Semantic Answer Type Prediction using Dense Type Embeddings

G P Shrivatsa Bhargav, Dinesh Khandelwal, Dinesh Garg, and Saswati Dana

IBM Research, India
{gpshri27, dikhand1, garg.dinesh, sdana027}@in.ibm.com

Abstract. In this paper we describe our submission to the SMART 2021 Answer Type Prediction task. We propose a BERT based solution to the problem. The proposed approach relies on type embeddings obtained based on the type names. It allows our model to predict types at test time that were not seen during training. Analysis of the training dataset reveals the presence of noise in the labels. Therefore, we develop a label augmentation scheme to reduce the noise in the annotations and increase the quality of the training data. Our model trained on the de-noised data achieves 0.986 accuracy on the answer category prediction task and 0.825 and 0.790 NDCG@5 and NDCG@10 respectively on the test sets.

Keywords: Answer Type Prediction · Question Answering · Natural Language Processing

1 Introduction

Answer Type Prediction in SMART 2021 [4] comprises two sub-tasks. The first task is to predict the answer category of the given natural language question. The set of possible answer categories is resource, boolean, literal. The second task is to predict the answer type of the given question. The set of possible answer types depends on the answer category. If the category is resource, the types are a subset of the DBpedia or Wikidata ontology classes. If the cateogry is literal, the type could be either number, date or string. If the category is boolean, the type is always boolean. Table 1 shows examples from the SMART 2021 Answer Type Prediction dataset. The metric used to evaluate answer category prediction is accuracy score. Type prediction performance is measured using lenient NDCG@k with linear decay [1] (with k=5,10).

In this paper, we explore how transformers like BERT [2] can be used to effectively address the problem of Answer Type Prediction.

2 Dataset processing

In this paper, we focus on the SMART2021-AT DBpedia dataset. The dataset has 40621 samples for training and validation. An additional 10093 samples are held out for testing. We perform elementary data cleanup (for example, removing

Table 1. Examples from the SMART 2021 Answer Type Prediction dataset with DBpedia as the Knowledge Graph.

Question	Category	Type
Who are the gymnasts coached by Amanda Reddin?	resource	[dbo:Gymnast,dbo:Athlete, dbo:Person, dbo:Agent]
How many superpowers does wonder woman have?	literal	[number]
When did Margaret Mead marry Gregory Bateson?	literal	[data]
Is Azerbaijan a member of European Go Federation?	boolean	[boolean]

samples with null labels, removing duplicates, etc) and split the dataset into training and validation sets in the ratio 80 : 20. Table 2 summarizes the size of different sets.

Table 2. Dataset size

Set	Number of samples
Train	29356
Validation	7340
Test	9104

To establish a performance (accuracy and NDCG@k) upper-bound on this dataset, we performed an experiment where we evaluated[1] the training and validation sets against themselves. The goal was to check what accuracy and NDCG@k a system would obtain if its predictions exactly matched the gold annotations. On the training set, we found that the category prediction accuracy was 1 whereas, NDCG@5=0.8261 and NDCG@10=0.7529. This indicates that the gold label set is not complete. Such incompleteness/noise in the training data will directly impact the model's performance. Upon inspection, we found that some of the ancestor types (also known as super types, parent types) were missing in the gold types list. In some training samples, the types were not sorted according to the descending order of their depths in the ontology. We modified the gold type list of each training sample in the following ways - (i) We completed the type list by adding the missing ancestor types. (ii) We sorted the completed type list in descending order of their depth. We refer to these two steps collectively as label augmentation. Table 3 summarizes the impact of each of the above steps. The metrics in Table 3 also serve as a soft upper-bound on the performance of any model trained on this dataset. We train our models on the modified training set (+ ancestor types + sorting) but we validate on the unmodified validation set.

[1] github.com/smart-task/smart-dataset/blob/master/evaluation/dbpedia/evaluate.py

Table 3. The impact of each gold label augmentation operation.

Data	**Category accuracy**	**NDCG@5**	**NDCG@10**
Train	1.0	0.826	0.753
+ sorting	1.0	0.846	0.768
+ ancestor types + sorting	1.0	0.892	0.808
Validation	1.0	0.823	0.748
+ sorting	1.0	0.842	0.804
+ ancestor types + sorting	1.0	0.888	0.804

The metric NDCG@k is sensitive to the ordering of the predicted types. The evaluation script expects the predicted types to be sorted from the finest to the coarsest (i.e, decreasing order of depth in the ontology). But in several samples, there are multiple types of the same depth. In such cases, their ordering in the predicted type list can be arbitrary. This phenomenon could be the reason why the train and validation NDCG@k are not 1.

3 Proposed Approach

In this section we will describe our proposed approach to solve the Answer Type Prediction Task.

3.1 Problem Reformulation

To simplify the modelling task, we work on an equivalent reformulation of the problem. The reformulated problem can be stated as follows - Given a natural language question, the first task is to predict the answer category from the set of labels $\mathcal{C} = \{resource, number, date, string, boolean\}$. If the predicted category is resource, then the second task is to rank the set of DBpedia types $\mathcal{T} = \{t_1, t_2, \ldots\}$ from most relevant to least relevant. We train the model on the reformulated task, transform its predictions and report the metrics on the original task.

3.2 Question Encoding

We embed the given natural language question $Q_i =< q_{i1}, q_{i2}, \ldots >$ (where q_{ij} are the tokens) into vector space and use this embedding to predict the categories and rank the types. We leverage BERT to obtain the question embeddings as follows: (i) We surround the question tokens with special tokens [CLS], [C] and [SEP] to obtain a sequence of the form <[CLS][C]$q_{i1}, q_{i2}, \ldots$[SEP]> (ii) The above sequence is passed through BERT to obtain a sequence of vectors $< v_{[CLS]}, v_{[C]}, v_{q_{i1}}, v_{q_{i2}}, \ldots >$.

$v_{[CLS]}$ is used for the purpose of ranking the set of types $\mathcal{T}$ whereas $v_{[C]}$ is used to predict the answer category from the set of labels $\mathcal{C}$.

3.3 Category Prediction

The vector $v_{[C]}$ is passed through a fully connected layer (with parameters W and b) followed by a softmax layer in order to obtain the probability for each of the categories in $\mathcal{C}$. To tune the parameters for this task, we use the cross entropy loss.

$$\mathcal{L}_{\mathcal{C}} = - \sum_{Q_i \in \mathcal{D}_{train}} \log p(c_i^* | Q_i)$$

where $\mathcal{D}_{train}$ is the training set and c_i^* is the true category for the question Q_i.

3.4 Type Embedding

In order to rank the types $\mathcal{T}$ in the next step, we require a vector embedding e_i for every type $t_i \in \mathcal{T}$. We use BERT to obtain the initial embeddings of the types. To do so, we first normalize the names of each type t_i to obtain English phrases. For example, the type "dbo:GovernmentalAdministrativeRegion" is transformed to "Governmental Administrative Region". The normalized type names are passed through BERT and the resulting output corresponding to the [CLS] vector is used as the initial type embedding. Thus, we obtain an embedding matrix $E_{\mathcal{T}} \in \mathbb{R}^{|\mathcal{T}| \times d}$, where d is the embedding dimension. Each row e_i of $E_{\mathcal{T}}$ corresponds to the embedding of the type t_i. By creating the type embeddings this way, we ensure that we will have a good representation of all the types even though they may be unseen during training.

3.5 Type Ranking

For a given question Q_i, we predict the probability $p(t_j = 1|Q_i)$ for each type t_j and then rank the types from the most probable to the least. $p(t_j = 1|Q_i)$ is computed as follows:

$$p(t_j = 1|Q_i) = \frac{1}{1 + \exp(-v_{[CLS]}^T \cdot e_{t_j})}$$

We train the model to maximize the probabilities of the gold types using the following loss:

$$\mathcal{L}_{\mathcal{T}} = - \sum_{Q_i \in \mathcal{D}_{train}} \mathbb{1}_{res}(Q_i) \Bigg(\sum_{t_j \in \mathcal{T}} \lambda_1 \mathbb{1}_{Q_i}(t_j) \log p(t_j | Q_i) + \lambda_2 (1 - \mathbb{1}_{Q_i}(t_j))(1 - \log p(t_j | Q_i)) \Bigg)$$

where, $\mathcal{D}_{train}$ is the training dataset, $\mathbb{1}_{res}(Q_i)$ indicates whether the category of question Q_i is "resource" or not, $\mathbb{1}_{Q_i}(t_j)$ indicates whether the type t_j is a valid answer type for the question Q_i, λ_1 and λ_2 are scalar hyperparameters. In every training sample, the number of negative types is far greater than the number of positive types. Due to this imbalance, the model will learn to predict zero probability for all the types. To remedy this, we use the hyperparameters λ_1 and λ_2 to balance the losses corresponding to the positive and negative types.

3.6 Training

We jointly optimize $\mathcal{L}_{\mathcal{C}}$ and $\mathcal{L}_{\mathcal{T}}$ in a multi task fashion. The multi-task learning objective function is:

$$\mathcal{L} = \mathcal{L}_{\mathcal{C}} + \alpha \mathcal{L}_{\mathcal{T}}$$

where α is a scalar hyperparameter that controls the relative importance of $\mathcal{L}_{\mathcal{C}}$ and $\mathcal{L}_{\mathcal{T}}$. The parameters of the question encoder BERT, W, b and $E_{\mathcal{T}}$ are all updated during training.

3.7 Inference

At the inference time, we first run the answer category prediction. The type prediction depends on the predicted category. Table 4 illustrates how the output on the reformulated task is converted to the output on the original task.

Table 4. Inference strategy

Category prediction	Transformed Output
resource	category = resource, type = types sorted by $p(t_j \vert Q_i)$
number	category = literal, type = number
date	category = literal, type = date
string	category = literal, type = string
boolean	category = boolean, type = boolean

4 Experiments

4.1 Implementation Details

We implemented the proposed approach in Pytorch. The source code and the trained models have been released on Github[2]. BERT-Base is used in all our experiments. We use ADAM [3] to optimize the objective function. Validation set performance is used for early stopping and model selection. Table 5 summarizes the hyperparameters and their values. Our set of DBpedia types contains 791 elements. We do not restrict the type set to only those seen during training. All the models were trained on a single Nvidia K80 GPU. Each epoch of training required approximately 35 minutes.

4.2 Results

Table 6 shows the performance of our models on the validation and test sets. First, we train BERT on the training dataset without any label augmentation. The model achieves near-perfect accuracy on the answer category prediction

[2] https://github.com/IBM/answer-type-prediction

Table 5. Hyperparameters

Hyperparameter	**Value**
Learning rate	$3e^{-5}$
Hidden dimension d	768
Batch size	8
α	1.0
Max question length	64
Max number of epochs	6
λ_1	1/num. positive types in the batch
λ_2	1/num. negative types in the batch

task. Next, we train BERT on the training dataset after label augmentation. Label augmentation gives a boost of 1% in NDCG@5 and 1.4% in NDCG@10. The performance of this model is only 6.8% and 1.9% short of the soft upperbound established in Table 3

Table 6. Results on the training and validation sets.

		BERT	**BERT + label augmentation**
Val set	Cat. acc.	0.986	0.986
	NDCG@5	0.810	0.820
	NDCG@10	0.771	0.785
Test set	Cat. acc.	-	0.985
	NDCG@5	-	0.825
	NDCG@10	-	0.790

4.3 Analysis

We performed analysis on the validation set to understand the strengths and weaknesses of our model (BERT + label augmentation).

Table 7 shows the confusion matrix on the answer category prediction task. Table 8 shows randomly sampled examples of each kind of mistake made by the classifier. In all cases where the model predicts boolean instead of literal and boolean instead of resource, the model is correct and the gold annotation in incorrect. The confusion between resource and literal is prominent but hard to resolve. The answer category (resource or literal) is completely dependent on the knowledge base. It is impossible to decide resource vs literal without finding the answer to the question first.

To study the errors made by the answer type predictor, we randomly sample the validation set examples on which the model's NDCG@5 is less than 0.2. We show some of these examples in table 9. In examples 1 and 3, we see that the model's ranking is correct but the gold annotations are incorrect. Example 2 however, is a mistake by our model and the reason is unclear.

Table 7. Confusion matrix of the answer category classifier

		Prediction		
		resource	literal	boolean
Gold	resource	0.9927	0.0064	0.0008
	literal	0.0675	0.9276	0.0048
	boolean	0.0023	0	0.9976

Table 8. Examples of errors made by the answer category classifier

Gold	**Prediction**	**Questions**
resource	literal	How many seats are in prefectural assembly? What is the demised place of Leo III What is the symbol for pi? what year did tim duncan enter the nba
resource	boolean	Did Barbados have a diplomatic relationship with Nigeria in the past? Was Natalia Molchanova born in the Bashkir Autonomous Soviet Socialist Republic? is ANZUS a signatory? Was Gustav Mahler‘s birth place located in the administrative territorial entity of Kalista ?
literal	resource	In what country is Mikhail Fridman a citizen? What's the original language for Close Encounters of the Third Kind? Who sponsors the FC Bayern Munich?
literal	boolean	Did Masaccio die before the statement of Gregorian Is the language of Neptune, Czech? Is Thom Enriquez part of the film crew for Beauty and the Beast? Is there an audio recording of Charles Duke?
boolean	resource	What is the geography of the planet, Mars?

Table 9. Examples of errors made my the type ranking module. The examples shown in this table are randomly sampled from those questions whose NDCG@5 is less than 0.2.

Question	Gold types	Top 5 Predicted types
What country signed the North Atlantic Treaty that has a spoken language of Portuguese?	[‘dbo:Person’, ‘dbo:Agent’]	[‘dbo:Country’, ‘dbo:Location’, ‘dbo:PopulatedPlace’, ‘dbo:Place’, ‘dbo:State’]
What year doug williams won the super bowl	[‘dbo:FootballLeagueSeason’, ‘dbo:SoccerClub’]	[‘dbo:Software’, ‘dbo:Work’, ‘dbo:VideoGame’, ‘dbo:TelevisionShow’, ‘dbo:FootballLeagueSeason’]
Where is the headquarters of the car manufacturer Lyon	[‘dbo:Company’, ‘dbo:Organisation’, ‘dbo:Agent’]	[‘dbo:Location’, ‘dbo:Place’, ‘dbo:Settlement’, ‘dbo:PopulatedPlace’, ‘dbo:City’]

5 Conclusions

In this paper we explored how BERT can be used to address the problem of answer type prediction. We first established a soft upper bound on the performance of models that are trained on the SMART 2021 dataset. We developed a label augmentation scheme to de-noise the gold annotations and hence improve the model. Our model achieves 0.986 accuracy on the answer category prediction task and 0.825 and 0.790 NDCG@5 and NDCG@10 respectively on the test set. On the validation set, the performance of our model (after label augmentation) is only 0.068 and 0.019 short of the soft upper-bound established in Table 3. We also present error analysis that shows the nature of errors made by our model and the noise in the dataset.

References

1. Balog, K., Neumayer, R.: Hierarchical target type identification for entity-oriented queries. p. 2391–2394. CIKM ’12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2396761.2398648, https://doi.org/10.1145/2396761.2398648
2. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT. pp. 4171–4186 (2019)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980

4. Mihindukulasooriya, N., Dubey, M., Gliozzo, A., Lehmann, J., Ngonga Ngomo, A.C., Usbeck, R., Rossiello, G., Kumar, U.: Semantic Answer Type and Relation Prediction Task (SMART 2021). arXiv (2022)