Fine and Ultra-Fine Entity Type Embeddings for Question Answering

Sai Vallurupalli, Jennifer Sleeman, and Tim Finin

University of Maryland at Baltimore County, Baltimore, MD 21250 USA
kolli,jsleem1,finin@umbc.edu

Abstract. We describe our system for the SeMantic AnsweR (SMART) Type prediction task 2020 for both the DBpedia and Wikidata Question Answer Type datasets. The SMART task challenge introduced fine-grained and ultra-fine entity typing to question answering by releasing two datasets for question classification using DBpedia and Wikidata classes. We propose a flexible framework for both entity types using paragraph vectors and word embeddings to obtain high quality contextualized question representations. We augment the document similarity provided by paragraph vectors with semantic modeling and sentence alignment using word embeddings. For the answer category prediction, we achieved a maximum accuracy score of 85% for Wikidata and 88% for DBpedia. For the answer types prediction, we achieved a maximum MRR of 40% for Wikidata and a maximum nDCG@5 of 54% for DBpedia datasets.

Keywords: Word Embedding · Document Embedding · Paragraph Vectors · Fine-Grained Entity Typing · Ultra-Fine Entity Typing.

1 Introduction

To further research in the area of question answering using entity types, the 19th International Semantic Web Conference (ISWC) 2020 has put forward the Semantic Answer Type (SMART) prediction task challenge. Two new datasets were released with the goal of classifying the questions into hundreds of fine and thousands of ultra-fine types using DBpedia and Wikidata ontologies. The two SMART challenge datasets include a total of 44,786 questions which are more varied than the short and single sentence factoid questions from the UIUC and the TREC QA datasets[9, 14]. The task goal is a dual classification where each question is assigned a single answer category, and an unknown number of answer types of the expected answer. While this task is considered a short-text classification, what makes the classification challenging is a few unique characteristics of the datasets: 1.) they are composed mainly of simple high frequency words, some of which are considered to be stop words, 2.) these simple high frequency stop words are distributed uniformly among all questions, 3.) sentence grammar and structure are varied and noisy, 4.) answer type labels are not uniformly distributed in the training set, with two-thirds of the labels having less than five training samples, 5.) a question can be labeled with multiple answer types, with

the number of labels being unknown a priori, and 6.) when no external resources are available, the task is similar to classification in a low resource language. These characteristics contribute to the data sparsity present in the datasets.

In this paper we describe a novel and flexible framework that builds upon the seminal work of distributed representations of paragraphs, sentences, phrases and words [11, 7]. It trains Doc2vec [7] and Word2vec[11] models. It extracts novel syntactic, grammatical and distributed representations from the question semantics. Using these semantic representations, it applies nested filtering to the top-N documents inferred by the trained Doc2vec model, with the overall goal of improving the accuracy of the dual classification. The framework leverages the unique aspects of the data to deal with sparsity. It generalizes well, such that it can be applied to both datasets. It is flexible and can be used for both DBpedia and Wikidata class labels with minor parameter changes. It offers an additional benefit in that it can incorporate external data for further refinement.

2 Background

Distributed Representation of words and phrases (word embeddings) [11] and distributed representations of documents and paragraphs (Paragraph Vectors) [7] have shown to be effective for a wide variety of tasks [2, 5, 15, 6, 3]. These models are called Word2vec and Doc2vec respectively. In this section, we briefly provide a background of these models and their theoretical basis.

Word Embeddings. Word embeddings are used as a neural network classifier trained to learn the surrounding words within a fixed window on either side of a word, aka the word's context. There are two models based on whether the given word predicts the context as in the skipgram model, or the context predicts the given word as in the continuous bag of words (CBOW) model. In this paper we use the skipgram model. For a training corpus consisting of C contexts, given a word w, and its context c, the objective of the model is to maximize the conditional log probability of $p(c|w, \theta)$ for the corpus, i.e., $\sum_{(w,c)\in D} \log(p(c|w, \theta)$. If we denote the vector for w as v_w and the vector for the context as v_c, the conditional probability is represented as:

$$p(c|w, \theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}} \tag{1}$$

After training on a number of words and contexts, the model increases the quantity $v_c.v_w$ for words that share contexts, and decreases it for words that do not share contexts. Any two words sharing similar contexts, and any two contexts sharing many words will end up having similar vectors, resulting in a high cosine similarity between the vectors.

Paragraph Vectors. Paragraph vectors are used as a neural classifier for question sentences to learn words in a question. Of the two paragraph vector models, distributed memory (DM) and distributed bag of words (DBOW), we used the

DBOW model. This model is similar to the skipgram Word2vec model described above, where a document, instead of a word, is used to predict the context. The model objective is to maximize the conditional log probability $p(c|d, \theta)$ for the corpus where the context c is for a document d.

3 Related Work

Most of the related work identify entity types for the given entity mentions to improve a downstream task of question answering. We highlight a few of these methods that offer incremental improvements and we describe how our method is different from these previous approaches. Sun et al. [13] used Freebase entity types to rank answer candidates for a given question. However, they used a search engine to retrieve sentences related to a question, for which they applied entity linking to extract entities. They used the answer candidates and Freebase for entity typing. Since our method uses low-dimensional embedding models, we are able to achieve a richer understanding of the context of the questions. Dong et al. [4] used 22 different types from DBpedia to classify entity mentions in questions with two methods. They combined the context representations obtained from a multilayer perceptron model, and the vector representations of entity mentions obtained using a recurrent neural network, to predict type information. Yavuz et al. [17] built upon the work of [4] using type information to improve semantic parsing for question answering. The semantic parsing component maps the natural form of a question to an abstract semantic representation of the question by replacing entity mentions with type information. They train a bidirectional LSTM on the abstract forms of questions to infer answer types. In a recent work, Choi et al. [1] proposed a bidirectional LSTM for predicting natural language phrases describing the entity mentions in a given sentence. We believe we can achieve sufficient answer typing with our approach which includes both a word and a document embedding. Not only does our method offer contextual understanding similar to previous work, our unique combination of the two embedding models offers more flexibility, and is able to work on more varied types of sentences.

4 Methodology

Our method is based on a unique collaborative combination of both Word2vec and Doc2vec models [7, 11] designed to achieve better contextual understanding of questions. Doc2vec builds distributional representations of questions. Word2vec helps with contextualizing the expected answer type. Training in both models is completely unsupervised using only the questions without the gold labels. Our trained Doc2vec model maps a given test question into the document embedding space. The top N similar training questions in the embedding space are filtered using syntactic and grammatical modelling applied with the Word2vec model.

We select syntactic and semantic feature words which aid in aligning questions of similar category and types. We group these words into three types:

Q-word, *Action words* and *Anchor words*. Using word vectors which we obtain from training a Word2vec model with the question sentences, we assign a Q-word and a similarity vector to every question. Given a test question, we filter the top N similarity matches obtained from the Doc2vec model into two groups. The first group consists of questions with the same Q-word as the test question. The second group consists of questions where the subject vector of the question has a high cosine similarity with the subject vector of the test question. The first group is used to find the answer category, and the second group is used to find the answer types. We describe our approach as a dual classification framework as shown in Fig. 1.

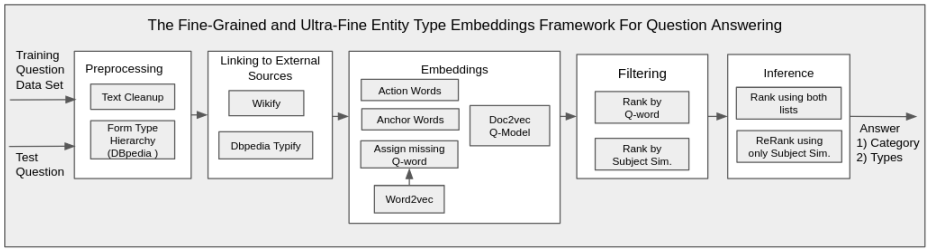

Fig. 1: The Fine and Ultra-Fine Entity Type Embeddings Framework.

Semantic Processing and Induced Type Hierarchy. The first step in the framework is the processing step which encompasses a pipeline constructed using Stanza [12] and an inducing of DBpedia type hierarchy. The pipeline includes a tokenizer, lemmatizer, POS tagger and dependency parser. We select a few syntactic and parts of speech (POS) tags for use in our filtering. From the dependency parse of a question we obtain root and subject/object (*nsub*, *nsubj:pass*, *csubj*, *csubj:pass* and *obj*) words, which contribute most to the answer type of a question (aka Action words). We select words from the Universal POS categories of ADP, AUX, DET and PART, which contribute most to the basic sentential structure of a question (aka Anchor words). Since DBpedia types form a hierarchical tree, we induce a hierarchy from the gold types for each question. A hierarchical path helps include any missing types in the gold labels.

Linking to External Sources. An important component of this framework is identifying entity mentions and replacing them with a type hierarchy. This normalizes questions into abstract forms aiding the Doc2vec model embed similar types of questions closer in the document embedding space. For identifying entity mentions, we collected n-grams (1 to 12) from question sentences and linked to two external sources: pretrained wikipedia2vec word embeddings [16] and a collection of 3.6 million DBpedia entity names with their associated types, collected from DBpedia using SPARQL queries. We performed wikification, the process of identifying entity mentions in text by checking against the titles of wikipedia entries [10]. We also performed *typification*, our novel contribution, similar to

wikification, where entity mentions with an associated type in the DBpedia hierarchy are normalized to a generic form. This normalization to a smaller set of types instead of a large number of noun forms helps reduce some of the data sparsity. For example, after typification, a question sentence "Who is the president of United States?" is transformed to "who is the _Thing_Agent_Person of _Thing_Place_PopulatedPlace_Country _Thing_Agent_Person ? where *president* and *united_states*, are replaced by their hierarchical DBpedia types. By inducing a hierarchy, we account for missing types and reduce the overall number of types.

Word Embeddings. Instead of using lexical understanding, we use similarity measures obtained from word embeddings trained on the questions, to aid in filtering. The answer category is highly dependent on the question words. For example, questions starting with a *Is* or *Does* always expect a *boolean* answer category, whereas, questions starting with *when* expect either a number or a date, a *literal* answer category. To reduce noise, we assign a single question word, a Q-word, to each question, if it falls into our predefined list of question words. About 10-15% of questions do not start with a question word. For these questions, we infer a Q-word using a Word2vec model we trained. The model was trained only on question sentences where the first words are question words. Inference with this model uses the entire question as the context and picks the question word such that the context and the question word have the highest cosine similarity. We measure the effectiveness of this inference, by obtaining the model's prediction accuracy on sentences which do not start with a question word but contain a question word in the sentence. The accuracy of predicting a question word contained in the sentence is 76%. The Q-word assigned to a question is used to filter and predict the answer category.

The answer type is highly dependent on the Action and Anchor words. We leverage the trained Word2vec model for Q-words to obtain word vectors for the Action and Anchor words. Not all Action and Anchor words will have word vectors because the model was only built with questions starting with a question word. In addition to the Anchor words belonging to the selected POS categories, we added two manually constructed word categories: a list of words that refer to a date and a list of words that refer to a number. We average the word vectors of Action and Anchor words to obtain a subject vector for the question. This subject vector aids in predicting the answer types.

Q-Word and Subject Similarity Filtering. Q-words and subject similarity are used for filtering the set of similar questions found by Doc2vec, for a test question, into two lists: questions with the same Q-word as the test question and questions with a high subject similarity. Subject similarity is the cosine similarity between the subject vector of the similar question and the test question weighted by a question sentence length measure. This measure assigns a higher weight to questions that are similar in length to the test question and assigns a lower weight to shorter and longer question sentences. After filtering, we attach the gold answer labels for the answer category, and the answer types to these two lists. The first list is sorted by the document similarity, and second list is sorted

by the length weighted document and subject similarity. For both the lists, we only consider the top 10% to reduce noise.

Inference. To infer answer category and types, the document vector for a given test question from the Doc2vec model is used to obtain similar questions from the embedding space. These are filtered to get the Q-word and subject similarity lists. The inferred category is the gold answer category of the top ranked question in the Q-word list. Inferred types are collected from the gold answer types of questions in the subject similarity list belonging to the inferred category. For Wikidata questions we collect up to 50 types, and for DBpedia we collect up to 10 types. For DBpedia types, we unroll the hierarchy, listing types in the higher levels only once. Reranking with only subject similarity is applied when there are no inferred types. This could happen when action words are used in a different context. For these questions, we change the inferred category to the gold answer type category of the top ranked question in the subject similarity list, and proceed to infer answer types based on the newly inferred category.

5 Experiments and Evaluation

The SMART task challenge consists of two separate datasets – one for assigning Wikidata type classes and another for assigning DBpedia type classes [8]; each of these datasets consists of a training set and a test set. The training set consists of natural language questions with their corresponding answer category and answer type. The number of training questions for Wikidata is 18,251 and the number for DBpedia is 17,571. The number of test questions for Wikidata was 4,571 and the number of test questions for DBpedia was 4,381.

We compared several methods to study the effects of various parameters on inference. For both datasets, we compared the results with and without the use of the external data sources. The baseline for comparison is our Doc2vec model without any additional filtering. For this baseline, both the Q-word and Subject similarity filtered lists are the same. We compared the performance of the filtering with three different settings, with and without using external source data. The three settings are: 1) using subject similarity calculated from average word vectors of the action words only, 2) using subject similarity calculated from average word vectors of the action words and the anchor words, and 3) re-adjusting the answer category when no answer types can be inferred.

Results. Accuracy is used for evaluating the answer category. MRR is used for evaluating answer types from the Wikidata classes. Lenient NDCG@5 and NDCG@10 with a linear decay [10] are used for evaluating the answer types from the DBpedia classes. Accuracy, MRR and NDCG values for the various settings for both the datasets are listed in Table 1.

Analysis - DBpedia Dataset. The dual filtering improved the accuracy of the model by 5% to 6%. It also doubled the NDCG@5 values and almost doubled

Experimental Settings	Wikidata		Dbpedia		
	Accuracy	MRR	Accuracy	NDCG@5	NDCG@10
No External Sources	.85	.26	.831	.275	.281
1) Subject Sim. using Action words	.85	.39	.883	.542	.519
2) Subj Sim. using Action & Anchor	.85	.39	.881	.535	.514
3) Rerank with Subject Sim.	.84	.39	.823	.523	.500
Using External Sources	.85	.26	.811	.266	.271
1) Subject Sim. using Action words	.85	.38	.870	.531	.508
2) Subj Sim. using Action & Anchor	.85	.39	.873	.527	.505
3) Rerank with Subject Sim.	.85	.40	.812	.517	.494

Table 1: Prediction Results for various experimental settings of the framework.

the NDCG@10 values, however, we are perplexed by the NDCG@10 values being lower than the NDCG@5 values. This shows that both our subject similarity and Q-word filtering are effective in identifying the correct answer category and types. Using Anchor words in the subject similarity calculation did not improve the model performance as expected, instead reduced the performance slightly. The could be because these words cause the model to overfit to the sentential structure. Unexpectedly, using external sources did not improve model performance. This could be attributed to sparsity created by long entity phrases. Readjusting the answer category using the subject similarity list also did not improve model performance. This implies the answer category inferred originally with the Q-word list is a better fit than the one found through the subject similarity and that Q-word is a good indicator for predicting answer category.

Analysis - Wikidata Dataset. The dual filtering improved model prediction for the answer types. The MRR for predicting answer types improved by 1.5 times. This shows that subject similarity filtering is effective in identifying the correct answer types. Using Anchor words in subject similarity did not improve model performance. This could be because the questions in this dataset are more of the factoid kind with a simpler sentence structure and the sentence weighting already contributed to finding similar sentential structures. External sources had a small effect on the MRR for type inference. Readjusting the category using subject similarity improved MRR for type inference slightly. Normalizing questions to abstract forms seemed to improve answer type inference.

6 Conclusions and Future Work

We show that paragraph vectors can be used effectively in Question Answering classification. Our results show that semantic modeling can add significant improvements when used in conjunction with low dimensional neural models. With the use of Word2vec, we show a collaborative approach to semantic modeling and model training. Future work will investigate how we can leverage Anchor words to learn sentential structures. In addition we will explore other ways of incorporating external sources to improve model performance.

References

1. Choi, E., Levy, O., Choi, Y., Zettlemoyer, L.: Ultra-fine entity typing. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 87–96 (2018)
2. Clark, K., Manning, C.D.: Improving coreference resolution by learning entity-level distributed representations. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics Volume 1. pp. 643–653 (2016)
3. Dai, A.M., Olah, C., Le, Q.V.: Document embedding with paragraph vectors. In: NIPS Deep Learning Workshop (2014)
4. Dong, L., Wei, F., Sun, H., Zhou, M., Xu, K.: A hybrid neural model for type classification of entity mentions. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
5. Hill, F., Cho, K., Korhonen, A.: Learning distributed representations of sentences from unlabelled data. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1367–1377 (2016)
6. Kim, H.K., Kim, H., Cho, S.: Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. Neurocomputing **266**, 336–352 (2017)
7. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International conference on machine learning. pp. 1188–1196 (2014)
8. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic web **6**(2) (2015)
9. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of the 19th International Conference on Computational Linguistics - Vol. 1. p. 1–7 (2002)
10. Mihalcea, R., Csomai, A.: Wikify! linking documents to encyclopedic knowledge. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management. p. 233–242. CIKM '07, New York, NY, USA (2007)
11. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
12. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A python natural language processing toolkit for many human languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 101–108. Online (Jul 2020)
13. 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. pp. 1045–1055 (2015)
14. Voorhees, E.M.: Overview of the trec 2001 question answering track. In: In Proceedings of the Tenth Text REtrieval Conference (TREC. pp. 42–51 (2001)
15. Wang, S., Tang, J., Aggarwal, C., Liu, H.: Linked document embedding for classification. In: Proceedings of the 25th ACM international on conference on information and knowledge management. pp. 115–124 (2016)
16. Yamada, I., Asai, A., Sakuma, J., Shindo, H., Takeda, H., Takefuji, Y., Matsumoto, Y.: Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia. arXiv preprint 1812.06280v3 (2020)
17. 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 (2016)