CitySAT: a System for the Semantic Answer Type Prediction Task*

Chaeyoon Kim[1] and Ernesto Jiménez-Ruiz[1,2]

[1] City, University of London, London
[2] SIRIUS, University of Oslo, Norway

Abstract. This paper describes the CitySAT system that we developed for the DBpedia Answer Type (AT) prediction task of the SMART 2021 challenge. The challenge can be interpreted as a multi-class classification task that takes natural language questions and returns pairs of the predicted answer category and types. For training, we merged the SMART 2021 DBpedia dataset with the 2020 dataset given for the previous year's AT task. In this study, three local Machine Learning (ML) models are deployed to cover the three different types of task and question (category prediction, literal type prediction and resource type prediction). The best model obtains a 98.36% accuracy for the category prediction using a Logistic Regression (LR) classifier. Similarly, another LR model results in 97.90% accuracy for the literal type prediction task. Lastly we also built a Multi-Layer Perceptron (MLP) model to deal with several ontology classes (∼760 classes for DBpedia) in the resource type prediction task. The best MLP model achieves 79.34% on the merged training dataset. The final system output obtained a 98.4% accuracy, 84.2% NDCG@5, and 85.4% NDCG@10 on the (official) test dataset.

Keywords: Semantic answer type prediction · SMART DBpedia challenge · multi-class classification

1 Introduction

In computer science, Answer Type prediction (AT) is a research domain providing with simplified tasks of the Question Answering (QA) discipline. It inherits the QA task missions to understand the meaning of natural language questions but identifies their answer types instead of retrieving the most relevant information among the answer candidates. Due to the number of classes of the categorical answer type candidates, AT task can be interpreted as a multi-class text classification task upon pre-defined classes.

The SeMantic Answer type and Relation prediction Task (SMART),[1] organized by the International Semantic Web Conference (ISWC), expands the AT

[1] https://smart-task.github.io/

Table 1. The size of datasets: SMART 2021 challenge increased a 231% w.r.t. 2021.

SMART 2020			**SMART 2021**			**Merged**
Train	Test	Total	Train	Test	Total	Train
17,571	4,369	21,940	36,670	9,104	45,774	39,556

challenge to a more complicated structure of knowledge base data which consists of the answer category and type in two hierarchical levels. They provide two versions of large-scale dataset for the AT task; one edition using DBpedia ontology (∼760 classes) and the other edition using the Wikidata taxonomy (∼50K classes). This paper demonstrates our participation in the SMART 2021 AT task [6], concentrating only on the DBpedia dataset, that takes a set of natural language questions alongside with the corresponding answer category and ontological type classes of DBpedia to predict a suitable type for new questions. The reason of the dataset choice is because of the clearer separation between ontology (i.e., terminology) and data (i.e., assertions) in DBpedia.

The CitySAT system firstly verifies the accuracy score of the answer category prediction and additionally uses the lenient Normalized Discounted Cumulative Gain (NDCG) at each 5 and 10 answer type values to compete with comparable systems. The best optimization confirms 98.4% accuracy, 84.2% NDCG@5, and 85.4% NDCG@10. As NDCG evaluation metric follows a linear decay, NDCG@10 is highly valued and our result proves to the beneficiaries that the more predefined knowledge the better the system predicts.

The rest of the paper is structured as follows. Section 2 (Context) surveys the used materials and related work. Section 3 (Methods) presents a detailed view of the entire research progress from data loading to evaluation with justifications for each of the steps. Section 4 (Results) summarizes the respective high-valued experiments. Section 5 (Discussion and Future work) examines what is a meaningful correlation between our research motivation and results that conclude the next stage of studies. The code for reproducing the experimental results of this study is publicly available at `https://github.com/chaeyoonyunakim/smart-2021-AT`.

2 Context: Materials and related work

SMART 2020 [7] published eight AT systems on the DBpedia dataset that our research aimed at comparing as reference models for SMART 2021.

To start with training datasets, the previous work (e.g., [2] and [10]) supports the positive effect of larger training datasets and thereby this study also tried to maximize the volume of the training data by merging every relevant resource. Table 1 gives an overview of our merged training dataset size compared to SMART 2020 and 2021 challenge dataset. Originally the prior DBpedia edition contained 21,940 questions but the later edition has increased the data size up to 45,774. At the end, our study settled down on the merged training data with 39,556 set of questions and model answers. SMART datasets are designed to provide a single answer category either “boolean”, “literal”, or “resource”. They

Table 2. SMART 2020 Leaderboard - DBpedia AT task.

System	Accuracy	NDCG@5	NDCG@10
Setty *et al.*[11]	0.98	0.80	0.79
Nikas *et al.*[8]	0.96	0.78	0.76
Perevalov *et al.*[10]	0.98	0.76	0.73
Kertkeidkachorn *et al.*[5]	0.96	0.75	0.72
Ammar *et al.*[1]	0.94	0.62	0.61
Vallurupalli *et al.*[14]	0.88	0.54	0.52
Steinmetz *et al.*[12]	0.74	0.54	0.52
Bill *et al.*[2]	0.79	0.31	0.30

assign the "boolean" category as "boolean", "literal" category into an answer type either "number", "date", or "string". For the "resource" category, DBpedia ontology classes are placed in the answer type.

Another one of the most important observations over the previous work is that they presented a very high accuracy for their answer category classifier on the upper level of the hierarchical data structure. For the 2020 performance results, Table 2 shows that top ranked outputs even exceeded over 90% accuracy. It led an initial decision that our experiments can be conducted on traditional Central Processing Units (CPUs). Due to the increased data volume, however, it is worth to define how to selectively adapt the referencing models in this study. Hence, some initial data analysis on lower level hierarchy (i.e., targets on the answer type) is following in next section.

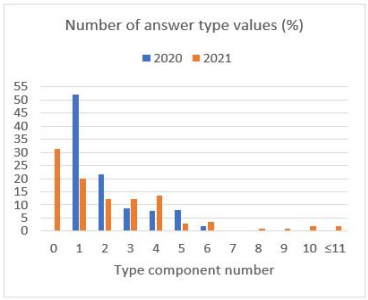
2.1 Initial data analysis

Regarding the shape of the bottom level data for answer type, Figure 1 gives details how many ontological classes are located in the resource type. Both 2020 and 2021 DBpedia editions are mostly consisted of equal or less than 6 answer types, but the distribution of type value numbers is more positively skewed in the 2020 edition. Therefore, the previous systems considered the first five (e.g., [10]) or six values (e.g., [2]) for their local classifier to predict the resource type. Some questions of 2021 edition are shown to have more than 10 ontology classes: 666 questions are assigned between 11 and 30 types, and 9 questions have up to 627 types which are all going to be excluded from the NDCG evaluation matrix as it measures up to 10 classes. In this study, we would try to train by every number of type values up to 10 so that it can be compared with last year performance.

To understand the context of the answer type, Figure 2 illustrates an example question and how its ontological classes are lined up in the given dataset. Among the classes, there is an exception (dbo:Location) which is defined as an entity in DBpedia but it is considered as a class having an equal level with dbo:Place specifically in SMART 2020 and SMART 2021 shared tasks. As explained in [10], the 2020 edition is sorted to have the most general class at the end. In spite of that, the 2021 edition is in the mixed order of classes. For example, the answer "type" in Figure 2 can be rearranged to ["dbo:River", "dbo:Stream", $\cdots$, "dbo:NaturalPlace", "dbo:Place", "dbo:Location"] if the same principle as

in the 2020 edition is applied. Because of the computational logic in NDCG, this study has an initial hypothesis that the order of the classes in the answer type does not affect the later normalised distance thereby there is no need to apply extra logic to sort the term orders.

2020 edition			
Number of Type	Count	%	Acc %
0	16	0.09	0.09
1	9,161	52.14	52.23
2	3,776	21.49	73.72
3	1,500	8.54	82.25
4	1,345	7.65	89.91
5	1,416	8.06	97.97
6	348	1.98	99.95
7	5	0.03	99.98
8	4	0.02	99.99
9	1	0.01	100.00
	17,571	100%	100%



2021 edition			
Number of Type	Count	%	Acc %
1	11,515	31.40	31.40
2	7,324	19.97	51.37
3	4,506	12.29	63.66
4	4,488	12.24	75.90
5	4,951	13.50	89.40
6	1,071	2.92	92.32
7	1,330	3.63	95.95
8	147	0.40	96.35
9	302	0.82	97.17
10	361	0.98	98.16
11~30	666	1.83	99.98
31~627	9	0.02	100.00
	36,670	100%	100%

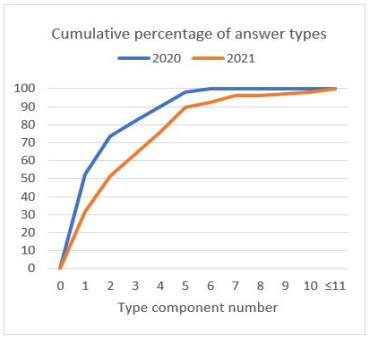


Fig. 1. Comparison table and charts between two years of SMART AT dataset for resource answer type.

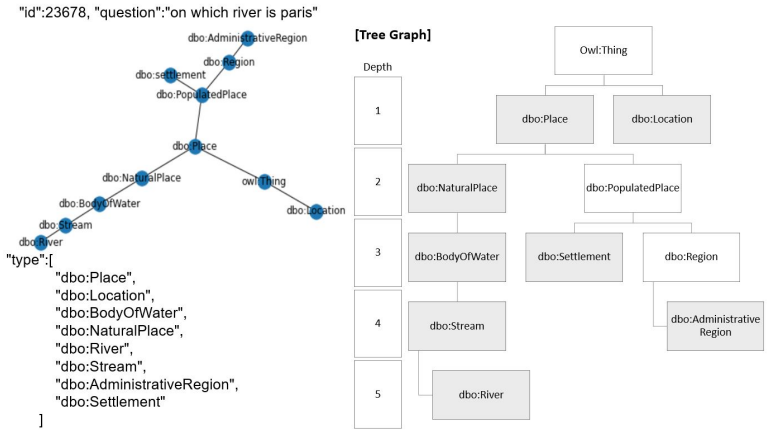


Fig. 2. Graphical representation of a sample question and resource answer type.

	id	question	category	type
0	1177	Was Jacqueline Kennedy Onassis a follower of M...	boolean	[boolean]
1	14427	What is the name of the opera based on Twelfth...	resource	[dbo:Opera, dbo:MusicalWork, dbo:Work]
2	16615	When did Lena Horne receive the Grammy Award f...	literal	[date]
3	23480	Do Prince Harry and Prince William have the sa...	boolean	[boolean]
4	3681	What is the subsidiary company working for Leo...	resource	[dbo:EducationalInstitution, dbo:Organisation,...

Fig. 3. Tabular representation of the SMART 2021 training dataset

3 Methods

3.1 Data loading and manipulation

In SMART AT task, the training dataset and test dataset share the same JSON format (see example below). Comparing the DBpedia datasets from the two SMART editions, at first glance, we found the "id" attribute changed its value from "dbpedia_1" to "1" and this has been taken into account when merging the datasets according to the arrangement in the 2021 edition.

```
{
"id": "1",
"question": "Who are the gymnasts coached by Amanda Reddin?",
"category": "resource",
"type": ["dbo:Gymnast", "dbo:Athlete", "dbo:Person", "dbo:Agent"]
}
```

This study uses Python pandas library [13] for data manipulation. Firstly, it loads each year's dataset into a tabular representation as shown in Figure 3. Through the data cleaning, both year's datasets match a single answer category (either boolean, literal, or resource) and at least one answer type per each question. Lastly, the duplicates are removed when a question and corresponding answer are the same, however, the diverse answers for a questions are kept. Figure 4 gives example cases of our dataset pre-processing.

To construct a robust data structure, feature engineering is necessary to drop the missing value observations such as zero number of type values in resource category and invalid values such as "n/a" in question. Due to the NDCG evaluation at the 10th answer type, we justify outliers as having more than 10 components in the answer type. In the case of not having the counter number of components, an indicator of missing value has to be marked since the evaluation matrix will return infinite distance of type path that means no relevance between the output (i.e., prediction_types) and the ground truth (i.e., gold_types).

As the ground truth was not open for the 2021 challenge, this study used 20% of the merged training dataset for validation to check the performance of the system. For submission, the test dataset (9,104 questions) is predicted by the best model trained with the entire training dataset (39,556 questions and answers).

(a)

question	category	type_str
which record did pole vault hold and which is the height?	resource	dbo:Person,dbo:Agent
What and where is the record for the pole vault held?	resource	dbo:Person,dbo:Agent
Name an actor.	resource	dbo:Person,dbo:Agent
Charles the Bald position is what and he got his position after which person?	resource	dbo:Media
When did chairperson of Communist Party of China and followed by?	resource	dbo:Person,dbo:Agent
Who is the chairperson of the Communist Party of China and who are his followers?	resource	dbo:Person,dbo:Agent
what is a drama film?	resource	dbo:Film,dbo:Work
What is it?	resource	dbo:Person,dbo:Agent
What position was held by George VI and when did he begin in that position?	resource	dbo:Profession,dbo:PersonFunction
What is it?	resource	dbo:City,dbo:Settlement,dbo:PopulatedPlace,dbo:Place,dbo:Location

(b)

	id	question	category	type
94	116	What is it?	resource	[dbo:Person, dbo:Agent]
937	1173	What is it?	resource	[dbo:Person, dbo:Agent]
3295	4120	What is it?	resource	[dbo:EthnicGroup]
5058	6342	What is it?	resource	[dbo:Taxon, dbo:TopicalConcept]
5073	6357	What is it?	resource	[dbo:City, dbo:Settlement, dbo:PopulatedPlace,...
5735	7173	What is it?	resource	[dbo:City, dbo:Settlement, dbo:PopulatedPlace,...
9782	12206	What is it?	resource	[dbo:Country, dbo:State, dbo:PopulatedPlace, d...
11872	14798	What is it?	resource	[dbo:City, dbo:Settlement, dbo:PopulatedPlace,...
12993	16164	What is it?	resource	[dbo:Person, dbo:Agent]

(c)

	id	question	category	type
22190	27690	what is a drama film?	resource	[dbo:Film, dbo:Work]
24504	30565	what is a drama film?	resource	[dbo:Film, dbo:Work]
31764	39656	what is a drama film?	resource	[dbo:Film, dbo:Work]
31794	39694	what is a drama film?	resource	[dbo:Film, dbo:Work]
32623	40754	what is a drama film?	resource	[dbo:Film, dbo:Work]
33534	41868	what is a drama film?	resource	[dbo:Film, dbo:Work]
40111	50078	what is a drama film?	resource	[dbo:Film, dbo:Work]

Fig. 4. (a) Top 10 list of duplicates when datasets are merged, (b) An example case of keeping: a single question has multiple answers–keep variants, (c) An example of removing: duplicated question and answer pairs.

3.2 Data derivation: Extract, Transform, Load (ETL)

This study uses the NLTK library [3] to parse natural language question inputs into tokenized words, and also normalise text with *PorterStemmer* (stemming) and *WordNetLemmatizer* (lemmatization) after stop-words removal.

Initially, this study was specifically interested in dealing with Wh-terms (Who, What, When, Where, Which, Whom, Whose, Why, and include How) questions which accounts for 84.4% of the training questions. So, we customized stop words dictionaries to exclude the Wh-terms. Unfortunately, however, the new stop-words removal works less efficiently in despite of the initial interests. Hence, this study keeps the original NLTK stop words for further analysis.

Additionally, this study explores term-frequency (TF) and inverse term-frequency (TF-IDF) in text feature extraction using *CountVectorizer* and *TFIDFVectorizer* from the scikit-learn library [9] from lessons of [11]. Empirical evidence concludes that applying the combination of stemming and TF for the first 10,000 unigram or bigrams is the best suitable for the SMART AT task.

Categorical targets (answer category and type) are mapped into numerical labels. In particular, we distribute the answer type by wish-number of training target classes and encode "missing" if nothing exists in the location. For example, Figure 5 illustrates a sample conceptual data when we select the 10 values of the answer type across all categories. The classifiers can be programmed to have the first and single value (i.e., *type1* column in Figure 5) if the category is either "boolean" or "literal". However, one or more values are taken when the

	type1	type2	type3	type4	type5	type6	type7	type8	type9	type10
0	boolean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	dbo:Opera	dbo:MusicalWork	dbo:Work	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	date	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Fig. 5. Conceptual table to refer each allocated location of values in type.

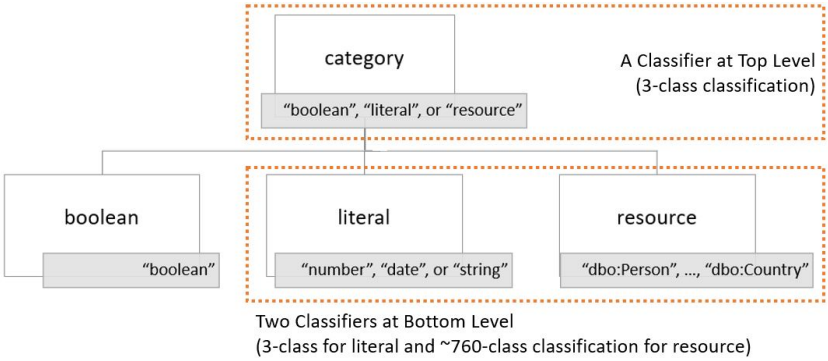


Fig. 6. System model design to perform hierarchical classification.

category is "resource". The order of values is mixed as stated in Figure 2, and our justification is to select the first five to ten values.

Furthermore, the selective number of features are mapped into a dictionary with keys for an indicator of the value's location. For example, by setting the argument type_no 11 (i.e., type1 to type10) in the reference code below, dictionary keys ["type1", ···, "type10"] and dictionary values [{"dbo:Opera": 0, ··· "dbo:RadioStation": 297}, ···, {"dbo:Politician": 0, ···, "dbo:Entomologist": 21}] will create 10 json files for {"type1": {"dbo:Opera": 0, ··· "dbo:RadioStation": 297}} to {"type10: {"dbo:Settlement": 0, ···, "dbo:Village": 21}}. The final accumulated dictionary of type_maps is {"type1": {"dbo:EducationalInstitution": 0, ···, "dbo:Holiday": 282}, "type2": {"dbo:MusicalWork": 0, ··· "dbo::Presenter": 171}, ···, "type10": {"dbo:Politician": 0, ···, "dbo:Entomologist": 21}}.

3.3 Multi-classification model design

To perform hierarchical classification, this study considers local classifiers per level as shown in Figure 6 which is widely used in the state-of-the-art (e.g., [8], [10], and [11]) because of the imbalanced number of classes among the classification group. Whilst both the category prediction and the "literal" type prediction are for each three unique classes, the "resource" type prediction is towards ∼760 unique classes.

To decide a suitable classifier for each level, we implemented a small sample batch of python codes with several Machine Learning (ML) algorithms by

	Top Level	Bottom Level				
Model	LR	LR	MLP			
	category	type1	type1	type2	...	type10
Features	boolean	date	class1	class1	...	class1
	literal	string	...	...	...	...
	resource	number	class298	class172	...	class22

Fig. 7. Diagram of the final multi-class classifier design: three ML models cover each number of unique features.

importing *SVM*, *LogisticRegression* (LR), and *MLPClassifier* (MLP) from scikit-learn [9] and used them on SMART 2020 dataset (17,571 questions and answers for training, 4,360 questions for test) for comparison with reference models performance. To check the baseline performance, three sample ML models are initially determined for answer category classification and they return the acceptable performance: *SVM*(kernel = 'linear', random_state = 0, probability=True) results 87%, *LogisticRegression*(multi_class='multinomial', solver='lbfgs') results 88%, and *MLPClassifier*(hidden_layer_sizes = (11, 11, 11), max_iter = 500) results 85% respectively. Additionally, this study finds that the MLP model is more efficient in many number of classes such as resource type (∼760 classes) than literal type (3 classes) classification. Then, we moved the confirmed implementation to the merged training dataset to align with the SMART 2021 task requirements.

3.4 Multi-classification model implementation

CitySAT system is programmed in Google Colab CPU environment with two processing threads. The evaluation of each stage algorithms has been conducted with validation data which is 20% of the training dataset.

By expanding experiments from Section 3.3, we get the best optimised hyperparameters in a combination system of two LRs and a MLP as briefly captured in the Appendix. Figure 7 shows the results of the design of our classification model from top to bottom level, starting from a single LR model which classifies the answer category at the top level. Two different models are used to classify the answer type at the bottom level: a LR model for the type of the literal category and a MLP model for the type of the resource category.

As last step, to meet the submission format specifications, it is essential to decode all mapping data and convert the format back to the JSON format as shown by the JSON below. The CitySAT system workflow is depicted in Figure 8.


```
{
"id": "5586",
"category": "resource",
"type": ["dbo:Company", "dbo:Activity", "dbo:RecordLabel", "dbo:Agent",
"dbo:Species", "dbo:Organisation", "dbo:AdministrativeRegion",
"dbo:Location", "dbo:Country", "dbo:PopulatedPlace"]
}
```

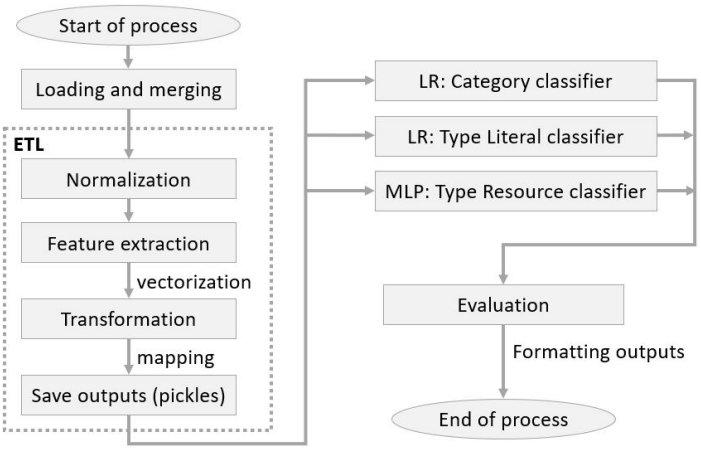


Fig. 8. System workflow chart of the final CitySAT. One of the key algorithms is for data transformation and mapping the intended number of types in the experiments looping sub-process in type classification (partially retrieved code is stated in Appendix).

4 Results

Table 3 summarizes our submission results with the corresponding configuration parameters in Table 4. The final CitySAT submission (*) obtained the best model performance on the DBpedia test dataset: 0.984 on accuracy, 0.842 on NDCG@5, 0.854 on NDCG@10.

Most importantly, the results table represents that the system design aims to perform better when it takes more knowledge-based information. For example, training with ten resource types (i.e., ontology classes) associated with a question is more informative for a MLP model than giving five types per question.

Table 5 shows the results of other systems participating in the SMART 2021 challenge. The better performance in NDCG@10 was highly valued when the

Table 3. Confirmed Results in participation of SMART 2021 challenge.

Conf.	Validation			Test		
	Accuracy	**NDCG@5**	**NDCG@10**	**Accuracy**	**NDCG@5**	**NDCG@10**
1	0.969	0.732	0.649	0.970	0.778	0.683
2	0.973	0.735	0.656	0.981	0.839	0.739
3	0.953	0.699	0.622	0.967	0.810	0.713
4	0.973	**0.737**	0.658	0.984	0.836	0.737
5	0.973	0.736	0.656	0.984	0.842	0.742
6*	**0.973**	0.736	**0.738**	**0.984**	**0.842**	**0.854**

Table 4. Configuration settings of submissions.

Conf.	Stopwords	Stemming	Lemma	Text feature	Iteration	No of Type
1	False	True	True	TF	100/10	5
2	False	True	True	TF	100/10	5
3	True	True	True	TF-IDF	100/10	5
4	False	True	False	TF	200/20	5
5	False	True	False	TF	200/10	5
6*	False	True	False	TF	200/10	10

other two scores (Accuracy, NDCG@5) are similar. The best results of CitySAT is ranked at the top in DBpedia AT task.

Table 5. SMART 2021 Leaderboard - DBpedia AT task.

System	Accuracy	NDCG@5	NDCG@10
CitySAT	0.984	**0.842**	**0.854**
Bhargav et al.	0.985	0.825	0.790
Celebi et al.	0.985	0.725	0.704
Hoang et al.	0.985	0.727	0.664
Steinmetz et al.	**0.991**	0.734	0.658
Perevalove et al.	**0.991**	0.643	0.577

5 Discussion and future works

With the given DBpedia data in the SMART 2021 AT challenge, this study tried various explorations on text normalization. Including filters of Wh-terms in stop words, there were multiple configuration settings we could have imagined to have improvement in classification performance which was not the case for this challenge. This opens the door for future work in finding how to improve the text features involving semantic meanings in a more human thought process.

Although CitySAT models are optimised in Section 3.4 to find the best combination of local classifiers for two levels, there might be more options and different combinations of models that can be discovered in the future. Especially, once we expand our evaluation environment to a Graphics Processing Unit (GPU), there are more applicable ML models for our problem.

As in previous studies, we have also found that several of the participating systems used the fine-tuned BERT models [4]. Because of limitations on computing resources, however, this study intentionally deploys the ML models in CPU computation with inexpensive computational cost during the project. In the future, injecting the BERT model in CitySAT is possible to check if any performance benefits in the AT task.

Acknowledgements

We would like to thank the ISWC conference and the SMART challenge organisers. This work was partially supported by the SIRIUS Centre for Scalable Data Access (Research Council of Norway).

References

1. Ammar, A., Mehryar, S., Celebi, R.: A methodology for hierarchical classification of semantic answer types of questions. In: SMART@ ISWC. pp. 41–48 (2020)
2. Bill, E., Jiménez-Ruiz, E.: Question embeddings for semantic answer type prediction. In: SMART@ ISWC (2020)
3. Bird, S., Klein, E., Loper, E.: Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc." (2009)
4. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). pp. 4171–4186 (2019). https://doi.org/10.18653/v1/n19-1423, `https://doi.org/10.18653/v1/n19-1423`
5. Kertkeidkachorn, N., Nararatwong, R., Nguyen, P., Yamada, I., Takeda, H., Ichise, R.: Hierarchical contextualized representation models for answer type prediction. In: SMART@ ISWC. pp. 49–56 (2020)
6. Mihindukulasooriya, N., Dubey, M., Gliozzo, A., Lehmann, J., Ngomo, A.N., Usbeck, R., Rossiello, G., Kumar, U.: Semantic answer type and relation prediction task (SMART 2021). CoRR **abs/2112.07606** (2021), `https://arxiv.org/abs/2112.07606`
7. Mihindukulasooriya, N., Dubey, M., Gliozzo, A., Lehmann, J., Ngonga Ngomo, A.C., Usbeck, R.: SeMantic AnsweR Type Prediction Task at ISWC 2020 Semantic Web Challenge. CEUR-WS **2774** (2020), `http://ceur-ws.org/Vol-2774/`
8. Nikas, C., Fafalios, P., Tzitzikas, Y.: Two-stage semantic answer type prediction for question answering using bert and class-specificity rewarding. In: SMART@ ISWC. pp. 19–28 (2020)
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
10. Perevalov, A., Both, A.: Augmentation-based answer type classification of the smart dataset. In: SMART@ ISWC. pp. 1–9 (2020)
11. Setty, V., Balog, K.: Semantic answer type prediction using bert: Iai at the iswc smart task 2020. arXiv preprint arXiv:2109.06714 (2021)
12. Steinmetz, N., Sattler, K.U.: Coala-a rule-based approach to answer type prediction. In: SMART@ ISWC. pp. 29–40 (2020)
13. pandas development team, T.: pandas-dev/pandas: Pandas (Feb 2020). https://doi.org/10.5281/zenodo.3509134, `https://doi.org/10.5281/zenodo.3509134`
14. Vallurupalli, S., Sleeman, J., Finin, T., et al.: Fine and ultra-fine entity type embeddings for question answering. In: International Semantic Web Conference (2020)

Appendix

```
# distribute categorical targets (answer category & type) to numericals
  def type_to_int(self, data, type_no):
    return data.type.map(
        lambda x: self.type_maps[f"type{type_no}"][x[type_no - 1]]
        if len(x) >= type_no
        else self.type_maps[f"type{type_no}"]["missing"]
        )
```

```
# model for category classification
clf_category = LogisticRegression(
    random_state=seed,penalty='elasticnet',solver='saga',
    l1_ratio=0.2,n_jobs=-1,verbose=2,max_iter=200)
    .fit(X_train_category,y_train_category)
# model for type of literal category
clf_literal = LogisticRegression(
    random_state=seed,penalty='elasticnet',solver='saga',
    l1_ratio=0.5,n_jobs = -1,verbose = 2,max_iter = 200)
    .fit(X_train_category[train_literal_rows,:], y_train_literal)
# model for type of resource category
clf_type = MLPClassifier(
    random_state=seed,max_iter=10,
    hidden_layer_sizes=(1000,500,300),verbose=2)
    .fit(X_train_category[train_resource_rows],y_train_type)
```