Question Answering using Random Forests

Zhao Li (Prid number: 1606786)

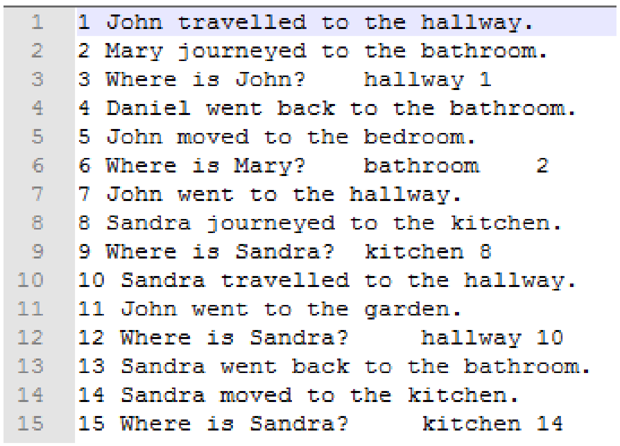
Journal format: IEEE Transactions on Knowledge and Data Engineering

Abstract—Question answering (QA) problem is a typical problem in natural language processing. There exists many approaches to dispose QA problem such as dynamic memory networks (DMN) and Memory network (MemNN). In this paper, a new model of processing QA problem is illustrated, which applies random forest classifier to test the accuracy of the training data, within the targeted datasets which are Facebook bABi tasks. It is found that the results of the new model are worse than the two other methods. In order to improve the model, word vectors are expected to be utilized in the future work, and the optimization on the random forest is also considered.

Index Terms—Question answering, random forest, bABi tasks

1. Introduction

This project mainly analyses the creation of a random forests classifier which is based on the data of bABi tasks. bABi tasks are a set of stories with specific questions and answers which can be downloaded from Facebook.[1] The core task in this project is to possess the data to solve the question answering (QA) problems. QA can be concluded as a complex natural language processing task. With comprehension to the content of the text, the QA is able to produce the correct answers through reasoning over related facts. An example of the bABi tasks is shown in Fig.1.



```
1 John travelled to the hallway.
2 Mary journeyed to the bathroom.
3 Where is John?    hallway 1
4 Daniel went back to the bathroom.
5 John moved to the bedroom.
6 Where is Mary?    bathroom    2
7 John went to the hallway.
8 Sandra journeyed to the kitchen.
9 Where is Sandra?  kitchen 8
10 Sandra travelled to the hallway.
11 John went to the garden.
12 Where is Sandra?     hallway 10
13 Sandra went back to the bathroom.
14 Sandra moved to the kitchen.
15 Where is Sandra?     kitchen 14
```

Fig.1 Example of bABi tasks

There are plenty of methods to possess the data, and the random forests classifier is implemented in this project. A random forest is a type of estimator which is composed of many decision tree classifiers. The main application of it is to maintain various sub-samples of the dataset to strength the accuracy of the prediction.

The targeted data is possessed firstly. All the stories are retrieved in the files, and each stories are divided into three parts including content of stories, questions and answers. Subsequently, every word in three parts is converted into a unique number respectively. Three specified arrays are then obtained. In order to apply the arrays to the random forest classifier, the arrays are shaped into two new arrays, which are an array composed of stories and questions, and an array including all the corresponding answers. Finally, after inputting the two arrays into the random forests classifier, it generates a mean accuracy by comparing the training data and testing data.

2. Background

Normally, question answering problems are discussed in many fields with the rapid development of complexity of applications. Without doubt, there exists different approaches to solve the problem of question answering. With respect to the possessing of bABi tasks, two related studies are referred in this project.

2.1 Dynamic Memory Networks

Dynamic memory networks (DMN) for natural language processing dataset is a neural network architecture. The function of it is possessing the input data of stories and questions, then producing relevant answer through a special module named episodic memories. DMN can be applied in the problems of sequence tagging tasks, classifications, sequence-to-sequence tasks and QA. For the QA problems, the DMN can calculate word representations for the input stories and question, and then an iterative attention action would be started to track the inputs and retrieves relevant facts. After that, the episodic memory module would analyse the retrieved facts and generate a word representation of the facts. The answer module is able to receive the generated representation, and then produce the corresponding answer. [2]

The DMN is divided into four modules, which are input module, question module, episodic memory module and the answer module. The input module has the function to convert original input texts into to vector representations. As for the targeted data, the original inputs are stories with long sentences. The Question module can encode the input questions into vectors representations by the same method as the input module. The production of question module is applied in episodic memory module to be iterated. The episodic memory module is the core part of DMN, which retrieves new information iteratively based on the input vector representations with considering the question vector representations. In each iteration, it would produce a new memory vector representation. The answer module is the module to generate the corresponding answers according to the final memory vector representation. In order to encode the original sequences, a recurrent neural network is applied. As for the build of the DMN, the researchers implemented a method named gated recurrent network (GRU). An overview of the DMN module is shown in Fig.2. The arrows in this figure represent the communications among the four modules. [2]

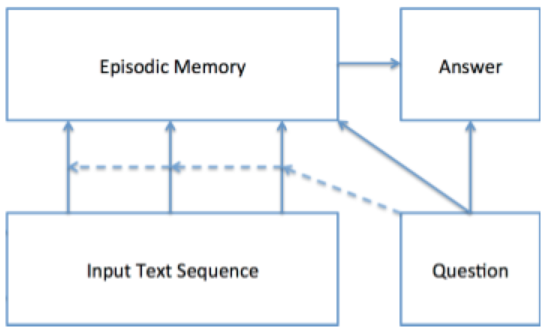


Fig.2 Overview of DMN modules

The researchers conduct an experiment focusing on the QA problem which is related to this project. The bABi tasks are trained in DMN modules and a set of accuracy is obtained. The results are compared with the results obtained from MemNN method. It is found that the DMN is not better than the MemNN, and the reason for that might be MemNN using n-gram vectors and sequence position features. [2] In this project, the generated accuracy would be compared with the above methods in discussion part.

2.2 End-To-End Memory Networks

End-to-end memory network (MemN2N) is a neutral network developed over MemNN method. It strengths the ability to adapt different tasks, and needs less supervision because it is trained end-to-end. The main concept of MemN2N is to build a model that contains a memory to store the inputs representations, a query q and answers a as the output. A dictionary including all the unique words is built previously. The inputs, q and a are composed of words from the dictionary. The memory can provide a continuous representation for the inputs and q. According to the continuous representation, the output a is generated finally. The advantage of this approach is it can detect errors through backpropagation when the training is processing. [3]

The experiment implemented two stages: single layer and multiple layers. And softmax is applied as the main tool to possess the vectors. The results of the experiments are compared with several other methods such as MemNN, RNNs and LSTMs. It is concluded that MemN2N can be applied in more tasks than MemNN because of the lack of supervision of supporting facts in MemNN. With respect to the RNNs and LSTMs, the MemN2N is simpler to conduct. [3]

MemN2N could be a suitable comparison to the random forest classifier which is the method used in this project. It can play a significantly important role in improving this project.

3. METHODOLOGY

The data possessed in this project is the bABi tasks which are consisted of 20 sets of tasks. In each task, there are 1000 stories with questions and corresponding answers. The tasks require deduction and comprehension for the models which possess them. In addition, the each task has two kind of files including training data and testing data respectively. The training data is used to be trained for the model, and the testing data will be applied to be compared with the result obtained from the training data. A brief structure of this project is illustrated in Fig.3

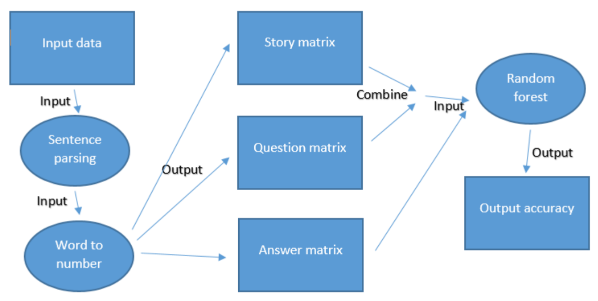


Fig.3 Structure of the created model

The approach in this project can be divided into three parts: sentence parsing which transforms the texts into appropriate lists; word to number which converts the lists of words into matrixes of numbers; random forest classifier which possesses the shaped matrixes to obtain the accuracy. The first two parts are based on the method applied in an example provided by the Keras documentation on github [4].

3.1 Sentence parsing

The first challenge in this project is to parse the sentences in the data. The origin text files should be transformed into appropriate a list firstly. For the reason that there are many useless numbers and punctuations, a parsing function is used as a filter to split and modify the text into a list. Given the number of the stories is N, and the statement of each story is s, question is q and answer is a. The function can return a large 3-dimension list which contains N lists like $[s,q,a]$. N, q and a are also lists which are composed of elements represented by every single word.

3.2 Word to number

The next stage of the project is converting the list obtained from the sentence parsing. In this project, a dictionary is applied to store the whole unique words and corresponding numbers. Firstly, a vocabulary list is created, and with the input of the obtained list, all the unique words are put into the list. As a result of this, the vocabulary list contains all the words appeared in the obtained list, and each word is different from others. Subsequently, the vocabulary list is imported into the dictionary. With the enumerating function, whenever a new word is read, the dictionary signs it a specified number. Therefore, every single word in the vocabulary list has a corresponding number.

A function is defined to achieve the numbering. In this function, story s, question q and answer a are converted into numbers respectively. In addition, the maximum lengths of each story and question (answer is a single word) which are applied as the condition of the function are calculated previously. Three matrixes: story matrix, question matrix and answer matrix are generated finally. Story matrix is a 2-dimension (N*maximum length of story) matrix, and each row includes the number representations of each story. Similarly, the question matrix is a 2-dimension (N*maximum length of question) matrix which contains rows of corresponding question-numbers. The answer matrix is a 1-dimension array including the numbers which represents the answers.

3.3 Random forest classifier

The random forest classifier is implemented in this project. Random forest is a combination of numbers of decision trees. With inputting the datasets, each decision tree will produce a predictive result according to the features of the datasets, and then the random forest will select the result which appears most frequently as the final result [5]. Random forest classifier is a classifier which can create a random forest based on the input data.

In order to import the targeted dataset into the classifier, the dataset needs to be shaped properly. To begin with, the training dataset is supposed to be fited, which requires two inputs X and y. X is the training dataset which is a matrix, and y contains the targeted values which is also a matrix. Then the testing data is possessed into the same format as the training dataset. Through the score application in the classifier, a mean accuracy will be produced by applying the testing data to test the training data. As for this project, X is a matrix including story and question, and y is the answer matrix.

4. EXPERIMENT

The implementation of the methodology is divided into two steps. The first step is to process the origin data through sentence parsing and word to number, and the second step is to shape the obtained matrixes and then import them into the random forest classifier.

To simplify the process, a single task file is implemented in the beginning. Three matrixes are obtained after the data processing. Then the story matrix and question matrix are concatenated as the input X of the classifier. After importing the X and y which is the answer matrix, the classifier generates the mean accuracy of the first type of task.

The aim of the project is to analyse all the tasks in bABi tasks. Therefore, a loop is created to read all the task files. The classifier then produces accuracies for each type of task. In addition, in order to test the accuracy of the whole text, all the text files are combined into one text file. The experiment on this unique text file is conducted. Finally, a mean accuracy which evaluates the whole text is generated. Table 1 shows all the results.

Table 1 Accuracy of each task and whole text

Task	Accuracy
1_single-supporting-fact	38.10%
2_two-supporting-facts	19.60%
3_three-supporting-facts	15.80%
4_two-arg-relations	67.70%
5_three-arg-relations	42.40%
6_yes-no-questions	53.50%
7_counting	70.50%
8_lists-sets	53.00%
9_simple-negation	63.10%
10_indefinite-knowledge	49.60%
11_basic-coreference	53.30%
12_conjunction	48.00%
13_compound-coreference	76.80%
14_time-reasoning	21.00%
15_basic-deduction	41.30%
16_basic-induction	39.90%
17_positional-reasoning	57.90%
18_size-reasoning	78.30%
19_path-finding	10.40%
20_agents-motivations	92.30%
Whole tasks	42.20%

5. DISCUSSION

The evaluation of the results can be achieved by comparing them with the results in DMN and MemNN. The following table shows the comparison [2].

Table 2 Comparison of three methods

Task	Random Forests	MemNN	DMN
1_single-supporting-fact	38.10%	100.00%	100.00 %
2_two-supporting-facts	19.60%	100.00%	98.20%
3_three-supporting-facts	15.80%	100.00%	95.20%
4_two-arg-relations	67.70%	100.00%	100.00 %
5_three-arg-relations	42.40%	98.00%	99.30%
6_yes-no-questions	53.50%	100.00%	100.00 %
7_counting	70.50%	85.00%	96.90%
8_lists-sets	53.00%	91.00%	96.50%
9_simple-negation	63.10%	100.00%	100.00 %
10_indefinite-knowledge	49.60%	98.00%	97.50%
11_basic-coreference	53.30%	100.00%	99.90%
12_conjunction	48.00%	100.00%	100.00 %
13_compound -coreference	76.80%	100.00%	99.80%
14_time-reasoning	21.00%	99.00%	100.00 %
15_basic-deduction	41.30%	100.00%	100.00 %
16_basic-induction	39.90%	100.00%	99.40%
17_positional-reasoning	57.90%	65.00%	59.60%
18_size-reasoning	78.30%	95.00%	95.30%
19_path-finding	10.40%	36.00%	34.50%
20_agents-motivations	92.30%	100.00%	100.00 %
Mean accuracy (%)	53.88%	93.30%	93.60%

As it can be seen in Table 2, the accuracies obtained from random forest classifier are worse than the results of DMN and MemNN. Despite the imperfections of the methodology, the random forest might not be a competitive approach to process the QA problems. Compared to DMN and MemNN, random forest is a steady model which is not able to learn. The DMN has the ability to provide continuous information in the memory module. With respect to the random forest, it needs to be rebuilt when the input data is changed which will increase the complexity of the experiment.

Additionally, in this project, the words in texts are only converted into corresponding numbers which is possibly reduce the accuracy. In DMN, the words are pre-trained by the GloVe which generates a number of vectors for each word. GloVe is a learning algorithm which can convert words into vector representations. The main advantage of GloVe is that it provides an overview of relationships among the words, which will increase the ability to reason over the corpus. It calculates the cosine similarity between two vectors that represents the linguistic or semantic similarity of two words [6]. The DMN takes advantage of the word vectors to produce more accurate results. In the early stage of the project, GloVe was applied. However, after a set of word vectors were obtained, it was found that it was difficult to correspond the vectors to each story. Therefore, GloVe was not considered.

6. CONCLUTION

This project creates a model to solve the QA problem based on the random forest. The evaluation of this model is not ideal, and this is possibly because of the limitation of the random forest classifier and imperfection of word processing.

In order to obtain an optimal model, the future work can focus on the application of word vectors. With word vectors, the relationship between two words will be clarified, which helps to comprehend the meaning of the text. Additionally, exploring the method to develop the random forest model is essential as well. An assumption is that make the model learn during training data, which means whenever a new set of data is imported, the model can add the data into the previous random forest and rebuilt a new random forest automatically.

REFERENCES

[1] J.Weston, *et al*, "Towards ai-complete question answering: A set of prerequisite toy tasks." New York, USA, *arXiv* preprint arXiv: 1502.05698. Dec. 2015

[2] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, “Ask Me Anything: Dynamic Memory Networks for Natural Language Processing,” *arXiv*, pp. 1–10, 2015.

[3] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, “End-To-End Memory Networks,” *Adv. Neural Inf. Process. Syst. (NIPS ’15)*, pp. 1–11, 2015.

[4] Keras Documentation. *Keras examples* [Online]. Available: https://github.com/fchollet/keras/tree/master/examples

[5] Breiman, “Random Forests”, *Machine Learning*, 45(1), pp. 5-32, 2001.

[6] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” *Proc. 2014 Conf. Empir. Methods Nat. Lang. Process.*, pp. 1532–1543, 2014.