Team-009 - Semantic Answer Type Prediction

Rihab Alzurkani
rh.al-zurkani@stud.uis.no

Ziad Razani
z.razani@stud.uis.no

Nathan Chavas
n.chavas@stud.uis.no

Daniel Gundersen
d.gundersen@stud.uis.no

ABSTRACT

The Semantic Answer Type prediction task (SMART) was a portion of ISWC 2020 tasks. The type of both question type and answer prediction has been the main process in knowledge base question answering systems giving the important aspects that will aid to produce the most accurate queries or rank the answer candidates.

1 INTRODUCTION

Semantic answer type prediction is a task where one or several types are designated to a question. The task is often used as a sub-part of a larger question answering system, where the type prediction is used by the system to provide more detailed and accurate answers to a query. Answer type prediction can be done using coarse-grained types, where type classification is done from a relatively small set of classes, or a fine-grained types, where the question is assigned a type from larger ontologies such as DBepdia or Wikidata [5].

1.1 Related Works

Balog and Neumayer [1] and Garigliotti et al. [2] both applied an entity-centric and a type-centric approach to find fine grained types for questions. The entity-centric and the type-centric approaches considers each type as a combination of its entities. The type-centric approach merges all its entities into one document and ranks this using a retrieval model. The entity-centric approach scores each entity separately using a retrieval model and then aggregates the scores of the entities into the score of its type. Balog and Neumayer [1] identified the entity-centric approach as returning more general types, moreover the performance are improved.

2 PROBLEM STATEMENT

In the The Semantic Answer type prediction (SMART) task a natural language question is provided and a question category and type should be predicted based on the question. There are three categories:

(1) Boolean
(2) Literal
(3) Resource

The type of each question depends on the category of the question. A question of category *Boolean* can only have the type *Boolean*. A question of the category *Literal* can have one of the types: *number*, *date* or *string*. Finally, a question of type *Resource* can have one or several classes from either the DBpedia ontology or the Wikidata ontology, depending on which dataset is used [5].

The task provides two dataset. One for the DBpedia ontology and one for the Wikidata ontology. Each dataset contains a list of questions. Each question contains a question id, the natural language question, the true question category and a ranked list of true question types. A summary of the datasets can be seen in Tables 1 and 2. In this paper we will focus on the DBpedia dataset.

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`	`["date"]`
Is Azerbaijan a member of European Go Federation?	`boolean`	`["boolean"]`

Figure 1: Example questions with relevant answer sets and categories the resources are established on DBpedia types [5].

Table 1: Distribution of data between training and test data in datasets [5]

	Train Data	Test Data
DBpedia	17,571	4,393
Wikidata	18,251	4,571

Table 2: Distribution of data between the different categories in datasets [5]

	boolean	literal	resource
DBpedia	2,799	5,188	9,584
Wikidata	2,139	4,429	11,683

3 BASELINE METHOD

The problem is divided into two parts [1]. In the first part a machine learning technique will be applied to identify the category of the question, as well as the type of questions belonging to the *Literal* category. In the second part, an information retrieval method is applied to all questions identified as belonging to the category "Resource" in part 1 to find the correct types. In part 1 SVM will be used as the machine learning model, and in part 2 an entity centric approach will be used. We will therefore call this method the *SVM+EC* approach.

3.1 Extended category classification

The category classification, within this part, predicate the high-level category of the answer type by a supervised classifier. We first proceed by creating the following classes: Boolean, Number,

[1]Github Repository for the Code can be found at: https://github.com/erthbison/DAT640-Team-009-Semantic-Answer-Type-Prediction-2020

Date, String, Resource. These answer types are the classes that will be used for the machine learning model after unpacking them from the training dataset using some data processing functions. We then train the SVM model, which we used scikit-learn's SVM implementation for, after vectorizing the questions (maps each vectorized question to a unique class from the classes mentioned above)[4]. After training, the SVM model is able to predict if an answer's type falls within the mentioned classes. If the result is a Boolean or one of the Literal types, our classification has ended since we successfully predicted both the category and type of the answer. If the result is "Resource", then we proceed to the second part of the process to identify the types.

The functions used in this part of the solution were mainly data manipulation functions, unpacking the answer types and transforming the couple (Category, Type) into one single class that can be trained on by the machine learning model. This was possible since all the information regarding whether the answer would be a number or a date or a boolean is contained in the question, unlike the "Resource" type which has too many classes to be solved through a machine learning model and would more efficiently be solved through an entity centric approach.

3.2 Resource type classification

The resource type classification problem can be cast as a ranking problem where, given a query, q, and a list of types, τ, a relevance score, $score(q, y)$, should be calculated for all $y \in \tau$. A list of relevant types can then be selected based on the relevance score.

In the entity-centric approach, a retrieval model is first used to calculate the score of all entities. Then the top k entities is selected and used to calculate the score of the type. The score of a type, y, for a query, q, can then be presented as equation 1, where $score(e, q)$ is a retrieval function and $w(e, y)$ is a weight function.

$$score_{ec}(y, q) = \sum_{e \in \epsilon_q(k)} score(e, q) \cdot w(e, y) \quad (1)$$

Because of its good trade-off between efficiency and performance, BM25 was used as retrieval model with the default parameters of $k_1 = 1.2$ and $b = 0.8$. For simplicity a weight function that uniformly weights all entities, e, of the type, y, is used. Furthermore, $k = 150$ was used, since through experimentation on the training set it showed a good trade-off between performance and efficiency.

The entity representation was created by using the combination of the "article category", "disambiguations", "long abstracts" and "redirects" fields of the DBpedia representation of the entity. These fields were combined into a document for each entity and preprocessed using a scikit-learn hashing vectorizer with stop-word removal enabled[4].

The scoring function was used for generating a ranked list of types. The top scoring type and all types directly above it in the hierarchy where selected as the type for the question. The top-level type, 'Thing', was ignored since it would be present in all questions.

4 ADVANCED METHOD

4.1 Approach

Our approach for the advanced method was focused on improving the resource type prediction since the first part revolving around category prediction (as well as literal type prediction) performed fairly well in the baseline method. Using the approach described here [3], we will be using a fine-tuned BERT model through Hugging-face transformers [2] implementation:

(1) Using BERT's understanding of the language to construct a correspondance between the questions and a numerical vector.
(2) The previous output is then injected into an additional layer that would predict the output class.

4.2 Implementation

Our approach is implemented through BertForSequenceClassification, which encapsulates the previous approach into one model, see the code in this article on how to Fine-Tune a BERT model for binary classification.[3] However, the large number of classes coupled with the small number of samples for many of these classes makes it unpractical to train the last layer on all these resource types. We therefore follow the method in this paper [3] where we only trained the model to predict a certain subset of classes that fulfill the following condition:

$$C = \{class | samples(class) > k\}$$

In our case, we chose the threshold value to be 20. After predicting the type of the resource, we search for its parent classes and formulate that as our prediction for the resource type.

5 EVALUATION METRICS AND SOFTWARE

For most language query in the test set, the participating systems has expectation for giving two predictions: answer category and answer type, the following is the same format as training data. The answer category would be either "resource", "literal" or "boolean". If the answer category is "resource", the answer type have to be an ontology class (DBpedia or Wikidata, related to the dataset). The systems may predict a ranked list of classes from the ontology. If the answer category is "literal", the answer type can be either "number", "date" or "string". We implement NDCG@k (the lenient metric)[1].

6 RESULTS

The methods are evaluated using three metrics. Since category assignment is a classification problem accuracy will be used as metric. For the type assignment NDCG@5 and NDCG@10 with linear decay is used as metric. This is because the types are organized in a hierarchy, and NDCG@k provides a lenient score where a model selecting types closer on the hierarchy to the true value will score better than a model selecting types on a completely different branch[1]. The linear decay can be calculated using the following formula, where $d(t, t_q)$ is the distance between the true type and

[2] https://huggingface.co/
[3] Fine-Tuning BERT for Text Classification

the predicted type, and h is the depth of the hierarchy.

$$G(t) = 1 - \frac{d(t, t_q)}{h} \tag{2}$$

For the types that are not a part of the type hierarchy, i.e. the Literal types, the NDCG@k is set to 0 if the wrong type was selected and 1 if the correct type was selected.

6.1 Baseline Model Results

Table 3: Baseline Model Results

	Accuracy	NDCG@5	NDCG@10
SVM+EC	0.927	0.512	0.495

6.2 Advanced Model Results

Using BERT instead of the entity-centric model with no depth rewarding for resource classes gives a clear 10% improvement over the baseline model when it comes to both NDCG@5 and NDCG@10 scores.

Table 4: Advanced Model Results

	Accuracy	NDCG@5	NDCG@10
SVM+BERT	0.927	0.625	0.594

7 DISCUSSION AND CONCLUSIONS

When it comes to the Advanced Model, it is clear that the gain in NDCG score comes from BERT's understanding of the language, which further facilitates associating a resource type to the question. Therefore it becomes easier to locate which parts of the question will define the category of the answer and which parts will help predict the its resource type. Despite the great number of classes, the method described our approach helps find a compromise between accuracy specificity since leaving out the more specific classes with a very small sample space does not greatly impact the score.

REFERENCES

[1] Kristian Balog and Robert Neumayer. 2012. Hierarchical Target Type Identification for Entity-oriented Queries. *21st ACM Conference on Information and Knowledge Management (CIKM 2012), pages 2391-2394, October 2012.* (Oct. 2012), 2391–2394.

[2] Darío Garigliotti, Faegheh Hasibi, and Krisztian Balog. 2017. Target Type Identification for Entity-Bearing Queries. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, Shinjuku Tokyo Japan, 845–848. https://doi.org/10.1145/3077136.3080659

[3] Christos Nikas, Pavlos Fafalios, and Yannis Tzitzikas. [n.d.]. Two-stage Semantic Answer Type Prediction for Question Answering using BERT and Class-Specificity Rewarding. ([n. d.]).

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[5] Krisztian Balog Vinay Setty. 2020. *SeMantic AnsweR Type prediction task (SMART) at ISWC 2020 Semantic Web Challenge.* Technical Report arXiv:2012.00555. arXiv. http://arxiv.org/abs/2012.00555 arXiv:2012.00555 [cs] type: article.

A DIVISION OF WORK DURING THE PROJECT

Daniel Gundersen: Wrote/Edited section 1 and 2. Implemented baseline part 2. Wrote about baseline part 2. Wrote part of Results section.

Rihab work on the proposal, problem statement and edited on part 2,Evaluation metric

Ziad Razani: Implemented the first part in the Baseline Method (Data manipulation and training of the SVM classifier), implemented, trained and evaluated the BERT classification model for the resource type prediction in the Advanced Method.