Two-stage Semantic Answer Type Prediction for Question Answering using BERT and Class-Specificity Rewarding

Christos Nikas[1,2], Pavlos Fafalios[1], and Yannis Tzitzikas[1,2]

[1] Information Systems Laboratory, FORTH-ICS, Heraklion, Greece,
[2] Computer Science Department, University of Crete, Heraklion, Greece

Abstract. Answer type prediction is a key task in Question Answering (QA) that aims at predicting the type of the expected answer for a user query expressed in natural language. In this paper we focus on semantic answer type prediction where the candidate types come from a class hierarchy of a general-purpose ontology. We model the problem as a two-stage pipeline of sequence classification tasks *(answer category prediction*, *answer literal/resource type prediction*), each one making use of a fine-tuned BERT classifier. To cope with the harder problem of answer resource type prediction, we enrich the BERT classifier with a rewarding mechanism that favors the more specific ontology classes that are low in the class hierarchy. The results of an experimental evaluation using the DBpedia class hierarchy (∼760 classes) demonstrate a superior performance of answer category prediction (∼96% accuracy) and literal type prediction (∼99% accuracy), and a satisfactory performance of resource type prediction (∼78% lenient NDCG@5).

1 Introduction

Question Answering (QA) is a task in the field of Natural Language Processing and Information Retrieval that aims at automatically answering a question posed by a human in a natural language [4]. An important sub-task of QA is the prediction of the type of the expected answer based only on the user question. The majority of existing approaches on this task considers a set of coarse-grained question types, usually less than 50. However, this is quite restrictive for the general case of cross-domain QA where the number of types is very large.

In this paper, we focus on a *two-stage* answer type prediction task where a first step aims at finding the general category of the answer (*resource*, *literal*, *boolean*), while a second step tries to predict the particular literal answer type (*number*, *date*, or *string*, if the predicted category of the first step is *literal*), or the particular *resource class* (if the predicted category of the first step is *resource*). We consider the case where the resource classes belong to a rich class hierarchy of an ontology containing a large number of classes (e.g., >500), and model the

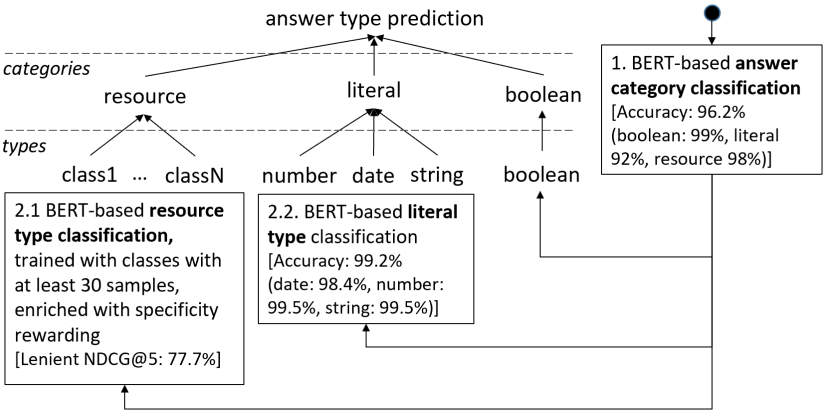

Fig. 1. Two-stage answer type prediction for QA and performance of our proposed methods.

problem as a set of sequence classification tasks, each one making use of a fine-tuned BERT model. For the more fine-grained (and thus more challenging) task of resource class prediction, we propose to enrich the BERT classifier with a rewarding mechanism that favors the more specific ontology classes that are low in the class hierarchy. Fig. 1 depicts this two-stage answer prediction task, the classifiers we use in each different sub-task, and the accuracy of the obtained results. The evaluation results using the DBpedia class hierarchy (∼760 classes) and a ground truth of 40,393 train questions for category prediction, 17,571 for resource/literal type prediction, and 4,393 test questions demonstrate the high performance of our approach. Specifically, we achieve 96.2% accuracy on answer category prediction, 99.2% accuracy on literal type prediction, and 77.7% NDCG@5 on resource type ranking.

The rest of the paper is organized as follows: §2 describes the context, §3 describes our approach, §4 reports the results of the evaluation, and finally, §5 concludes the paper.

2 Context and Datasets

The context of this work is the SMART (SeMantic AnsweR Type) challenge of ISWC 2020[1] [8]. Given a question in natural language, the challenge is to predict the type of the answer using a set of candidates. The problem is modeled as a two-stage classification task: in the first step the task is to predict the general category of the answer (*resource*, *literal*, or *boolean*), while in the second step the task is to predict the particular answer type (*number*, *date*, *string*, or a particular *resource class* from a target ontology).

Two datasets are provided for this task, one using the DBpedia ontology and the other using the Wikidata ontology. Both follow the below structure: Each

[1] `https://iswc2020.semanticweb.org/program/semantic-web-challenges/`

question has a (a) question id, (b) question text in natural language, (c) an answer category (*resource*/*literal*/*boolean*), and (d) answer type. If the category is *resource*, answer types are ontology classes from either the DBpedia ontology (∼760 classes) or the Wikidata ontology (∼ 50K classes). If the category is *literal*, answer types are either *number*, *date*, or *string*. Finally, if the category is *boolean*, answer type is always *boolean*.

An excerpt from this dataset is shown below:

```
[ {
  "id": "dbpedia_14427",
  "question": "What is the name of the opera based on Twelfth Night?",
  "category": "resource",
  "type": ["dbo:Opera", "dbo:MusicalWork", "dbo:Work" ]
 },{
  "id": "dbpedia_23480",
  "question": "Do Prince Harry and Prince William have the same parents?",
  "category": "boolean",
  "type": ["boolean"]
 } ]
```

With respect to the size of the datasets, the DBpedia dataset contains 21,964 questions (train: 17,571, test: 4,393) and the Wikidata dataset contains 22,822 questions (train: 18,251, test: 4,571). The DBpedia training set consists of 9,584 resource, 2,799 boolean, and 5,188 literal questions. The Wikidata training set consists of 11,683 resource, 2,139 boolean, and 4,429 literal questions.

3 Approach

Here we describe our approach for answer type prediction: in §3.1 we provide some background, in §3.2 we describe *question category* prediction, in §3.3 we describe *literal answer type* prediction, and in §3.4 we describe *resource answer type* prediction. The models and code are publicly available at: `https://github.com/cnikas/isl-smart-task`.

3.1 BERT for Sequence Classification

BERT [3], or Bidirectional Encoder Representations from Transformers, is a language representation model based on the Transformer model architecture of [11]. A pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. Because of BERT's massive success and popularity, several methods have been presented to improve BERT on its prediction metrics, by using more data and computational speed [7,12], or by creating lighter and faster models that compromise on prediction metrics [10].

3.2 Question Category Prediction

A question can belong to one of the following three categories: (1) boolean, (2) literal, (3) resource. *Boolean questions* (also referred to as Confirmation questions) only have 'yes' or 'no' as an answer (e.g. *"Does the Owyhee river flow into Oregon?"*). Thus, there is no further classification for this category of questions. *Resource questions* have a specific fact as an answer (e.g. *"What is the highest mountain in Italy?"*) that can be described by a class in an ontology (e.g. `http://dbpedia.org/ontology/Mountain`). *Literal questions* have a literal value as answer, which can be a *number, string*, or *date* (e.g. *"Which is the cruise speed of the airbus A340?"*).

To detect question categories, we fine-tune a BERT model using the Huggingface PyTorch implementation[2]. We choose this model because we approach answer type prediction as a classification problem where each question is a sequence of words. To fine tune BERT we used the training datasets provided for the SMART challenge (described in §2). Specifically, we used questions from both the DBpedia and the Wikidata dataset. Because the data is imbalanced for categories (13.7% boolean, 26.6% literal, 59.4% resource) we randomly sampled questions for each class so that all classes had the same number of samples.

As we will see below, this model achieves 96.2% accuracy on our test set in this prediction task.

3.3 Literal Answer Type Prediction

The answer type for questions that belong in the *literal category* can be: 1) a date, i.e. a literal value that describes a date, 2) number, i.e. a numeric value, or 3) a string, i.e. a text value. Due to the small number of classes (3), it is very effective to train a language model. We again use a fined-tuned BERT model to classify literal questions in one of the 3 types. Similar to question category prediction, we used questions from both the DBpedia and the Wikidata dataset and also randomly sampled questions for each class to cope with class imbalance (29.1% date, 27.3% number, 43.6% string). As we will see, the model achieves 99.2% accuracy for literal questions in our test set.

3.4 Resource Answer Type Prediction

The prediction of the answer type of questions in the *resource* category is a more fine-grained (and thus more challenging) classification problem, because of the large number of types a question can be classified to ($\sim$760 classes on DBpedia and $\sim$50K classes on Wikidata). Therefore, it is not effective to train a classifier on all the ontology classes, especially for open-domain tasks.

To reduce the number of possible types for classification, we selected a subset (C) of all ontology classes, based on the number of samples of each class in the training set. This subset C contains classes that have at least k occurrences in

[2] `https://huggingface.co/transformers/`

the training set. We set $k = 10$ as this number provides a good trade-off between number of classes and performance.[3] The choice of this parameter is described more extensively in section 4.2. The final number of classes in C is 88. Because we chose to train the system on a subset of all the classes, our classifier cannot handle questions with labels that are not included in this subset. To tackle this problem, we replace their labels with the labels of super classes that belong in C. Then we fine tuned a BERT model on them.

Since most questions in the dataset have several answer types ordered by specificity, according to the semantic hierarchy formed in the ontology, in the fine tuning stage we use these questions multiple times, one with each of the provided types as the label. The goal is to find an answer type that is as specific as possible for the question. However, the model may classify a question to a more general answer type in the ontology. To tackle this problem, we 'reward' (inspired by [2]), the predictions of the classes that lie below the top class. The reward of a class c is measured by the depth of the class in the hierarchy, specifically, $reward(c) = depth(c)/depth_{Max}$, where $depth(c)$ is the depth of c in its hierarchy, while $depth_{Max}$ is the maximum depth of the ontology (6 for DBpedia). This means that, after applying normalization and adding the rewards on the output of the model, the top class can be a sub-class that was originally ranked below a more general class. For example, for the question *"What is the television show whose company is Playtone and written by Erik Jendresen?"* the top 5 classes that the classifier predicts are: 1) Work, 2) TelevisionShow, 3) Film, 4) MusicalWork, 5) WrittenWork. Then rewards are applied to classes that are a subclass of Work. After applying the rewards, the top 5 classes are: 1) TelevisionShow, 2) Work, 3) Film, 4) Book, 5) MusicalWork. We can see that TelevisionShow, is now the top prediction, which is both correct and more specific than the previous top prediction (Work).

4 Evaluation

4.1 Evaluation Metrics

We report results for the following metrics:

- *Accuracy*, for category prediction (the percentage of questions classified in the correct category).
- *Precision*, for type prediction (the percentage of the questions for which the top *type* found by the system was one of the types provided in the test dataset, without considering type specificity).
- *Lenient NDCG@k (with a Linear decay)* [1], for resource type prediction.

Lenient NDCG@k, which has been introduced in [1], measures the distance between the predicted type and the most specific type of the answer $d(t, t_q)$.

[3] For the SMART challenge, we had submitted our outputs using k = 30. After further experiments on the training dataset, we changed this value to 10 (more in Sect. 4.2).

Then it converts this distance into a Gain measure, with a linear decay function. The gain is calculated as: $G(t) = 1 - d(t, t_q)/6$, where 6 is the maximum depth of the hierarchy. For example, for the question *"Which company founded by Fusajiro Yamauchi gives service as Nintendo Network?"*, the top 5 classes found as the answer type by our system are: 'dbo:Company', 'dbo:Organisation', 'dbo:University', 'dbo:Agent', 'dbo:RecordLabel' (in this order). The true types specified on the dataset are: 'dbo:Company', 'dbo:Organisation', 'dbo:Agent'. The most specific of these 3 classes is 'dbo:Company', so we calculate the gain for each type found by our system using the distance from the class 'dbo:Company'. Then we compute DCG as: $DCG_p = gain_1 + \sum_{i=2}^{p} \frac{gain_i}{\log_2 i}$. We also compute the ideal DCG ($iDCG$) using the gains of the correct types provided in the dataset, and normalized DCG ($nDCG$) as $\frac{DCG}{iDCG}$. Finally we compute and report the average $nDCG$ over all questions in the test dataset.

4.2 Results on split of the DBpedia training set

Initially, we had no access to the final test dataset of the SMART challenge, so we used 90% of the DBpedia training set[4] as our training dataset and the remaining 10% as our test dataset. For category prediction and literal type prediction we also use the questions from the training dataset for Wikidata for training the classifiers. Our approach achieved the results shown in Table 1. We notice a superior performance of category prediction (96.4% accuracy) and a very high performance of type prediction (83% precision and 79% lenient NDCG@5).

Running the same experiments without the rewarding mechanism, we notice an around 2% drop in the performance (Lenient NDCG) of literal/resource type prediction.

Table 1. Evaluation results

Accuracy (category prediction)	0.964
Precision (literal/resource/boolean type prediction)	0.826
Lenient NDCG@5 with linear decay (literal/resource type prediction)	0.786
Lenient NDCG@10 with linear decay (literal/resource type prediction)	0.778

Tuning of the k parameter To find the optimal value for the parameter k, which is the minimum sample size required to include a class in the subset of classes included in the classifier, we evaluated our system using 4 different values: 5, 10, 30 and 50. Table 2 shows the number of classes included in the classifier for each different value of k and the corresponding performance. We notice that the best results are obtained using k=10, while the results for all other cases are slightly worse.

Error analysis. To better understand the classification performance of category prediction, literal type prediction, and resource type prediction, we inspected their confusion matrices. The results are shown in Table 3. As regards

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

Table 2. Results for different values of k

Value	Classes	NDCG@5	NDCG@10
5	180	0.775	0.765
10	151	0.786	0.778
30	79	0.785	0.772
50	55	0.785	0.748

category prediction, we see that our system classifies in the correct category 99% of the boolean questions, 92% of the literal questions, and 98% of the resource questions. For literal type classification, our system classifies in the correct type 98.4% of date questions, 99.5% of number questions, and 99.5% of string questions. We notice that, for category prediction, most errors occur between the classes literal and resource. For instance, 41 questions of literal type are misclassified as of type resource. As regards resource type prediction, the table shows the confusion matrix for the top-5 (most frequent) resource classes. We notice that there is significant confusion between the classes City and Country, as well as between the class Person and other classes.

Table 3. Confusion matrices for *category* (top left), *literal* (top right), and *resource* (bottom) type prediction.

		Actual			
		Boolean	Literal	Resource	Sum
Predicted	Boolean	287	2	5	294
	Literal	1	497	13	511
	Resource	2	41	905	948
	Sum	290	540	923	1753

		Actual			
		Date	Number	String	Sum
Predicted	Date	120	0	0	120
	Number	2	182	1	185
	String	0	1	191	192
	Sum	122	183	192	497

		Actual					
		Person	City	Country	Award	Organisation	Other
Predicted	Person	148	4	3	3	0	86
	City	3	67	16	0	0	23
	Country	4	2	42	0	0	17
	Award	1	0	0	37	0	0
	Organization	1	2	5	1	32	42
	Other	15	1	8	3	6	351

By manually inspecting several of the misclassification cases, we noticed that some of these errors occur on questions where the correct category is very ambiguous, such as the question *"In what area is Fernandel buried at the Passy Cemetery?"* (labeled as a literal question with type 'string', while our system classifies it as a resource question of type 'dbo:Place'), or the type provided in the dataset is wrong, e.g. the question *"What did the pupil of Mencius die of?"* is labeled as a literal question with type 'date', while our system predicts that the question category is resource and 'dbo:Disease' is one of the predicted classes.

4.3 Results over the final DBpedia test set

After the final test dataset was released, we evaluated our system again, using the script provided by the challenge organizers. We obtain the results shown in

Table 4 (using k=30). We notice that the results are very close to those reported for the split on the training dataset (cf. Table 1).

Table 4. Evaluation results over the final test set

Accuracy (category prediction)	0.962
Lenient NDCG@5 with linear decay (literal/resource type prediction)	0.777
Lenient NDCG@10 with linear decay (literal/resource type prediction)	0.762

4.4 Efficiency

Fine-tuning. We fine-tuned the models on Google Colab[5], a Jupyter notebook environment that runs in the cloud and offers access to GPUs. With a batch size of 32, number of epochs set to 3 and using an Nvidia Tesla K80 GPU, the time required for fine-tuning each classifier is: 49 mins and 25 secs for the resource question type classifier, using 26,259 questions, 27 mins and 51 secs for the question category classifier, using 14,814 questions, and 15 mins and 3 secs for the literal question type classifier, using 8,025 questions.

Execution. To classify a question into a category and predict its answer type, we execute the system locally on a machine with 2 cores and 8 GB of RAM, without using a GPU. While the system is running, it requires approximately 2.3 GB of RAM to load the 3 classifiers in memory.

This means that the proposed approach has low main memory requirements.

Moreover, this memory footprint can be further reduced if we use a smaller and lighter language model, such as DistilBERT [10], while sacrificing a small percentage of accuracy. The time required to classify a single question is less than a second (0.17 seconds on average), which is important for the application context that we have in mind (more below). To obtain the system output required to evaluate our system for the SMART challenge, we classified each one of the 4,381 questions provided in the test set sequentially. The process took 12 minutes and 24 seconds.

4.5 Application Context

We plan to integrate the proposed classification models in the Question Answering module of Elas4RDF [9,5], a keyword search system where users can input questions as queries and receive answers in real time according to various *perspectives*; one of them is the "QA perspective".

Screenshots of the system for the query "Greek philosopher from Athens who is credited as one of the founders of Western philosophy" are shown in Figure 2.

Moreover the classification model presented in this paper can be exploited also in the "Schema perspective", that shows the classes of the top-ranked triples (for allowing the user to refine as she wishes to), in order to promote (or just mark) the class that corresponds to the predicted answer type.

[5] https://colab.research.google.com/

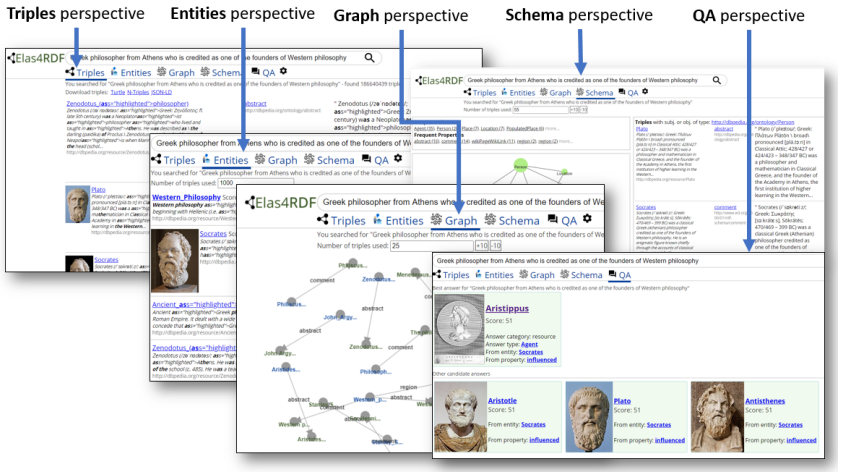

Fig. 2. Application Context: Elas4RDF

A demo of Elas4RDF over DBpedia [6] is publicly accessible at: `https://demos.isl.ics.forth.gr/elas4rdf/`.

5 Concluding Remarks

We have presented an approach for *semantic answer type prediction*, an important sub-task of QA which splits the problem into a two-stage pipeline of classification tasks: answer category prediction and answer literal/resource type prediction. We model the problem as a set of sequence classification tasks, each one making use of a fine-tuned BERT classifier. For the more fine-grained (and more challenging) problem of *answer resource type prediction* (since the classes can be hundreds or thousands), we have proposed the enrichment of the BERT model with a rewarding mechanism that considers the hierarchy of the ontology classes, favoring the more specific classes that are low in the class hierarchy. The evaluation results demonstrated the performance of the proposed method, achieving >96% accuracy in predicting the general answer category, >98% accuracy in predicting the literal type, and >77% NCDG@5 in ranking the predicted resource classes.

Our results showcase that it is feasible to achieve fine grained answer type prediction with very high precision and without expensive computations.

Issues that are worth further research include: methods for fine-tuning the parameter k that determines the minimum amount of training data needed to obtain a certain degree of performance, and evaluating the rewarding scheme in different datasets, e.g. in knowledge bases that have ontologies with more deep class hierarchies.

References

1. Balog, K., Neumayer, R.: Hierarchical target type identification for entity-oriented queries. In: Proceedings of the 21st ACM international conference on Information and knowledge management. pp. 2391–2394 (2012)
2. Deng, J., Krause, J., Berg, A.C., Fei-Fei, L.: Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3450–3457. IEEE (2012)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2018)
4. Dimitrakis, E., Sgontzos, K., Tzitzikas, Y.: A survey on question answering systems over linked data and documents. Journal of Intelligent Information Systems pp. 1–27 (2019)
5. Kadilierakis, G., Fafalios, P., Papadakos, P., Tzitzikas, Y.: Keyword Search over RDF using Document-centric Information Retrieval Systems. In: Extended Semantic Web Conference (ESWC'2020) (2020)
6. Kadilierakis, G., Nikas, C., Fafalios, P., Papadakos, P., Tzitzikas, Y.: Elas4RDF: Multi-perspective triple-centered keyword search over RDF using elasticsearch. Extended Semantic Web Conference (ESWC) – Posters & Demonstrations Track (2020)
7. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach (2019)
8. 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`
9. Nikas, C., Kadilierakis, G., Fafalios, P., Tzitzikas, Y.: Keyword Search over RDF: Is a Single Perspective Enough? Big Data and Cognitive Computing **4**(3), 22 (Aug 2020), `https://www.mdpi.com/2504-2289/4/3/22`
10. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter (2019)
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
12. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding (2019)