COMP90042 Project 2018: Question Answering – Team: Xudong & Winn

Winn Chow <winnchow> (38315) and Xudong Han <xudongh1 > (881045)

Introduction

An information retrieval (IR) based factoid QA system is developed for the challenge. It is designed based on the IR-based factoid QA system framework outlined in [6]. spaCy is used for NLP processing as it is highly accurate and fast in comparison with other NLP libraries such as CoreNLP [12]. Scikit-learn is used for machine learning. WordNet is used for finding word sense relationships. A Kaggle public F1 score of 0.25609 is achieved.

Method Description

The system is information retrieval (IR) based since an analysis on the training data shows that the questions are primarily factoid questions such as "What was X?" and "Who was X?". The system has three modules: Question Processing, Passage Retrieval and Answer Processing [6].

1. Question Processing

Firstly, the system will process the question to detect its answer type. A taxonomy of a total of 20 answer types is used. 18 of them are entity types from spaCy, such as PERSON, DATE and LOC [13], and two others are UNKNOWN and HYPERNYM.

Each question in the training and development set is annotated with an answer type using Named Entity Recognition (NER). NER recognizes whether the gold standard answer is a named entity in the answer paragraph. If it is, the recognized entity type is annotated as the answer type. Otherwise, UNKNOWN or HYPERNYM is annotated. HYPERNYM is identified using hand-written rules and will be discussed later. Once the training and development set are annotated, the answer type detection of the system can be trained and evaluated.

Answer type detection is performed using both (1) Hand-written rules and (2) Machine learning. Hand-written rules have higher precision but lower recall [6]. They are firstly used for answer type detection. A total of about 90 string matching-based rules are used to detect named entity questions, such as "what is the name" is classified as a PERSON or ORG type question. One non-named entity type, HYPERNYM, is added at a later stage. The idea is that there are several "What X" questions that X is a hypernym of the answer, for instance "What color …?" and the answer is "green". A list of words (X), such as color, shape and animal, is added to the rules to identify HYPERNYM questions.

If the rules fail to detect the answer type of a question, machine learning classifier is secondly used. A support vector machine classifier (SVC) with RBF kernel is used. It is a common choice for answer type detection [7,11]. A set of features is selected to train the classifier with reference to [6,7,10,11], including "wh" word, bi-gram of "wh" word, part-of-speech (POS) tag of "wh" word and its bi-gram, and POS tag of the root token. POS and dependency parsing are used to extract the features from a question.

Secondly, the system will identify query keywords from the question for querying the documents. The question is first tokenized and then lemmatized. Stop words and punctuations are also removed. The question is then stored as a bag-of-words (BoW) of query keywords.

2. Passage Retrieval

In this module, the system will retrieve the top-k paragraphs that best match the query keywords (BoW of the question). Each paragraph is turned into a BoW using the same normalization steps as those for the question. TF-IDF is used to score and rank the paragraphs. The top-k paragraphs are returned. k = 3 is selected based on the evaluation result of the development set.

3. Answer Processing

   a) Named entity questions

If the answer type detection classifies the question as a named entity question with a NER label. The system will run NER parsing on the top-k paragraphs and identify all the candidate answers. If at most one candidate answer is found in each paragraph, the candidate answer in the highest ranked paragraph is returned as the answer. If no candidate answer is found, the question is passed on as an UNKNOWN question to be further answered. If more than one candidate answers are found in one paragraph, all candidate answers are ranked using TF-IDF as follows. Firstly, each paragraph with N candidate answers are segmented into N units. All words starting from the beginning of the paragraph until the middle word between the 1st candidate answer and 2nd candidate answer will be segmented as 1st unit and assigned to the 1st candidate answer. Starting from the end of the 1st unit until the middle word between the 2nd candidate answer and next candidate answer or the end of the paragraph if there is no more candidate answer will be 2nd unit. The segmentation continues until reaching the end of the paragraph. Secondly, TF-IDF is applied to rank each candidate answer and the highest ranked candidate answer is returned as the answer.

   b) HYPERNYM questions

If it is a HYPERNYM question, it is of the form "What X" and X is the hypernym. Firstly, X is extracted from the question. Secondly, the top-k paragraphs are segmented into sentences. TF-IDF is used to rank the sentences. Starting from the highest ranked sentence, each normalized non-stop word in the sentence is checked whether it is a hyponym of X using WordNet. If a hyponym is found, it is returned as the answer. If no answer is found, the question is passed on as an UNKNOWN question to be further answered.

c) UNKNOWN questions

The question is answered using different methods in the following order.

a. Subject – Verb – Object

The subject and verb of the question are extracted. The idea is that if the subject and verb of the question appear in a sentence of the top-k paragraphs, the answer is likely to be the object of the sentence. Firstly, the top-k paragraphs are segmented into sentences. The sentence with the longest sequence of question words is chosen. Secondly, the subject and verb of the question are matched with those in the sentence using dependency parsing. If they are the same, the object of the sentence is returned as the answer.

b. "What X" or "What X of Y"

The question is checked whether it matches the pattern of "What X" or "What X of Y". The idea is that there are questions of "What X" or "What X of Y" that X or Y either appears in the answer or is located very close to the answer, for instance, "What law …?" and the answer is "stefan–boltzmann law" and "What kind of prediction …?" and the answer is "accurate prediction". Firstly, X and Y are extracted from the question. Secondly, the top-k paragraphs are segmented into sentences. The longest sequence of question words is used to rank the sentences as before. All noun chunks, base noun phrases, are extracted from the highest ranked sentence using spaCy. If X or Y appears in a noun chunk, the noun chunk together with its context (one noun chunk before it and two noun chunks after it) is returned as the answer.

c. "what" and all other questions

For all other questions, the highest ranked sentence is located using the longest sequence of question words. The question is checked whether "what" appears in the question. If "what" is not found, all the noun chunks of the highest ranked sentence are returned as the answer. If "what" is found, the tokens with a dependency tag of root, nsubj and nsubjpass are identified from the question using dependency parsing. The answer is expected to locate near the identified tokens. The highest ranked sentence is tokenized and searched for the identified tokens. If an identified token is found, the identified token with its context (2 tokens before and 8 tokens after) is returned as the answer.

Discussion of Results

1. Question Processing

Using NER for answer type annotation of the development set, 67% of the questions are annotated with an entity type and 33% are annotated as UNKOWN.

Questions annotated with an entity type	Questions annotated with an UNKNOWN type
1867 (60%)	1233 (40%)

Table 1 Answer type annotation in the development set

Hand-written rules have slightly better precision of 61% than machine learning classifier of 58%.

Answer Type Detection	Coverage	Precision	F1
Hand-written rules	43%	61%	0.505
Machine learning classifier	100%	58%	0.551
Both	100%	59%	0.530

Table 2 Answer Type Detection

The machine learning answer type classifier is trained on the training set. The development set is used to evaluate the performance of the classifier with different C and gamma (parameters of SVC). The best F1 score is obtained with C = 3 and gamma = 0.3. A random forest classifier is also tested and found to give similar results.

C	Gamma	F1
0.5, 1, 3, 10	0.3	0.52, 0.54, 0.55, 0.54
3	0.1, 0.3, 1, 10	0.54, 0.55, 0.43, 0.16

Table 3 Support Vector Machine classifier with different C and Gamma on the development set

Error analysis and enhancement:

- As Hand-written rules have higher precision, more rules are added. The coverage of hand-written rules increases from 34% to 43% and precision remains at 61%.
- SpaCy has a total of 18 entity types. It is found that answer type detection cannot accurately differentiate between some similar types such as PERSON and ORG. It is decided to combine some types in answer processing. For example, PERSON and ORG are combined into a single entity type.

- HYPERNYM type is added after reviewing the error in answer type detection. It is identified that "What X" with X being the hypernym of the answer is a high precision rule for some words (X) such as color and shape.

2. Passage Retrieval

The top-k paragraph retrieval achieves 94% accuracy with k = 3. As k = 5 gives only 2% higher accuracy, k = 3 is used.

Top-k paragraphs	Accuracy
k = 1, 3, 5	78%, 94%, 96%

Table 4 Top-k paragraph retrieval on the development set

Error analysis and enhancement:

- Several other methods are tried to enhance the top-k paragraph retrieval performance. (1) doc2vec from gensim [3] using sentence embedding [9] to calculate the similarity of the question with each paragraph. It is found that as the words in the question are mostly the same as the words in the answer paragraph, doc2vec performs worse than TF-IDF as word sense similarity can cause more ambiguity. The accuracy is less than 90% for k = 3. (2) Selecting the paragraphs with the longest sequence of question words [6]. The performance is found to be very similar to using TF-IDF with 92% accuracy for k = 3. Word ordering may not be important for passage retrieval.

3. Answer Processing

Named Entity Questions	UNKNOWN Questions	Total
F1 = 0.271	F1 = 0.125	F1 = 0.212

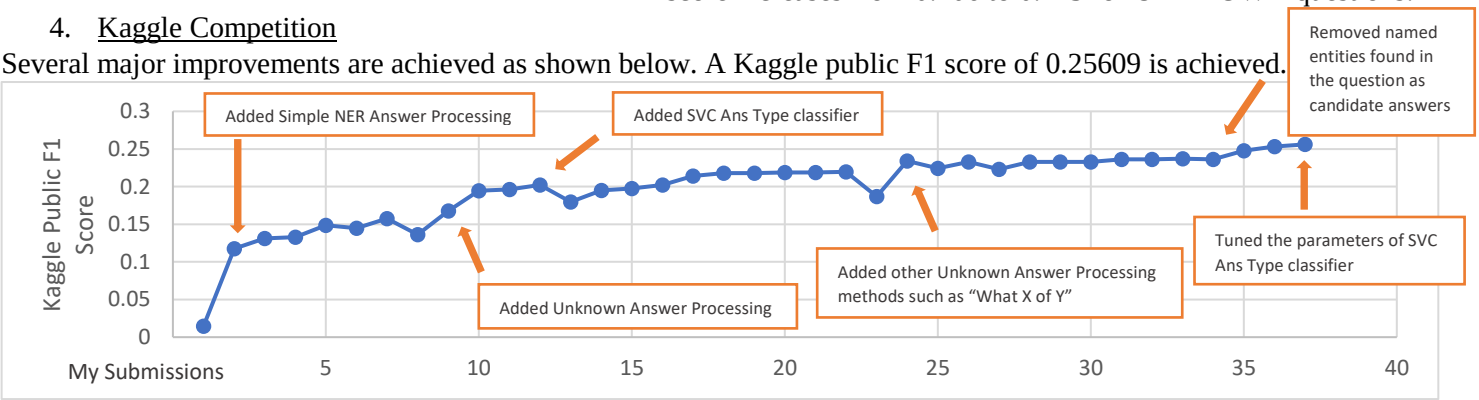
Table 5 Performance on Answer Processing

Error analysis and enhancement:

- As TF-IDF is used to rank the candidate answers based on counts, if a candidate answer (named entity) also appears in the question, it will be ranked high but almost always be the wrong answer. For instance, "Who is chairman of Bank of America?" and the answer is definitely not "Bank of America". Hence, candidate answers that appear in the question will be dropped. F1 score increases from 0.262 to 0.271 on the development set. Kaggle public F1 score increases from 0.24763 to 0.25322.
- A total of 10 HYPERNYM questions are identified in the development set. The system can achieve 50% accuracy on the questions.
- At first, only the simple method of returning all noun chunks from the highest ranked sentence is used to find the answer for UNKNOWN questions. After reviewing the errors, other methods are added such as "What X" or "What X of Y" and "stand for" or "refer to". F1 score increases from 0.106 to 0.125 for UNKNOWN questions.

4. Kaggle Competition

Several major improvements are achieved as shown below. A Kaggle public F1 score of 0.25609 is achieved.



Conclusion

An information retrieval (IR) based factoid QA system is developed. A F1 score of 0.212 is achieved on the development set. A Kaggle public F1 score of 0.25609 is achieved. The system has been iteratively improved with error analysis and enhancements, improving Kaggle public F1 score from 0.11760 to 0.25609 (118% increase). A range of NLP techniques have been effectively applied such as POS parsing, dependency parsing, named entity recognition and TF-IDF. The accuracy of top-k paragraph retrieval is 94% with k = 3. Answer type detection achieves a F1 score of 0.530.

Future work

- The accuracy of our answer type detection is low (59%) in comparison with the state of the art system that can achieve about 97% accuracy [4]. [1] suggests that an answer type detection classifier using SVC with a tree-kernel can achieve 90% accuracy. A tree-kernel can be added to our SVC classifier.
- Our system assumes that answers are named entities or noun chunks. However, named entities and noun chunks cover only 58% of the answers. Hence, other methods can be developed to locate the other 32% of the answers.
- This system is IR-based. Knowledge-based methods using relation extraction can be added [14].
- The current state of the art QA system is built using dynamic memory network (DMN) that can achieve 93.6% accuracy on the bAbI dataset [2,8]. Our QA system can be re-developed using DMN in reference to [5,15]. An initial running of DrQA from Facebook research achieves a F1 score of over 0.30 on the development set [15].

References

[1] Michael Collins and Nigel Duffy. 2001. Convolution Kernels for Natural Language. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic* (NIPS'01), 625–632. Retrieved May 22, 2018 from http://dl.acm.org/citation.cfm?id=2980539.2980621

[2] Facebook. bAbI. *Facebook Research*. Retrieved May 21, 2018 from https://research.fb.com/downloads/babi

[3] Gensim. gensim: topic modelling for humans. Retrieved May 21, 2018 from https://radimrehurek.com/gensim/models/doc2vec.html

[4] Madabushi Tayyar Harish and Mark Lee. 2016. High Accuracy Rule-based Question Classification using Question Syntax and Semantics - Semantic Scholar. Retrieved May 21, 2018 from /paper/High-Accuracy-Rule-based-Question-Classification-Madabushi-Lee/12182ddbba1edd62050c4cc18c0f0662bdfde2b9

[5] Steven Hewitt. 2017. Question answering with TensorFlow. *O'Reilly Media*. Retrieved May 21, 2018 from https://www.oreilly.com/ideas/question-answering-with-tensorflow

[6] Dan Jurafsky and James H. Martin. Speech and Language Processing (3rd ed. draft). Retrieved May 20, 2018 from https://web.stanford.edu/~jurafsky/slp3/

[7] Shirish Kadam. 2017. NLP: Question Classification using Support Vector Machines [spacy][scikit-learn][pandas]. *SHIRISH KADAM*. Retrieved May 20, 2018 from https://shirishkadam.com/2017/07/03/nlp-question-classification-using-support-vector-machines-spacyscikit-learnpandas/

[8] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2015. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. *ArXiv150607285 Cs* (June 2015). Retrieved May 21, 2018 from http://arxiv.org/abs/1506.07285

[9] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *ArXiv14054053 Cs* (May 2014). Retrieved May 21, 2018 from http://arxiv.org/abs/1405.4053

[10] Xin Li and Dan Roth. 2006. Learning Question Classifiers: The Role of Semantic Information. *Nat Lang Eng* 12, 3 (September 2006), 229–249. DOI:https://doi.org/10.1017/S1351324905003955

[11] Donald Metzler and W. Bruce Croft. 2005. Analysis of Statistical Question Classification for Fact-Based Questions. *Inf. Retr.* 8, 3 (January 2005), 481–504. DOI:https://doi.org/10.1007/s10791-005-6995-3

[12] spaCy. spaCy · Industrial-strength Natural Language Processing in Python. Retrieved May 20, 2018 from https://spacy.io/

[13] spaCy. Annotation Specifications · spaCy API Documentation. Retrieved May 20, 2018 from https://spacy.io/api/annotation

[14] Stanford University. The Stanford Natural Language Processing Group - Stanford Open Information Extraction. Retrieved May 22, 2018 from https://nlp.stanford.edu/software/openie.html

[15] 2018. *DrQA: Reading Wikipedia to Answer Open-Domain Questions*. facebookresearch. Retrieved May 21, 2018 from https://github.com/facebookresearch/DrQA