COMP90042 Project 2018: Question Answering

Muhammad Atif (924009)
Kaggle Name: Muhammad Atif - 924009

INTRODUCTION

In this project, given a set of questions and corresponding English documents about a subject, the challenge was to develop an automated system capable of finding answer from the specified document. Main components of the designed system included most relevant sentence retrieval using TF-IDF based scheme, named-entity recognition, and prediction of expected answer type from the way question is phrased.

CANDIDATE SENTENCE AND ANSWER RETRIEVAL

Sentences were pre-processed by removing punctuation, lower-casing, and stemming. For each question, all the sentences in the specified document were ranked according to their similarity with question using Gensim's BM25 algorithm. This algorithm was chosen because it is state-of-the-art method for document relevance using TF-IDF like scheme [1].

To select final answer, answer type of question was first predicted from the word types appearing in question. Starting with highest ranked sentence (from BM25), in an iterative way, each of the sentences were then searched one-by-one to find the expected answer tag. The search stopped when first such tag was found given that this tag term does not appear in the question text, and this was marked as answer.

EXPECTED ANSWER TYPE PREDICTION

(1) All the sentences in 'documents' data file were tagged using named entity tagger in spaCy package. After merging of similar tags, the employed named entity tagging resulted in 6 distinct named entity tags.

Following steps were performed on train dataset:

(2) Using sentence relevance algorithm, all sentences in the specified document were scored in terms of similarity with the question.

(3) The answer text was searched in each of the sentences starting with the highest scored sentence until the text was found in any sentence. Corresponding named entity tag was saved in the train set. 43% of the questions got tagged in first-pass.

(4) For remaining untagged answers, manual checking was done to find out why these were not getting tagged, and using a simple rules-based approach, described in the subsequent section, percentage of questions tagged increased to 69% after the second-pass.

(5) Manual inspection revealed that there were some instances where answer terms were being wrongly tagged. To address this, terms were re-tagged by voting. For e.g. term 'EINSTEIN' was 12 times tagged as 'PERSON' and four times as 'INSTITUTION'; since the most frequent tag was 'PERSON', 'EINSTEIN' was also re-tagged as 'PERSON' on the four occasions when it was mis-tagged as 'INSTITUTION'.

Data preparation for expected answer type prediction

(6) For each answer tag in train, a dictionary of of unigram, bigram, and trigram term frequencies was built from the corresponding question text. 13,672 untagged questions in train were ignored.

(7) Top 800 most frequent terms of each tag were selected as features which after removing duplicates, resulted in a final list of 1246 terms/features across all tags. Test set was prepared by extracting the same set of features in similar fashion.

Prediction

(8) To predict expected answer type from question, LighGBM classifier was used. This classifier is based on tree-based ensemble and has gained popularity in recent times on Kaggle competitions due to its good performance, low memory usage and faster training times [2]. Parameter tuning of the algorithm was done using 3-fold cross-validation.

DESIGN CONSIDEARTIONS AND ERROR ANALYSIS

To evaluate if a tweak in the approach improved system's performance, a strategy was formulated, referred as 'validation method' in this report, by randomly shuffling train examples and making 5 splits of train data, each

containing 1000 examples. Evaluation metric was checked for each of the splits, and if an improvement was observed in at least 3 of the splits, the tweak was considered an improvement and retained.

Design consideration # 1 – For answer type prediction using LightGBM classifier, decision on how many most frequent terms to retain as features was taken by measuring improvement in tagging accuracy given by LightGBM cross-validation method using 3 folds on train data. Accuracy is given by number of correctly tagged instances as a percentage of total instances. Algorithm was repeatedly trained using increased number of top features in each iteration until adding more features gave diminishing returns as depicted in Fig.1. Using this approach, top 800 features were selected in the final model, which gave tag classification accuracy of ~74%

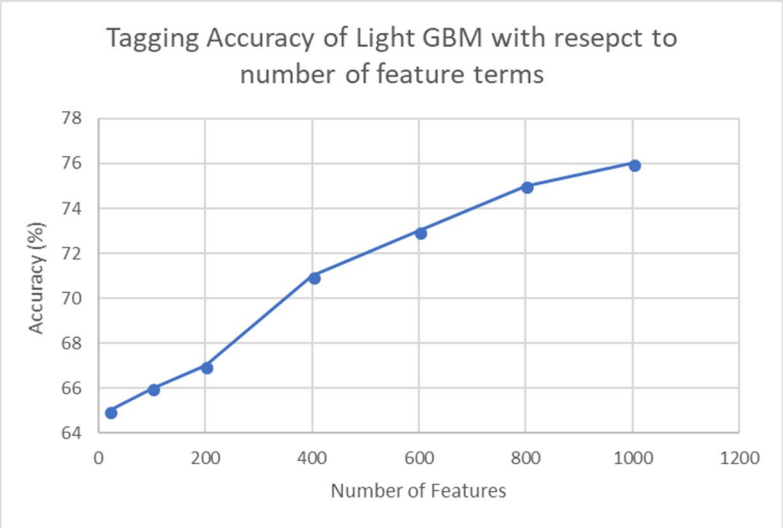

Fig. 1

Design consideration # 2 – NER tagger classified tags into 16 distinct categories. This was speculated to have an adverse effect on tag prediction accuracy because (1) Some of the tags had very few training examples (low evidence), and (2) Higher number of tags mean that there are more outcome classes to predict, making the task complex. To address this, it was noticed that some of the classes of tags were too fine-grained with subtle difference between them. Having such level of detail was clearly not required for the task at hand. Using intuition and repeatedly checking prediction results on 'validation method', following categories were merged:

'PERCENT', 'MONEY', 'QUANTITY', 'ORDINAL' -> 'NUMERIC' 'DATE', 'TIME' -> 'DATETIME'
'WORK_OF_ART', 'LAW', 'PRODUCT' -> 'DOCUMENT' 'ORG', 'GPE', 'NORP', 'LOC', 'FAC' -> 'INSTITUTION'

Design consideration # 3 – Number of questions for which the answers got tagged using exact string match of the answer term with the tag term was 18,458 out of a total of 43,379. This was clearly on a lower side. To further increase this number, untagged answers were manually checked, and following rules were applied which gave 29,707 tagged questions:

- Split the answer terms and tagged terms and match each term in both with each other. Example – '36,000' got matched with the tag '36,000 rial'
- Split the answer terms on special symbols and match first part of each term with tag terms. Example – '6999♠222332' got matched with the tag '6999'
- Remove space between first two tag terms and then match with answer terms. Example – '$140.4 billion' got matched with the tag '$ 140.4 billion'

More of such similar rules could have been crafted to further improve the tagging.

Design consideration # 4 – For sentence retrieval, first attempt was made using cosine similarity over a TF-IDF scheme in a vector space model using inverted index approach. Subsequent enhancements included addition of bigram, and trigram terms to TF-IDF's. However, the biggest jump here came from using the BM25 ranking function. To quantify gain, 'validation method' was used with sentence retrieval accuracy as the metric, which is defined as the number of instances where the selected sentence contained the answer as a percentage of total instances.

	Split 1	**Split 2**	**Split 3**	**Split 4**	**Split 5**
Unigram TF-IDF	28.6	31.6	28.8	27.9	29.1
Bigram TF-IDF	29.2	31.4	29.5	28.4	31.0
Trigram TF-IDF	28.6	31.6	28.5	28.0	30.6
BM25	34.3	35.1	36.1	36.8	36.0

Table 1. Sentence retrieval accuracy percentage on each of the splits

IMPROVEMENTS

Text pre-processing – Text-cleansing by removing gibberish characters and rectifying typos with the help of automated spelling corrections can help improve the quality of text. Pronoun-based transformation (replacing 'he/she/his/her' with the entity name for e.g.) based on preceding sentences can help, especially with who-type questions .

Sentence Retrieval – The major improvement area is in the document retrieval part of the system. Sentence retrieval accuracy over the train set currently stands out at 35% and can be improved considerably by employing smarter strategies. These include n-gram language models and/or the use of contextual word embeddings to better capture the context instead of a simple TF-IDF based approach which does not keep track of the order in which words appear. Lexical resources like wordnet to include word synonyms while computing sentence relevance along with query expansion techniques to rewrite queries is also worth experimenting with.

Answer Retrieval – Another enhancement in the answer retrieval part could come from enacting more thorough rules such as if the next term of the predicted answer in the sentence is a single character, make it part of the answer as well.

REFERENCES

[1] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, *3*(4), 333-389.

[2] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (pp. 3149-3157).