A Methodology for Hierarchical Classification of Semantic Answer Types of Questions

Ammar Ammar[1], Shervin Mehryar[2], and Remzi Celebi[3]

[1] Department of Bioinformatics - BiGCaT, NUTRIM, Maastricht University, The Netherlands a.ammar@maastrichtuniversity.nl
[2] University of Toronto, Canada shervin.mehryar@utoronto.ca
[3] Institute of Data Science, Maastricht University, Maastricht, The Netherlands remzi.celebi@maastrichtuniversity.nl

Abstract. Question answering systems have recently been integrated with many smart devices and search engines. Answer type prediction plays an important role in question answering systems as it can help filter irrelevant results and improve overall search and retrieval performance. Here, we present our approach for answer type prediction using the datasets provided for the International Semantic Web Conference (ISWC 2020) SMART Task Challenge. Predicting granular answer types for a question from a big knowledge graph is a greater challenge due to the large number of possible classes. Thus, we propose a 3-step approach to tackle the challenge task. We start with building a classifier that predicts the category of the types and build another classifier just for resource types. The latter model will predict the most general (frequent) type for each question, ignoring type hierarchy. We use a multi-class text classification algorithm built-in fastai library for these two models. The models' accuracies are 0.95 and 0.73 for category and generic type classification respectively in the validation set (20% randomly chosen samples) of the DBPedia dataset. Next, we train a third classifier to find more specific types (sub-classes) for each question based on the previous general predicted types. We achieve 0.62 and 0.61 using NDCG@5 and NDCG@10 metrics respectively for the test set.

Keywords: Question answering · Hierarchical classification · Semantic type prediction

1 Introduction

Question answering systems have recently been integrated with many smart devices and search engines. Answer type prediction plays an important role in question answering systems as it can help filter irrelevant results and improve overall search and retrieval performance. Here, we present our approach for answer type prediction using the datasets provided by the International Semantic Web Conference (ISWC 2020) SMART Task Challenge organizers [5]. The task consists of training data of questions, categories, and types from large ontologies, and the challenge participants are asked to provide the categories and types from the WikiData and the DBpedia ontologies questions in the test dataset. The DBpedia dataset contains 17,571 training questions and 4,393 test questions, and the WikiData dataset has 18,251 training questions and 4,571 test questions. Each training dataset is labeled with an answer category and a list of types. These questions consist of short text that can be classified based on the answers into three main categories: boolean, literal or resource. The granular categorization is possible for 'resource' categories using knowledge graphs/ontologies classes including the WikiData ($\sim$ 50K classes) and the DBpedia ($\sim$ 760 classes). Predicting granular answer types for a question from an ontology is more challenging due to the large number of possible classes. Thus, we present a 3-step approach to tackle the challenge task.

2 Related Work

Most question answering systems are able to answer a wide range of world knowledge on a production scale. However, they tend to have a modular architecture and depend laboriously on information retrieval techniques. These systems are known to rely on limiting the possible subset of candidates, known as answer type modeling, which notably increases both speed and quality. Generally, the main approach of question answering starts with building a labeled query-type dataset and then selecting a answer prediction model. Several model architectures including recurrent neural networks and feed-forward transition-based neural networks have been proposed to tackle this problem. For example, in Ivan Bogatyy's work [2], a model was developed using normalized transition-based neural network parser. The data was encoded into two types of features: binary features generated for the slots corresponding to the ROOT and NSUBJ of the sentence, and integer features generated for every slot based on its depth in the syntactic tree. Also, J. W. Murdock et.al [7] proposed a method that utilizes a variety of strategies and resources to decide whether the candidate answer has the desired type. These strategies and sources provide a set of type coercion scores for each candidate answer. They used these scores to give preference to answers that are more likely to have the right type. This method with type coercion is significantly more accurate than it is without type coercion and has a combined impact of nearly 5% on the accuracy of the IBM Watson. Moreover, Huan Sun et.al. [10] constructed answer-type related features with two novel probabilistic models. Such semantic features appeared to play outstanding roles in determining the true answers from the large answer candidate collection. Using two test datasets, the aforementioned question-answering system achieved an improvement of 18% and 54% with the F1 metric, in comparison to several available QA systems. Finally, Semih Yavuz et.al. [11] generated an abstract form of the questions by replacing their topic entities with their types. A bidirectional LSTM model was built to train over the abstract form of questions and predict their answer types. The model was able to improve the F1-score from 49.7% to 52.6% on the WE-BQUESTIONS dataset.

3 Methodology and Results

We have developed a workflow to address the challenge of the ISWC 2020 where the categories and types of questions must be predicted using classes from two main Knowledge Graphs (KGs) ontologies: Wikidata and DBpedia. The workflow for the proposed methodology is shown in Fig. 2 and the code for the workflow can be accessed at our Github repository . For prediction of the question categories (Task 1), we model the problem as multi-class classification to predict one of the extended categories (`boolean, number, string, date and resource`) from question text using the fastai text classification library [4]. For each of the datasets (Wikidata and DBpedia), the dataset given for training is split into 2 subsets: training set (80%) and validation set (20%). The preprocessing (tokenization and numericalization if needed) is handled automatically within the fasiai data class (`TextDataBunch`). The classifier (`text_classifier_learner`) uses the LSTM neural network model [3] and is trained with the best learning rate ($1e-2$), after which the process of fitting is repeated two cycles; the first cycle with learning rate ($1e-2$) and the second cycle with learning rate ($1e-3$). The model trained achieves an accuracy of 0.947 for category prediction.

For the type prediction problem (Task 2), we first obtain the most frequent type among all the question types that are predicted to have a 'resource' category, and try to predict the generic type from the question text using the same text classification model (`LSTM neural network`) of the fastai library. The model achieves an accuracy of 0.73 for generic type prediction. To examine the errors made by the classifier , we plot the confusion matrix summarizing errors for validation set shown in Figure 1. Agent type is the classifier's most predicted type, and also the type for which the classifier has the most errors. It can be seen from the confusion matrix that the classifier confuse especially the Agent class with Place class.

Once the prediction for generic (frequent) types was obtained, a specific type was determined for each question by selecting a more specific type in each question type list . We used a random forest model to predict the specific type. The input representation for this model was constructed by integrating three vector representations (embeddings) for the question text, the frequent type and the specific type. For the question text, two vector representations were used. First, questions sentences were embedded using a pre-trained BERT language model [8], resulting in a vector of length 1024 for each question. Next, word and entity embeddings (vector size of 100) were extracted using Wikipedia2Vec. Wikipedia2Vec is a Python-based open-source tool for learning the embeddings

https://github.com/rcelebi/iswc2020-smarttask
https://wikipedia2vec.github.io/wikipedia2vec/

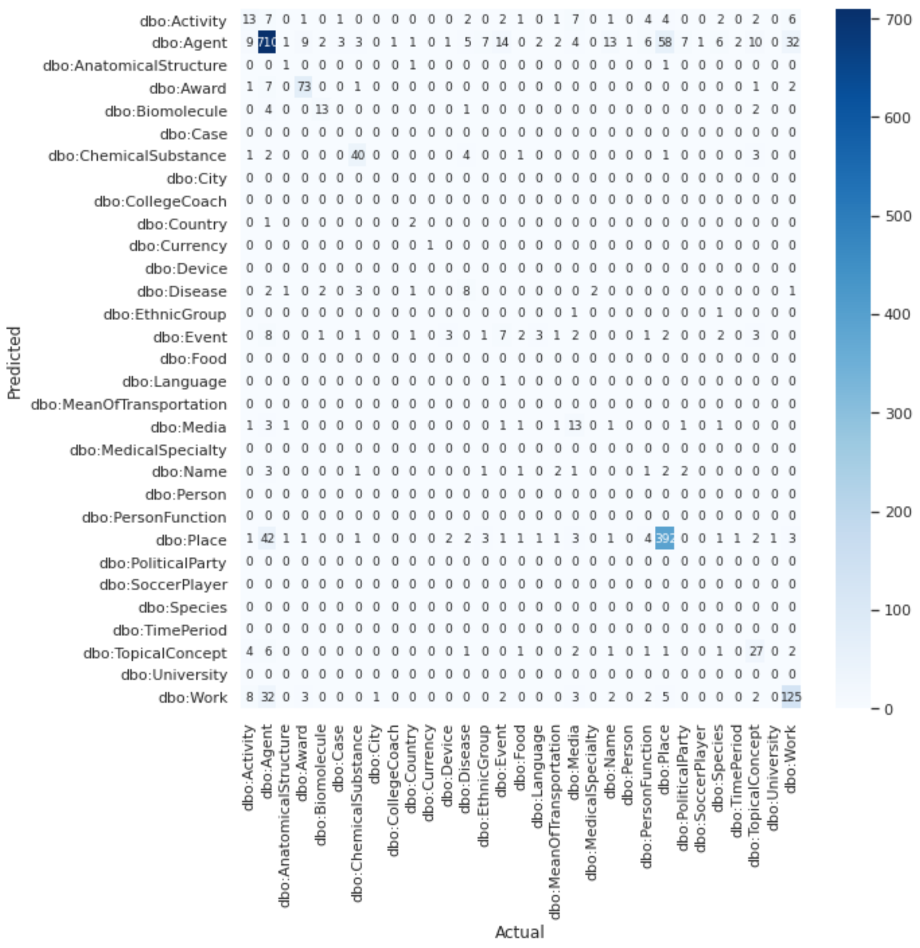

Fig. 1. Confusion matrix summarizing the errors made by the classifier for generic type prediction

of words and entities from Wikipedia. Its results on the KORE entity relatedness dataset achieved the state-of-the-art results, and it also achieved competitive results on various standard benchmark datasets. Next, the embeddings of tokens (word or entity) extracted from each question) were averaged to obtain a fixed size vector for each question. The sentence embedding and the word embedding are concatenated to represent a question. For generic and specific types, vector representation was learnt by training a word2vec model [6]. To generate the embeddings that capture the hierarchical relationship of types, we used the ontology hierarchy of Wikidata/DBpedia KG. Next, the hierarchies were flattened into a sequence of ontology terms that were used as input for embedding learning. Here, we used the Word2Vec approach in which the "CBOW" neural network with a layer size of 100 was used to generate type embedding. The three embedding vectors (question, generic type and specific type) were integrated averaging the embedding for generic and specific type followed by concatenating with question's embedding. We modeled this problem as a binary classification problem in which the given generic type and specific type match the question in the positive examples, but not for negative examples. Because we lacked negative examples to train the classifier, we generated negative examples by replacing the specific type in the positive instances by another type that shares the same parent type in the hierarchy but not a correct specific type (see Fig. 2). The positive instances were labeled as "1" (output variable) and the negative instances were labeled as "0". The resulting dataset was split into 2 subsets: training set (80%) and validation set (20%). The random forest model was trained on the training set and achieved an accuracy of 0.89 on the validation data. Finally the random forest model is used to provide the top-K predictions (question, generic type and specific type) for a given question and a generic type (obtained from the previous text classifier). We report that our final submission using the proposed workflow achieves 0.62 and 0.61 using NDCG@5 and NDCG@10 metrics respectively for the DBpedia set. The correct handling of negative example selection has an impact on this stage and proper hierarchical structures that take advantage of the underlying connections which we leave for future research direction. When there are multiple classes to predict, these results can be improved upon via examplar selection [1] and reasoning type methods for knowledge basis [9] for example.

4 Conclusion

Since most state-of-the-art classifiers have limited capabilities in granular classification tasks, we propose a framework that focuses on hierarchical type prediction thus enabling current methods to take advantage of this method for engineering features that can subsequently improve upon the already existing methodologies such as Random Forests. Our methodology consists of a 3-step process. In step one, by using off-the-shelf multi-class text algorithms, the task of category prediction and type prediction, was separated from general versus specific predictions, thereby ignoring type hierarchy. The models' accuracy levels are 0.95 and 0.73 for category and generic type classification respectively. Next, a third classifier is trained to predict specific types using the previous general categorization results. By combining the feature vectors from the question text, general prediction, and specific type, and applying averaging to their embeddings into a fixed size binary classification method, we determine a positive versus negative specific type for a given generic one. We report performance of the frame-work on granular categorization data for 'resource' categories using ontologies classes including the DBpedia using `NDCG@5` and `NDCG@10` metrics. For test, we achieve 0.62 and 0.61, respectively in top-K prediction task and aim this for future research direction.

References

1. Awasthi, A., Ghosh, S., Goyal, R., Sarawagi, S.: Learning from rules generalizing labeled exemplars. cs/arXiv **abs/2004.06025** (2020), https://arxiv.org/abs/2004.06025
2. Bogatyy, I.: Predicting answer types for question-answering (2016), https://cs224d.stanford.edu/reports/Bogatyy.pdf
3. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997)
4. Howard, J., Gugger, S.: Fastai: A layered api for deep learning. Information **11**(2), 108 (Feb 2020). https://doi.org/10.3390/info11020108
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. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013)
7. 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). https://doi.org/10.1147/JRD.2012.2187036
8. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (11 2019), http://arxiv.org/abs/1908.10084
9. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems. vol. 26, pp. 926–934. Curran Associates, Inc. (2013)
10. Sun, H., Ma, H., Yih, W.t., Tsai, C.T., Liu, J., Chang, M.W.: Open domain question answering via semantic enrichment. In: Proceedings of the 24th International Conference on World Wide Web. p. 1045–1055. WWW '15, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2015). https://doi.org/10.1145/2736277.2741651
11. Yavuz, S., Gur, I., Su, Y., Srivatsa, M., Yan, X.: Improving semantic parsing via answer type inference. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 149–159. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/D16-1015

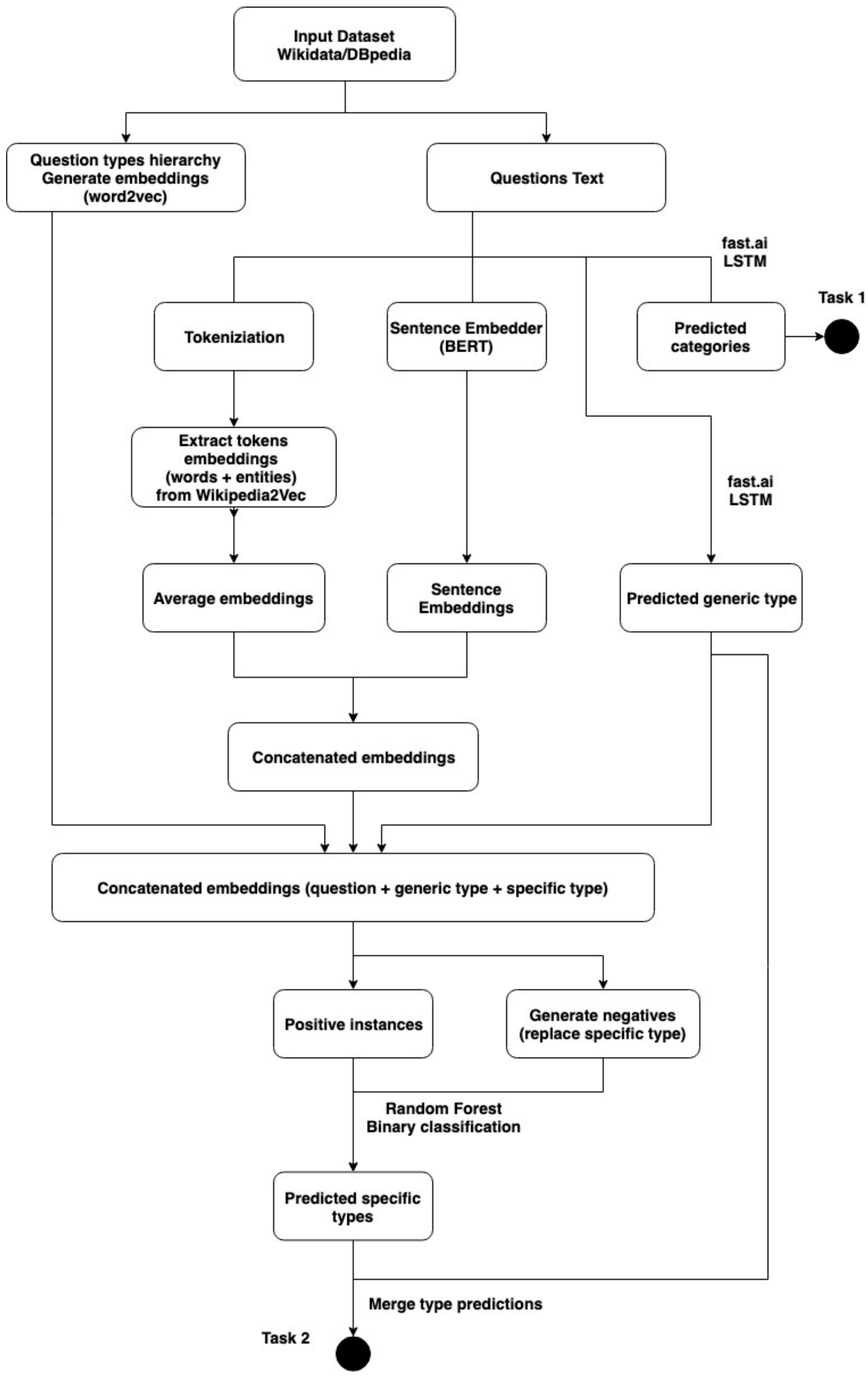

Fig. 2. The workflow for the proposed methodology

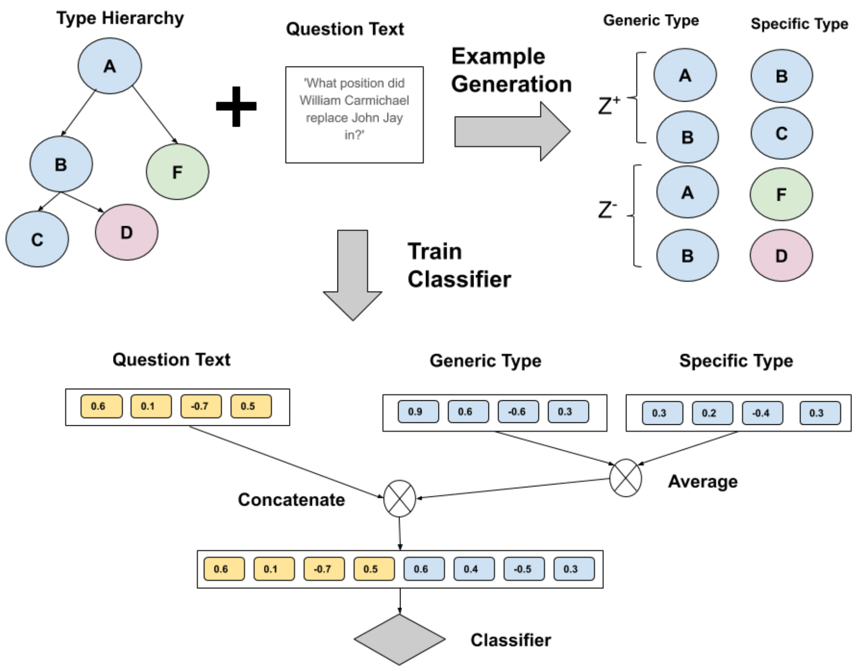

Fig. 3. For a particular type hierarchy, a question can be labeled with multiple types of type paths that represent branches in a type tree. For the given toy example, the question is labelled with (A, B, C) types that are part of a type path in the given hierarchy. A hierarchical type prediction problem can be formulated as a binary classification that maximizes the probability of finding correct generic type and specific type for a given question. For this toy example, the parent and child pairs (A, B), (B, C) that are seen in the training data are taken as positive instances, and the parent and its other siblings (A,F), (B,D) are taken as negative instances. The binary classifier takes a concatenation of the question embedding and the average type embedding as input to discriminate postives from negative instances.