Semantic Answer Type Prediction 2020

Team-001

Stephan Frederik Werner Brandasu
University of Stavanger
sf.brandasu@stud.uis.no

ABSTRACT

We describe our solution to the SeMantic AnsweR Type (SMART) prediction task 2020 for the DBpedia dataset. Our methods will take advantage of BM25 scores that get modified by the presence of specific keywords in the questions.

KEYWORDS

information retrieval, answer category classification, answer type prediction, natural language understanding, understanding query types

1 INTRODUCTION

Answer Type Prediction (ATP) is a challenging problem in the natural language processing field which is an important step to being able to understand natural language queries. Understand the category and type of a given question is an important step in reaching this goal.

The SMART task has provided 2 datasets, one using the DBpedia ontology and another using the Wikidata ontology. Both datasets follow the same structure and will have a question id, a question text in natural language, an answer category and an answer type. They do differ in one sense though, If the category is a 'resource' then the types from the DBpedia dataset are different from the ones in the Wikidata dataset. This has to be taken into account when attempting to classify the questions.

The goal of this paper is to as accurately as possible classify the category and type of a natural language question.

2 RELATED WORKS

2.1 Question word categories

A previous submission to the SMART task by **?**] from the University of Maryland found that the category of an answer is highly dependent on the question words. They found for example that questions starting with 'Is' or 'Does' always expect a boolean answer and questions starting with 'when' could expect either a number or a date[**?**]. A graph showing the most common question words found by this paper can be seen in figure **??**.

2.2 Most frequent resource types

The submission by **?**] from the University of Applied Sciences in Kothen Germany did some analysis on the dataset and found that which types of resources came up most commonly over the whole dataset. This information can be used to weigh the possible answers to try to deduce the correct answer type for a given question[**?**]. A graph showing the most common answer types found by this paper can be seen in figure **??**.

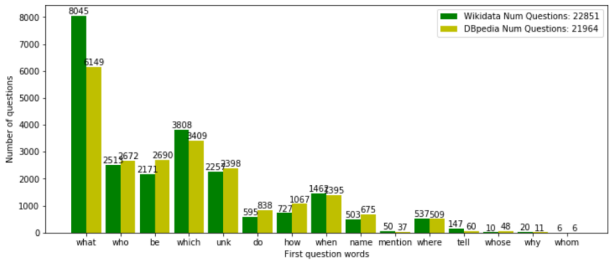

Figure 1: "First words in sentences from Wikidata and DBpedia datasets" by ?]

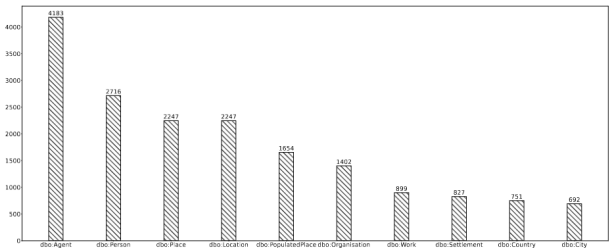

Figure 2: "TOP 10 resource answer types" by ?]

2.3 Common methods for classification

A model used frequently used for question classification by previous submissions is Bidirectional Encoder Representations from Transformers (BERT)[**?**][**?**][**?**][**?**][**?**].

BERT is an open source machine learning framework used for natural language processing. The model has been pre-trained but also can be fine-tuned with your own datasets. What is unique about BERT is that it can learn bidirectional representations with transformers. This allows for much larger data sets to be analyzed more quickly since words can be processed in relation to all other words at once, instead of one at the time. Additionally by using transformers BERT can better understand the context of a word since it compares it to all the other words at once, this makes it extremely capable for text classification tasks such as this.[**?**]

3 PROBLEM STATEMENT

The objective of this task is to take a natural language question and correctly classify the category and types expected from said question. More broadly this means that a systems needs to be created which takes a normal human written sentence and knows whether to categorise this question as a resource, a literal or as a boolean. Depending on the category it must also identify the correct types for each given category.

A boolean can only have boolean as a type. Meanwhile a literal could be expecting a string, a date or a number as a type. Resources expect more specific type predictions, first of all if the category is resource then there is a chance more than 1 type is expected as an answer. Second of all the types are expected to exist within a hierarchy based on how specific they are.

4 BASELINE METHOD

The baseline method used is a basic BM25 implementation from the $rank - bm25$ Python library. The reason for using normal BM25 instead of BM25F or BM25+ is that these modifications aim to improve BM25's performance on multi fielded documents or when dealing with documents with large variance in length but this is not something we need to take into account here.

The input questions and the training data was preprocessed to make it lower case, remove punctuation, stem them and return each question as a list of words. Stopwords are not being removed because questions are so short that the stopwords become a useful part of the context of the question. The results from the preprocessing and the BM25 scoring can be seen in table **??**

Table 1: Baseline method results

Accuracy	*NDCG*@5	*NDCG*@10
0.342	0.113	0.112

In the baseline method the category and type predictions were not done separately. This means that when the program is scoring the question against each question in the training dataset it simply takes the category and types from the highest matched question without any other modifications.

Additionally the scoring is done by multiple parallel processes using the standard multiprocessing library. This allows the algorithm to finish processing the dataset much faster. While this doesn't improve the score this will be useful to more quickly test any changes being made in the advanced method and see if they improve the results as expected.

5 ADVANCED METHOD

trying to weigh answer type based on the first word of a question (who, where, what, why)

Table 2: Advanced method results

Accuracy	*NDCG*@5	*NDCG*@10
TODO	TODO	TODO

6 RESULTS

7 DISCUSSION AND CONCLUSIONS

Table 3: Summary of results

	Accuracy	*NDCG*@5	*NDCG*@10
Baseline	0.342	0.113	0.112
Advanced Method	TODO	TODO	TODO
Advanced Method 1	**TODO**	**TODO**	**TODO**

A GITHUB

The project code and datasets can be found here: github.com/sbthepotato/DAT640-Project-22H