Team-009: Semantic Answer Type Prediction

Eirik Haraldsen
University of Stavanger
Stavanger, Rogaland, Norway

Karl Østrådt
University of Stavanger
Stavanger, Rogaland, Norway

ABSTRACT

The semantic answer type prediction task is a challenge for the International Semantic Web Conference (ISWC) 2020. The challenge involves answer type prediction using a set of candidates from a target ontology. In this paper we propose a solution to the challenge by first developing a baseline method for answer category and answer type prediction. Second, we identify areas of improvement in the baseline method, and develop an advanced method with the proposed improvements. Finally, we evaluate the performance of both models, discuss implementation choices and suggest further improvements to our approach.

CCS CONCEPTS

• **Information systems** → **Information retrieval**.

KEYWORDS

Query classification, information retrieval, semantic search

ACM Reference Format:
Eirik Haraldsen and Karl Østrådt. 2020. Team-009: Semantic Answer Type Prediction. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The ultimate goal for information retrieval systems is to provide useful and correct data regardless of the intent and context of a query. The intent indicates whether the answer should be binary, a date, a number, text, or a semantic entity. Knowing the context will help differentiate homonyms. E.g. a *bat* can either mean an instrument to hit a ball or an animal depending on the context. Predicting what type an answer should have is crucial for an accurate information retrieval system. Identifying and understanding the intent and context of a query will significantly help the information retrieval system accurately predict the answer types of a question.

In this work we present a solution to the semantic answer type prediction task described in [3]. We employ an artificial neural network to classify the answer category of a question. The answer category is used to understand the intent of the question and what the answer type should be. Furthermore, we use Elasticsearch to index and rank documents, and word embedding to understand the context of the queried question. Together, these technologies are used to predict answer types of a question.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 PROBLEM STATEMENT

We define the semantic answer type prediction task as follows: given a question in natural language, predict the answer category and the associated answer type(s) from a target ontology. The valid answer type(s) are defined in the DBpedia class ontology [2]. A question is typically short in length, thus the category prediction task can be defined as short text multi-class classification problem.

In this project, we use the DBpedia dataset and the corresponding evaluation script from [5] to develop and evaluate our solution. A 75/25 split was used to divide the DBpedia dataset into a training and test dataset. The dataset contain a list of questions, where each question has an ID, question text in natural language, answer category and answer type(s). The categories can be either *boolean*, *literal* or *resource*. If a question's category is boolean, then the answer type is always boolean. The literal category indicate that the answer type is *string*, *date* or *number*. For the resource category, all valid answer types are defined in the DBpedia class ontology. If a question contains the answer type t, then it is expected that all parent types in the ontology hierarchy should also be assigned as an answer type. All questions with missing answer type were removed from the training and test datasets.

The classification of answer category is heavily reliant on the WH-terms (who, what, when ...) in the question. However, it is not feasible to perform this prediction with a trivial mapping. Some questions contain multiple WH-terms where the word order define the context, while other questions do not contain any WH-terms at all. The answer type of a question is dependent on the answer category of the question. Therefore, it is more natural to predict a question's answer category first and then use the predicted category to predict the answer type. In the answer category prediction task, the input is the list of questions from the dataset, and the output is list of their corresponding category prediction. In answer type prediction task, the input is the list of questions from the dataset & the corresponding predicted categories, and the output is the predicted answer types.

3 BASELINE METHOD

3.1 Category prediction

The answer category prediction task can be defined as a multinomial classification problem due to the fixed number of class labels. The baseline method for this task is a neural network model using a multi-layer perceptron (MLP) approach. The training dataset has to be converted into a set of feature vectors in order to train the MLP model. All questions in the training dataset are extracted and tokenized to create a vocabulary dictionary of all the words. This vocabulary dictionary is applied to every instance in the training

set to create a list of sparse term frequency feature vectors. The main reason for choosing term frequency over TF-IDF is that interrogatives are frequently occurring words. These interrogatives are strongly indicative of the answer category of the question. The feature vectors and corresponding labels are fed to a MLP classifier in order to train the MLP model. The MLP classifier is trained using a back-propagation technique with a stochastic gradient-based optimizer (adam). The parameters for the MLP classifier includes a hidden layer size of 100, the activation function is relu and the L2 penalty (alpha) is set to 0.0001. Optimization of model parameters are further explored in section 4.

3.2 Type prediction

After predicting the answer category, the next step is to predict the questions answer type. Each answer category is handled separately due to the difference in answer type structure, as described in section 2.

3.2.1 Boolean. Questions with answer category boolean should always be assigned boolean as the answer type. This means there is no need for further prediction for question predicted with this answer category.

3.2.2 Literal. Type prediction in the literal category can also be interpreted as a multinomial classification task. The task can be defined as: given a collection of questions with literal answer category, classify each instance into one of the three possible answer types. The implementation of this predictive model is very similar to the model described in section 3.1. A list of term frequency feature vectors was created from only questions with literal answer category in the training dataset. The feature vectors and the corresponding answer type labels were used to create a MLP model designed for literal answer type prediction. The parameters used for this model is the same as the category prediction model.

3.2.3 Resource. The most complex answer category is the resource category. Questions in this category can have one or more answer types from the DBpedia ontology. Each ontology class has a label in camel-case and the prefix 'dbo:' (e.g. dbo:MusicalArtist). The baseline approach for this answer type was to identify similar questions in the training dataset, extract the associated ontology classes, and then use word embedding to determine possible answer types.

The search and analytics engine Elasticsearch was used to index all questions with resource category in the training dataset. The ID of each document is set to the question ID, while the question text and the list of answer types are indexed as separate field. Elasticsearch uses the ranking function BM25 to identify relevant questions from a given query.

The word embedding technique used in this project is the word2vec skip-gram approach. The benefit of using skip-gram over continuous bag-of-words (CBOW) is better performance on infrequent words, and the ability to capture multiple meanings of a single word. This is very important for words like "Apple", where the context could be either the company or the fruit. The word embedding model use a list of strings as input in the training process. Each string in the input consist of the question text and the corresponding answer types. This implementation allows the model to understand the context between the question and the expected answer types. The word2vec model use a windows size of 10, the dimension of the word vectors is set to 100, and no terms are ignored (min_counter = 1).

Stop-word removal was performed on the test question. The resulting string is used to query the Elasticsearch engine. The stop-word removal was performed to reduce the amount of useless terms in the query, resulting in more accurate search results. All unique answer types from the top five most relevant documents are stored as possible answer types. Every word in the query is matched against every possible answer type. If the word similarity of two words is equal to or higher than a threshold, the answer type is added to a list of predicted answer types. This process is repeated for all questions predicated with answer category resource. The baseline approach use a threshold of 0.85 to determine if an answer type should be added or not. We believe that combining document ranking and word similarity will capture the context of a question, thus creating an interesting baseline for answer type prediction.

4 ADVANCED METHOD

Our advanced method aims to improve upon the baseline method. The first step is to experiment with parameters and look for improvements for the MLP classifiers. We modify the parameters in an attempt to solve any potential over- or under-fitting of the model. At first, we attempted a grid-search approach for finding the optimal parameters. However, we quickly realized that such an exhaustive approach would take too long, and settled for manually trying out different values for the *alpha* and *max_iter* parameters. Only the *alpha* parameter affected the performance of the classifier as it appeared to have converged before triggering the *max_iter* limitation. The category classifier performed best when the alpha parameter was increased to 1. The literal type classifier did not appear to have any real improvement when changing alpha and max_iter.

In some cases, the function that predicts the resource types were unable to predict any types. A resource question with zero predicted types would have a NDCG@k score of zero. The root of the problem was the similarity threshold being too high. This was solved by repeatedly lowering the threshold until a satisfactory amount of five or more types were predicted. The threshold was not allowed to go below 0.60 because terms with such low similarity are unlikely to be a correct answer type. This means that it is still possible to predict zero answer types, however, it is much more unlikely.

Due to time constraints, we were unable to implement additional features for the advanced method. Further improvements to the advanced method are discussed in section 6.

5 RESULTS

In this project, accuracy is used to evaluate category prediction performance. Lenient NDCG@k with a Linear decay, from the paper by Balog et al. [1], is used as evaluation metric for type prediction. The lenient NDCG@k metric grade predicted answer types based on the DBpedia ontology class hierarchy. Wrongly predicted answer types are still rewarded if they are on the same branch in the hierarchy. The DBpedia evaluation script from the SMART Task challenge github repository is used to evaluate the performance of our models [5].

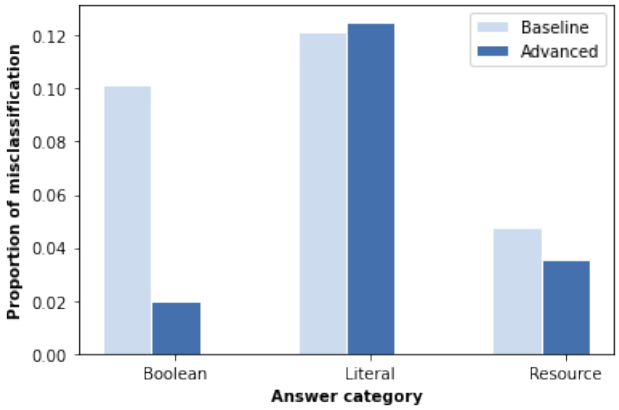

Figure 1: Proportion of answer category misclassification

Table 1: Evaluation of models

Model	Accuracy	NDCG@5	NDCG@10
Baseline	0.922	0.547	0.537
Advanced	0.941	0.618	0.590

Figure 1 shows the proportion of misclassification of each answer category for both models. It is evident that the improvements made in the advanced approach has a significant impact on the classification of boolean questions. The results also show an improvement in classifying resource questions, and a slight increase in misclassification for the literal category. The majority of questions is of the resource category. Thus, a minor reduction in resource misclassification yields a larger improvement in type class prediction compared to the other categories.

Table 1 presents the evaluation results for the baseline and advanced approach. The results shows that the advanced approach outperform the baseline method for all evaluation metrics. Presumably, the main reason for the improved results is the modification in the resource type prediction function. The improved function is more likely to predict answers where the baseline method predicted none. The advanced approach is also more likely to predict additional answer types. The average number of predicted answer types increased from 3.0 in the baseline approach, to 4.5 in the advanced approach. The additional predictions seem to be near-miss predictions, increasing the NDCG@k score for each question.

6 DISCUSSION AND CONCLUSIONS

In this paper we present a solution to the semantic answer type prediction task by Dubey et al. [3]. We introduced a baseline method using a MLP neural network for answer category and literal answer type prediction. For resource answer type prediction, we use the Elasticsearch engine to identify similar questions in the training dataset and extract the associated ontology classes. Furthermore, we use a word2vec skip-gram approach to predict the answer types based on word similarity. We also introduce an advanced method in an attempt to improve the performance of the baseline approach. The advanced method dynamically change the word2vec similarity threshold, and utilize hyper-parameter optimization to improve performance.

MLP classifiers were used to classify the category and the literal type of a question. We chose this approach because multi layer perceptrons are versatile and well suited for classification prediction problems. Additionally, the MLPClassifier from scikit-learn is easy to implement and use. It only requires the input to be on a tabular or vectorized form. Transforming the dataset into feature vectors could be done using term frequency or TF-IDF. The weight of frequent terms would be diminished with the use of TF-IDF. However, interrogative words are frequently occurring terms which are strongly indicative of a questions answer category. For that reason, term frequency is better suited for answer category prediction.

Elasticsearch provides a simple and quick way to index questions and their corresponding categories and types as a document. This also alleviated the need for implementing an information retrieval system by ourselves. Moreover, Elasticsearch enabled the baseline method to identify and rank similar questions from a query. In other words, Elasticsearch is one of the core technologies in our approach for answer type prediction.

Our initial word embedding approach involved using a word2vec model with a continuous-bag-of-words architecture. This model was quickly discarded due to poor performance on infrequent word, and words with multiple meanings. However, by changing the model structure to skip-gram, the model was now able to understand words with multiple meanings. This is mainly due to the skip-gram architecture using a single word to predicting the context. Additionally, word2vec models with skip-gram architecture are also known to perform better with infrequent words [4]. An important limitation of this approach is the window size used in the model. The windows size determines how many words before and after the target word should be used to determine the context. If this number is too large, the model will use the question text and answer types from multiple questions to determine the context. If the number is too low, the model will only use parts of the question text and answer types to determine the context. Both cases will result in a model with poor performance.

The results presented in section 5 show that the advanced method has a noticeable increase in performance compared to the baseline method. It was evident that the baseline approach had its limitations. The average number of predicted answer types for each resource question was three, with some questions having zero predicted answer types. By dynamically changing the word similarity threshold for each question, we were able to predict more correct and near-miss answer types in the advanced method. Additionally, the increased accuracy in the category prediction reduced the proportion of misclassification for both boolean and resource categories.

As stated in section 4, we were unable to implement all our ideas for the advanced method due to time constraints. Our original plan was to implement three additional measures that were likely to improve the performance of the advanced method. Currently, every indexed document with Elasticsearch contains minor amounts of text. The documents could be expanded by appending a short abstract description for each answer type in the document. Increasing the amount of text in each document would likely improve the search results. The second and third measures both involve using

the ontology class hierarchy to predict additional answer types. We already know that if a question contains the answer type t, then it should also contain all ancestor types of t as well. Whenever a new answer type is predicted, the program should search through the type hierarchy and look for any ancestor types to be added. The addition of this measure would likely result in additional correct answer type predictions. Finally, the order of the answer types in the training dataset appears to be ranked in the order of most specific to least specific. Therefore, the predicted answer types should be sorted or re-ordered in accordance with the DBpedia ontology class hierarchy. We believe that these additions would significantly improve the performance of the advanced method.

REFERENCES

[1] Krisztian Balog and Robert Neumayer. 2012. Hierarchical target type identification for entity-oriented queries. *ACM International Conference Proceeding Series*, 2391–2394. https://doi.org/10.1145/2396761.2398648

[2] DBpedia. 2020. *DBpedia Ontology.* Retrieved November 4, 2020 from https://wiki.dbpedia.org/services-resources/ontology

[3] Alfio Gliozzo, Axel-Cyrille Ngonga Ngomo, Jens Lehmann, Mohnish Dubey, Nandana Mihindukulasooriya, and Ricardo Usbeck. 2019. *Semantic Answer Type Prediction Task.* Retrieved November 4, 2020 from https://smart-task.github.io/

[4] Ria Kulshrestha. 2019. *NLP 101: Word2Vec — Skip-gram and CBOW.* Retrieved November 15, 2020 from https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314

[5] Mohnish Dubey Nandana Mihindukulasooriya and Krisztian Balog. 2020. *smart-dataset.* Retrieved November 4, 2020 from https://github.com/smart-task/smart-dataset