Automatic Question Answering System

Kazi Abir Adnan
Student ID: 90406
kadnan@student.unimelb.edu.au
Kaggle id: kadnan

Daniel Gil
Student ID: 905923
gild@student.unimelb.edu.au
Kaggle id: gild2018

1) Introduction

Question Answering systems are designed to automatically predict an answer to a question that is formulated in natural language. To achieve this goal, the system have as an input a question and a set of documents where it can extract the answer.

2) Motivation and background

The main motivation to build the system is to obtain the best answer possible and accurate enough to score higher than a heuristic answer output. This was the final project for Web Search and Text Analysis course (COMP90042) of the Semester 1 2018 at the University of Melbourne .

We have chosen IR and IE based methods to learn the model of question answering from the knowledge base. in the Figure 1 we present the general process to build the system.

3) Methodology

The QA system was built according to components defined from the process described in figure 1. The general idea is highlighted here in this section. First data preparation is performed to extract a knowledge representation from datasets given. This includes Parts of Speech (POS), Named Entity, Keywords, document-term frequency and so on. The following task is to process the question along with the given answer sentence for the system to detect the answer type requested. This followed by an Information Retrieval component which performs the query built in the previous step and finally, an answer processing component where the system ranked the possible sentences where the answer could be found and extract the answer from the most likely sentence.

a) Question Processing

This component uses supervised machine learning model to detect the answer type of the question. The answer type is basically the Named Entity.

Answer Type Detection: the output of this component is to determine the answer type. A set of labels were previously defined to establish an answer type taxonomy based on coarse-grained entities.

Answer type detection is performed using multi-class classifiers described in table 1:

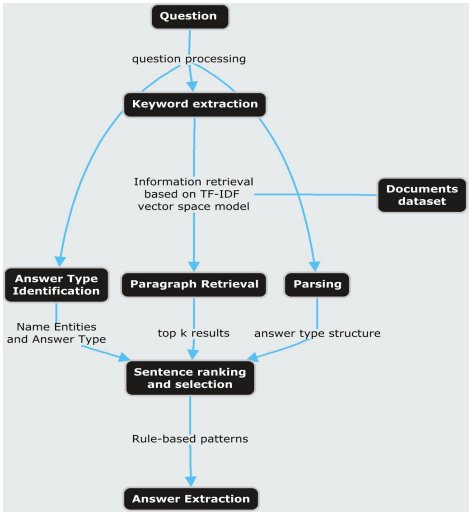

Figure 1: System components

Model	Features	Status
RandomForest	-BOW (Question and answers) -Question type	Active
MultinomialNB	-BOW (Question and answers) -Question type	Inactive

Table 1. Models for question classification

Query formulation: The goal of this component is to prepare a query to use with the Information Retrieval component. Query formulation component is used to query the vector space model, a basic query was built using question keywords as lemmatized non-stop words tokens in the question. No query expansion was needed to built as per the results were satisfactory for paragraph retrieval as shown in section 6.

b) Information Retrieval

The retrieval method is consist of two components. The paragraph retrieval and the candidate sentence retrieval. Each method is briefly mentioned below.

Passage Retrieval: The goal of this component is to obtain a set of paragraphs (documents) where potentially the system will find the answer. The paragraph retrieval is performed using TF-IDF to build a Vector Space Model over the documents dataset containing a list of paragraphs for every document id in the corpus. When querying the Information Retrieval component the system uses an inverted index built from the vector space model and returns the *top K* (k=5) paragraphs as ranking. Okapi

BM25 [3] was tried briefly but as results were not validated completely is proposed as future work.

Sentence Retrieval: Since the paragraphs contain a considerable amount of data, the goal of the sentence retrieval component is to extract the most relevant sentences from all top k paragraphs retrieved. A score is used to calculate the relevance of the sentence and the question:

$$relevance = common\ keywords + common\ entities + exact\ sequence - proximity + bigram\ overlap$$

The system use the highest scored sentence as the most probable candidate to find the answer. Two classifiers were built based on this features but with without good results, therefore, models were removed.

- Support Vector Machines: A binary classifier to predict a sentence as probable or not probable to be candidate to contain the answer.
- Regression Forest Regressor: Predicts a continuous variable score given the features above.

c) Answer processing

Answer-type pattern extraction:The answer extraction component was built to find the answer in the sentence selected. The technique used is pattern extraction based on the predicted answer type given a question.

An exploration of the development dataset give us some clues for patterns to be used, as a result, table 2 presents the rules established for answer extraction:

Rule	Description
First Answer-Type match	If more than one (1) answer type matches in the sentence. The system extracts the first found if the first entity found in the sentence matches with the label.
More than 1 answer-type match and the match was not with the first entity in the question	For this rule, all matches are collected and the answer will be predicted as all the entities found separated by a space.
No entities of answer-type found in the sentence	Use of POS tags in common (question-answer sentence) retrieving only Nouns and Numbers.
No entities and no POS in common between question and answer sentence	Retrieve unique keywords in common except for stop words
No entities, no POS and no keywords in common	Answer will be predicted as blank

Table 2. Rules for answer extraction

4) Experimental Results

The QA system has 4 core components and the performance of each component has been evaluated using the development dataset. It is a cascaded system where each component's performance impacts the next one.

Table 3 shows the number of samples used to train and evaluate the model. The system extracted successfully Named Entities of given answer for the mentioned samples and used them to train the Question Classifier.

Dataset	Sample Size
Training Dataset	29,174
Development Dataset	1,969

Table 3. Sample size for training and development dataset.

a) Question Classification

Question classification results are shown in table 4. The random forest classifier performed better to predict the labels using keywords as features. Particularly labels related to object descriptions were difficult for the system to train and predict 2 types of features were evaluated, first common keywords of question and all answer sentences in training set, secondly, gensim vector. As per performance results the keywords were chosen:

Model	Feature	Presion	Recall	F1 score
Random Forest	Keywords	0.74	0.72	0.69
Random forest	Gensim	0.55	0.61	0.53

Table 4. Performance of question classifications model

b) Paragraph Retrieval

The paragraph retrieval component was based on TF-IDF of terms in paragraphs returning the *top K* candidate paragraphs where the answer can be found. We can parametrize *K* and select the sentences for next step: candidate answer sentence selection. Figure 2 shows results for accuracy for the top k=5 paragraphs.

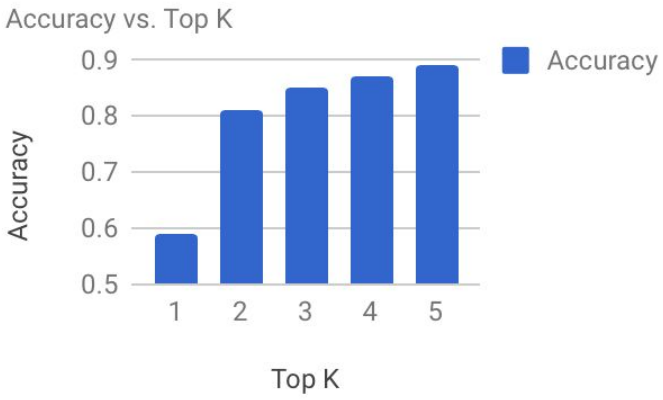

Figure 2. Accuracy in paragraph retrieval given top k documents

c) Answer sentence ranking and retrieval

Candidate answer sentence retrieval method heavily depends on the previous paragraph retrieval method. If the candidate sentences are large in number the performance and efficiency degrades significantly. We have finally selected K = 2 which has modest accuracy of best answer sentence detection, accuracy results are shown in figure 3.

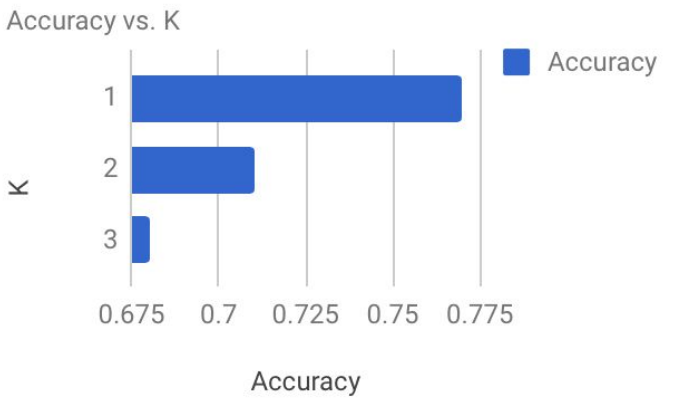

Figure 3. Sentence retrieval

It is worth to note that, to evaluate this method's performance we filtered out the samples where we couldn't find the correct paragraph. It can be seen that, if the length of the list of candidate answer grows the retrieval method performs poorly. In terms of run time, it increases with increase of the number of candidate answer sentences. The sentences are ranked based on the score described in section 5.d and the system selects the sentence with maximum score. It is noted that the systems penalizes the score according to sentence paragraph's rank and proximity of the keywords in the sentence.

Additionally, two models were incorporated but discarded due to poor performance. First, a binary classifier to predict if the sentence is a candidate or not, as a result, the system produced an imbalanced since the number of successes was 1 against *n* incorrect sentences. Secondly, a regression classifier was built but time constraints limited the final implementation.

d) Answer extraction from best sentence

The last part is extracting the answer from the best answer sentence. The answer extraction heuristic is based upon Named Entities, Parts of Speech (Noun), common keywords and standalone keywords of best sentence. First priority is the predicted Named Entity of our learned model and last priority is the keywords. All of the mentioned following results are based on K = 2. The method extraction was evaluated in terms of sequential string matching with the actual answer and predicted answer for the convenience of understanding the answer prediction accuracy. Results are shown in table 5.

Answer matching	Answer selection accuracy
Exact match	0.15
More than 80% match	0.02
60-80% match	0.09
40-60% match	0.12
20-40% match	0.30
>0.0-20% match	0.32

Table 5. Criteria for answer selection and evaluation

The performance of our heuristics are mentioned below. Table 6 shows the prediction of percentage represents the answer extraction using that heuristic and accuracy represents correct prediction of answer type.

Heuristic	Percentage of detection	Accuracy
Predicted answer type in sentence	0.67	0.63
Named Entity in Sentence	0.26	0.07
Nouns	0.07	0.01
Keyword	0.00	0.00

Table 6. Accuracy and percentage of detection for heuristic rules

5) Key findings & discussion of project

This section emphasizes mainly on key findings of working on this project and what we learned. This is not focusing on analyzing the result we achieved from the competition. Answers in the training dataset were in small case letters and it was initially very hard to find the Named Entities. On the other hand, the answer was in actual case letters on answer sentence in the paragraph. The answers in training dataset were tweaked and we reverse engineered it using regular expression to increase data to train the model. For about 15% of training samples we used rule based method to define the answer type of a question given that we couldn't find the Named Entity. For example, if there is a question with 'Who' keyword, we defined the answer type to PERSON.Stanford coreNLP [1] for python uses JVM which makes the IE process slower.

6) Conclusion

Question Answering (QA) system is a combination of document preprocessing, meaningful knowledge base creation, computational linguistics, knowledge representation, information retrieval, information extraction. The system not only relies on learned model from the knowledge-base but also rule based mechanism based e.g. synthesis and patterns of a specific language. A proper question answering system should consider syntax, semantics, morphology and pragmatics. The Kaggle in Class prediction competition illustrated a clear step by step guideline to build an automated QA system. This experience and concepts we developed will help to go deeper and contribute meaningfully for the NLP community.

7) References

[1] Manning, C., et al. (2014). The Stanford CoreNLP natural language processing toolkit. Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations.

[2] Li, X. and D. Roth (2002). Learning question classifiers. Proceedings of the 19th international conference on Computational linguistics-Volume 1, Association for Computational Linguistics.

[3] Jurafsky, D. and J. H. Martin (2014). Speech and language processing, Pearson London:.

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. Leveraging Linguistic Structure For Open Domain Information Extraction. In Proceedings of the Association of Computational Linguistics (ACL), 2015.

[3] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, A. Payne, "Okapi at TREC-4", Proceedings of the 4th TREC, 1995.

[4] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. Leveraging Linguistic Structure For Open Domain Information Extraction. In Proceedings of the Association of Computational Linguistics (ACL), 2015.

8) Appendix

Future work

The developed QA system can be improved by a robust NER tagger, Question Classifier and smarter Information Extraction component.

Knowledge Representation

i) Using word senses
ii) Word embedding using a Word 2 Vector model
iii) Incorporating word semantics

Question Classifier (QC)

iv) Learning a Question Classifier using the labelled question of [2].
v) Using a robust Word2Vector model for word embedding and learning QC
vi) Predefined set of rules for some question patterns

Information Retrieval(IR)

I. Paragraph Retrieval

Our model used TF-IDF at paragraph level for each document to retrieve the top paragraphs based on query. TF-IDF is based on term frequency. But, frequent terms might not always be the candidate answers. Instead we can use BM25 model. In our system we only retrieve the best 2 paragraph for computation efficiency which is a major limitation. Instead, we need to send more candidate paragraphs as answer can be in any of the paragraphs.

- Sentence Retrieval

Our scoring system for sentences in paragraph relies on common keywords, common named entities, proximity, longest common sequence of keywords, n-gram overlap. We select the highest scored sentence as our best sentence from candidate answer sentences. However, in this case, we can learn a regression model from training data using these features and predict the best sentence from the candidates. Pattern based approaches will also help to select the best sentence in cases where we are confident about the answer type.

Information Extraction

Our information extraction step from best answer sentence is not robust. Our top priority is the 'Answer Type' predicted from our Question Classifier which is basically the Named Entity. Instead, we can use

- Answer-type pattern recognition [3]
- N-gram tiling [3]
- Learning a model from training dataset using word embedding, POS, NER and predict the answers
- Include other tools like OpenIE[4]

9) Implementation Technology

Libraries and languages used to build the system is mentioned below: Python version: 3.5.4, NLTK: 3.3.0, CoreNLP[1]: English, 3.9.1,spaCy:2.0, Scikit-learn: 0.19.1

During the data preparation, we set up additional data computed previously to improve the time taken from our models to train or to to evaluate performance. Table 7 presents datasets and Table 8 the objects and models.

Folder	File	Description
project_folder/	df_training.pkl df_devel.pkl df_testing.pkl	Dataframes loaded from datasets and added with pre-computed data like question and answer's Name Entities for Error Analysis.

Table 7. List of datasets

File	Description
gensim_model	Gensim model tried for question classification
ner_corpus.pkl	NER tagged for all corpus
punk_tokenizer.pkl	Tokenizer
random_forest.pkl	Model for question classification
Vectorizer.pkl	Feature vector for question classification
vsm_inverted_index_corpus.pkl	Inverted index for vector space model

Table 8. List of objects