on these questions.

The same applies for all other question types for which the answer type cannot be deduced directly from the first words. For these remaining questions, all answer types are taken into account and we try to detect the answer type according to several processing steps.

Our analysis approach regarding the different question types is described in detail in the following sections.

3.2 Question Analysis

This section describes the analysis of a question in detail. We first perform some preprocessing steps to prepare the question for further steps. Answer types are mainly derived from the ontology classes that are directly asked using a (combined) noun, or the range of an ontology property that is referenced in the question by a verb or noun. For instance, the question *Which*

languages were influenced by Perl? requires the answer to be an instance of `dbo:ProgrammingLanguage` and the class is referenced by the noun *languages*. For the question *Who is starring in Spanish movies produced by Benicio del Toro?* the referencing verb is *starring* and not the (combined) noun *Spanish movies*. Therefore, we extract the first occurrence of a noun or the first occurrence of a main verb. Indications and methods are described in detail in the next sections.

Preprocessing In the first step, the questions are cleaned for redundant quotation/punctuation marks and other characters not necessary for the analysis. Subsequently, the question is analyzed for Part-of-Speech (POS) using the CoreNLP annotation pipeline[3]. The annotator pipeline consists of the annotators `tokenize`, `ssplit`, `pos`. As the test and training datasets contain questions collected from web inquiries, the capitalization is not correct in some cases and we do not find any relevant combined nouns or verbs. In that case, we add the `truecase` annotator to the pipeline.

For the identification of the key term that hints the answer type, we follow two different strategies:

- extraction of the first noun, and
- extraction of the first verb.

These strategies have emerged from several analyses on the training dataset. For instance, we compared the labels of the ontology classes in the type list with the questions and extracted the POS for the words with the highest similarity to the class labels. After several (manual) evaluation rounds, these two strategies turned out to be the most effective.

First Noun Extraction Consider the question *What is the birth place of Konrad Adenauer?*. Here, the (combined) noun *birth place* references a property in the ontology and we need to examine the range of this property. Furthermore, consider the question *What team does John McGeever play for?*. The noun *team* references a class from the underlying ontology which represents the answer type of the question. Eventually, the request *Give me the number of home stadiums of teams managed by John Spencer.* references the answer type with the word *number*. For all these questions, the first noun depicts the phrase that requires to be extracted to further examine the answer type of the question.

There are a few rules we follow to do that:

- The noun must follow the first *which/what* in the sentence, if these words occur in the question. For instance, the question *The capital of Vilnius is in which sovereign state?* requires the answer to be a *sovereign state* and not a *capital*.

[3] https://stanfordnlp.github.io/CoreNLP/

- We take into account several sequential combinations of word types regarding Part-Of-Speech: [JJ N IN N], [N IN N], [JJ N], [N IN DT N], [N]. Table 1 shows sample phrases for each combination. For this task, another option would be the usage of a language model (cf. [1]). A comparison of evaluation results will be included in future approaches.
- A combination might be followed by one our more simple nouns, such as [JJ NN NN] as in *official birth place*.
- After extraction of relevant combined nouns, we only consider the longest combination with the lowest start index.
- The combination of nouns must not contain a proper noun as proper nouns reference named entities and do not reference the answer type.
- A combined noun must not be followed by an apostrophe. For the question *What is a neutron's gyromagnetic ratio?* the first noun would be *neutron*, but the referencing combined noun for the answer type is *gyromagnetic ratio*.

Table 1. Sample phrases for combined nouns and the respective sequential POS pattern.

POS pattern	Sample question with marked terms following the pattern
JJ N IN N	What is the *approximate date of birth* of Eusebius of Caesarea?
N IN N	What is *branch of biology* that starts with z?
JJ N	What is the *official language* of Papua New Guinea?
N IN DT N	What is the *name of the opera* based on Twelfth Night?
N	Which *films* are located in New York City?

Combined nouns are extracted from the question according to these rules – along with the start index to be able to extract the first occurrence and compare them to the start index of the extracted verbs.

Verb Extraction Consider the following sample questions:

- What are Breann McGregor and Anika Knudsen, both known for?
- Who coached the Marquette golden eagles men team in 09 to 10 and then again in 13 to 14?
- Who painted Mona Lisa?

For these questions, the verbs *known for*, *coached*, and *painted* are referencing the required answer type respectively. The CoreNLP tagger uses the Penn Treebank tagset. Within this tagset, the tags for verbs are starting with `VB`, such as `VBZ` for a *Verb, 3rd person singular present* or `VBD` for a *Verb, past tense*. Therefore, we extract the first occurrence of a term tagged by the POS tagger with tags starting with `VB`. Naturally, there are combined verbs, such as *is married*, or *will be dancing*. Therefore, we extract combinations of terms that are tagged with a verb tag consecutively.

Class identification The extracted combined noun is stemmed[4] and mapped to the search index for DBpedia ontology classes. The search index consists of the original labels of the classes, provided by DBpedia, as well as additional labels obtained by utilizing the datamuse API[5] for each original label. In addition, we enriched the search index of the generalized versions of the labels. This means, for each original label, we extracted the last term (only in case the label consists of more than one term) and added this term as additional label. In this way, we still find the ontology class *basketball player* when only *player* is referenced in the question.

Property Identification The extracted noun combination could also reference a property. Hence, we also stem and map the phrase to the search index for properties. Similar to the search index for classes, this index consists of the original labels of the properties and additional labels obtained via datamuse. In contrast to the class label index, we also utilized data from the Wikidata knowledge graph. Wikidata often contains a lot more information on entities in terms of specific characteristics. For instance, for the chemical substance *Malathion* DBpedia only lists one essential property describing the substance (iupac name). Wikidata lists around 20 facts about it. Moreover, many questions in the training and test dataset of the challenge refer to facts only contained in Wikidata. Therefore, we decided to add properties, their ranges and their occurrence with entities to our search index.

Disambiguation Due to the enrichment of the search indices for properties and classes with additional labels, we are able to find the relevant classes and properties for a higher amount of natural language phrases. But, concurrently we created ambiguous mappings and it might occur to obtain more than one class or property for an extracted phrase. Therefore, the disambiguation step is necessary to find the most appropriate and relevant property or class.

For the disambiguation between different classes, it is essential to semantically distinguish them from each other. For an algorithm-based disambiguation, context information for each class is necessary. Unfortunately, the DBpedia ontology does not provide much information about each class. Therefore, we created context texts by gathering all abstracts of the instances of each class. We use these texts to map the context information from the question to the context information of each relevant class for the extracted phrase. In this way, a ranking score for each class is calculated by simply counting the occurrences of contextual terms from the question within the context information of each class. All scores are normalized to a range between [0.0...1.0].

Another option for the disambiguation of classes is the utilization of Wikipedia page links and check, if the relevant classes are linked directly or indirectly to named entities mentioned in the question – via their instances. We identify

[4] Utilizing the Snowball Stemmer: https://snowballstem.org/
[5] https://www.datamuse.com/api/

named entities within the question, retrieve all resources (in case of ambiguous surface forms) and check, if these resources are instances of the relevant classes or if there are links between the resources and the instances of the classes. Again, we simply count the number of links and normalize the number to a ranking score between [0.0...1.0].

For the disambiguation of phrases that are mapped to ontology properties, we utilize the named entities mentioned in the question alike. Here, we retrieve all properties that are part of a triple with the named entities (as subject or object). We then check, if the list of properties mapped from the extracted noun phrase contains properties that are associated with the named entities identified in the question. A property achieves a ranking score by simply counting the named entities it is associated with.

In this way, all classes and properties resulting from the mapping of the extracted noun phrase achieve a ranking score. Naturally, the class respectively property with the highest score is selected as most relevant for the extracted phrase. In case of a class, it constitutes the answer type at once. In case of a property, the range of the property is assigned as answer type for the question.

4 Evaluation

Our work has been developed as part of the SMART task challenge at ISWC 2020. For this challenge, a training dataset and eventually a test dataset has been published. The training dataset consists of 17,571 items of which 44 items do not contain a proper question. Therefore, we processed 17,527 questions for the training dataset. The test dataset consists of 4,381 items of which 3 items do not contain a proper question. Therefore, we processed 4,378 questions for the test dataset. Both datasets are publicly available[6].

As we developed a rule-based approach (in contrast to a trained approach), we did not split up the training dataset for evaluation issues and provide results for the complete dataset. The system output consists of a result category which is either `literal`, `boolean`, or `resource`, and a (list of) type(s). The types depend of the category. For the `literal` category, the type list either contains `string`, `date`, or `number`. If the category is `resource`, the type list consists of several ontology classes from the DBpedia ontology. Therefore, the evaluation includes three different metrics: accuracy, NDCG@5, and NDCG@10. The accuracy score is used as metric for the category prediction of the system output. NDCG@5 and NDCG@10 are used to calculate the quality of the ranking of the type list. NDCG@5 is calculated of the first 5 results in the type list and NDCG@10 for the first 10 results respectively.

4.1 Results

Evaluation results are presented in Table 2. The results show that our approach achieved similar results for the training and the test dataset having better re-

[6] https://github.com/smart-task/smart-dataset/tree/master/datasets/DBpedia

sults for the class prediction of the training dataset. Out of 17,527 and 4,378 questions, 14,272 respectively 3,683 questions were answered by our approach. The remaining questions – 3,255 and 695 respectively – have an answer type to be detected as `unknown`.

Table 2. Evaluation results for the SMART task challenge datasets against the DBpedia ontology

Dataset	Processed	Answered	Accuracy	NDCG@5	NDCG@10
Training	17,527	14,272	0.746	0.655	0.620
Test	4,378	3,683	0.744	0.540	0.521

4.2 Discussion

The reasons for failures of our approach can be summarized in the following categories:

1. Follow-up errors because of erroneous Part-of-Speech tagging
2. High complexity of the question
3. Missing mapping from verb/noun to class and property labels
4. Incomprehension of questions

We discuss these issues further in detail in the following paragraphs:

Follow-Up errors The questions have been derived from web questions and often lack the correct capitalization or are completely in upper case etc. Therefore, the POS tagging has a higher error rate than for correct sentences. Our approach is based on the correct extraction of the first noun or verb. If that fails, the approach fails in most cases. As stated in Section 3.2, we add the `truecase` annotator to the CoreNLP pipeline, but only when no noun or verb could be extracted. Therefore, the error rate is still considerable high because of wrong capitalization. Sometimes, we are able to derive a general class from the characteristic of the question. For instance, if the question starts with *who* or *where*, we add the respective ontology class `dbo:Person` or `dbo:Place`. But this results in a lower ranking score, if the expected class is more specific.

Complexity Some questions are quite complex and are even hard to answer at all. So, the detection of the answer type is also more complex and cannot be achieved by our straight forward approach. For instance, the question *What is the songwriter of Hard Contract known for?* requires to first detect who the songwriter of the song *Hard Contract* is. As the property `dbo:knownFor` has not a range constraint, the concrete object for the songwriter having this property must be retrieved. Subsequently, the type can be deduced. But, the answer type detection should be a first step for a QA approach and supporting the result

of the complete approach. At the end of the detection for this sample question, we already have the answer for the question – after an elaborate procedure. From our point of view, the answer type detection should be a tradeoff between required cost and result. In addition, the dataset also contains questions which we are unable to answer as humans. Naturally, it is hard to find rules for those type of questions. But, of course, the limits of artificial intelligence should not be the individual human brain. So, this is an issue for future work.

Missing mapping As described in Section 3.2, we utilize the original labels of the classes and properties derived from DBpedia (for classes and properties) and Wikidata (only for properties). For the properties, Wikidata also provides additional labels via `skos:altLabel`. In addition, we retrieved more labels by utilizing the datamuse API and thereby enriched our search index for classes and properties. But still, some phrases cannot be mapped to a property or class and therefore the approach either fails to predict an answer type at all or detects a wrong type. Therefore, the lexical gap is a major difficulty for the field of Semantic Question Answering and remains an aspect to work on further.

Incomprehension In addition to very complex questions, the dataset also contains a few items where the question is rather an abstract than a question (cf. question with id `dbpedia_4857`) and it is not clear what it asks for. Also, there are questions like *What it is?* which we naturally skipped, too.

5 Summary

With this paper, we present our straightforward rule-based approach COALA on answer type prediction. As ATP is supposed to be a subtask of a QA system, we focussed on a simple, but efficient procedure. As shown in Section 4, we achieved an accuracy on the category prediction of almost 75% for both, the training and the test dataset. The NDCG score for the ranking quality of the type lists ranges between 52% and 65%.

As discussed in Section 4.2, we identified several error categories and future work tasks. We expect the most effects on accuracy and ranking quality by investigating more in the comprehension of complex questions. We already achieve a good result with the identification of the references for the answer types. But, questions with sub-sentences and listings of required characteristics for the demanded answer are still a challenge. Therefore, we plan to take further investigations in lexical patterns to identify all references mentioned in a question and take these hints into account for more specific predictions of the answer type. Another challenge is the missing mapping of natural language phrases to the labels of ontology classes and properties. For this issue, we plan to examine the ClueWeb12 dataset[7] and find additional phrases similar to Ziegler et al.[9] and Lin et al.[4].

[7] http://lemurproject.org/clueweb12/index.php

Overall, the typing of answer candidates should an efficient and contributing subtask of QA systems. With our approach, we anticipate some tasks that are necessary for the construction of the correct SPARQL query in any case, such as question type analysis and named entity identification and disambiguation. In this way, the process is contributing to the complete QA procedure and facilitates the expected type of answer for a higher accuracy of the overall system.

References

1. Akbik, A., Blythe, D., Vollgraf, R.: Contextual string embeddings for sequence labeling. In: Proceedings of the 27th International Conference on Computational Linguistics. pp. 1638–1649. Association for Computational Linguistics, Santa Fe, New Mexico, USA (Aug 2018), https://www.aclweb.org/anthology/C18-1139
2. Bast, H., Haussmann, E.: More accurate question answering on freebase. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. p. 1431–1440. CIKM '15, Association for Computing Machinery, New York, NY, USA (2015)
3. Höffner, K., Walter, S., Marx, E., Usbeck, R., Lehmann, J., Ngonga Ngomo, A.C.: Survey on challenges of question answering in the semantic web. Semantic Web **8**(6), 895–920 (2017)
4. Lin, T., Mausam, Etzioni, O.: Entity linking at web scale. In: Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX). pp. 84–88. Association for Computational Linguistics, Montréal, Canada (Jun 2012), https://www.aclweb.org/anthology/W12-3016
5. Mihindukulasooriya, N., Dubey, M., Gliozzo, A., Lehmann, J., Ngomo, A.C.N., Usbeck, R.: SeMantic AnsweR Type prediction task (SMART) at ISWC 2020 Semantic Web Challenge. CoRR/arXiv **abs/2012.00555** (2020), https://arxiv.org/abs/2012.00555
6. Murdock, J.W., Kalyanpur, A., Welty, C., Fan, J., Ferrucci, D.A., Gondek, D.C., Zhang, L., Kanayama, H.: Typing candidate answers using type coercion. IBM Journal of Research and Development **56**(3.4), 7:1–7:13 (2012)
7. Steinmetz, N., Arning, A., Sattler, K.: From natural language questions to SPARQL queries: A pattern-based approach. In: Datenbanksysteme für Business, Technologie und Web (BTW 2019), 18. Fachtagung des GI-Fachbereichs ,,Datenbanken und Informationssysteme" (DBIS), 4.-8. März 2019, Rostock, Germany, Proceedings. pp. 289–308 (2019). https://doi.org/10.18420/btw2019-18, https://doi.org/10.18420/btw2019-18
8. Unger, C., Freitas, A., Cimiano, P.: An Introduction to Question Answering over Linked Data, pp. 100–140. Springer International Publishing, Cham (2014)
9. Ziegler, D., Abujabal, A., Saha Roy, R., Weikum, G.: Efficiency-aware answering of compositional questions using answer type prediction. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers). pp. 222–227. Asian Federation of Natural Language Processing, Taipei, Taiwan (Nov 2017), https://www.aclweb.org/anthology/I17-2038