Question Answering Literature Review

Stephanie Uduji (suduji@stanford.edu)

Hujia Yu (hujiay@stanford.edu)

We focus on question answering (QA) and its related tasks (information retrieval, relation extraction, etc) in this literature review. QA is a complex natural language processing (NLP) task that is found commonly needed in various real-life situations. Moreover, most tasks in NLP can be essentially formulated as QA problems over some language input. For example, machine translation problems can be seen as asking questions (What is it translated to Chinese?), Relation extraction problems (What is the relationship between Steve Jobs and Apple?) As a result, we believe that understanding how to approach QA tasks can render us a powerful framework for solving various related NLP tasks in general. We have read several papers covering topics on relation extraction, information retrieval, and question answering. We present our research on dataset and reviews to some of the papers below.

Datasets

First, there exist high-quality publicly available datasets for working in QA. For example, Facebook bAbI dataset [9] is the standard dataset of 20 toy QA tasks. BAbI dataset is a synthetic dataset for testing a model's ability to retrieve facts and reasoning. Also, Stanford has recently developed a community for its own QA dataset, which is named Stanford Question Answering Dataset (SQuAD)[8]. According to its website, "…[It] consists of questions posed by crowd workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage." It is of huge size (100,000+ question-answer pairs on 500+ articles). There are plenty more sources of dialogue dataset and knowledge base dataset to use for both information retrieval and relation extraction related tasks. Therefore, data availability is not likely to be an obstacle to progress on our project.

Question Answering

Jurafsky and Martin provide a summary of the standard in QA in *Speech and Language Processing Chapter 28* [2]. The task (specific to factoid questions) has multiple common methods. The first of which is Information Retrieval-based QA. In this method, answers are found by extracting related texts and sources on the web. The Question Processing step involves determining the answer type, marking keywords in the query, and determining the subject. The type of answer that the question is looking for is also determined with supervised learning. This requires a hand-labeled dataset; however, the paper includes a potential data source, Webcopedia QA Typology, which has 276 hand-written rules and 180 different answer types (Jurfasky, Martin 4). Texts that hold answer candidates are found by performing a search for relevant documents either over a set corpus or a web search engine and relation extraction is used to filter out answers.

Another method, Knowledge-based question answering, involves running a query over a structure database. The previous method was a search over *unstructured* text. While this method involves the usage of hand-written rules, bootstrapping (like in relation extraction) can be used to find new patterns. Unlike the bootstrapping method in relation extraction, the data structure is now of a logical form in which entities are members of predicates (i.e. from relational logic). The challenge in this method lies in creating logical forms that accurately describe relations between entities. From what I've seen and experienced, one of the most challenging is the concept of superlatives (i.e. the *largest*, the *smallest*, etc). However, logical forms seems like they would be much more powerful in that they are more generalizable than hand-written regular expressions/string patterns.

The last stand-alone method, semi-supervised learning, involves generating multiple versions of paraphrases of questions and performing queries with each of these sets of paraphrased questions. The intuition behind creating paraphrases is to avoid confusion about syntax, word omissions/additions, etc. in the varied forms the same question may take. This involves manual work; however, another option is to use a public data source of questions and common paraphrases. The paper cites wikianswers.com's PARALEX corpus which has sets of millions of questions that have the same meaning. This will not do so well with very specific or uncommonly asked questions; however, it is an option that avoids the need to hand-label multiple sets of paraphrases.

Hybrid methods that utilize both Information Retrieval and KB-based question answering are also other options. The paper gives an example, the DeepQA system from IBM's Watson, a computer that can rival champions at Jeopardy. In future work and further research, it would be in our best interests to learn more about the specifics of the underlying methods used in the DeepQA system as it is an example of successful Question Answering. It is also another example of a successful combination of the most common methods in QA.

The paper describes that common evaluation of Question Answering uses the mean reciprocal rank (MRR) and requires that the test set be hand-labeled with correct answers. The classifiers utilized would need to return a ranked list of candidate answers, rather than just one answer.

QA problems differ from other tasks in that generally, it involves variety tasks to be done, including sequence tagging tasks, classification problems, sequence-to-sequence tasks, etc. The paper named *Dynamic Memory Networks for Natural Language Processing* by Kumar, Ondruska, et al. [3] had well taken care of all above perspectives and introduces a dynamic memory network (DMN) that processes input sequences and questions using attention process to give better answers.

This DMN model provides a perfect overview of what QA tasks are comprised of by designing a combination of RNN units, which they refer to as modules in the paper.

More specifically, DMN is made of four major modules that help with different parts of QA process. They are: input module, question module, episodic memory module, and answer module. These modules are, not surprisingly, essentially just different layers of neural networks stacked on top of each other. However, separating them into different modules helps the model to focus on different part of the input at a time throughout the QA process, which allows this model to reach high performance.

Firstly, the input module helps with text encoding, where it encodes raw text inputs into distributed vector representation. It's interesting that it encodes the input sequence via a recurrent neural network with given word embeddings. More specifically, it uses a gated recurrent network (GRU). Next, the question module also encodes the question into a distributed vector representation, except that it is later fed into the episodic memory module, whereas the input module does not.

Most importantly, the episodic memory module is the key to the success of DMN model. It uses the attention mechanism to choose which parts of the inputs to focus on given the question. More specifically, it takes in the final hidden state from question module and iterates over hidden states from the input RNN and updates its memory on which input hidden states to focus on. It updates previous memory and outputs an episode at each iteration, which is then used to update next memory. Lastly, the last module is the answer module, which generates sentence/words based on memory module.

An interesting point brought by the authors is that for the hard reasoning tasks related to NLP problems, multiple passes for the memory cell (attention) over the same inputs are crucial to achieving high performance. Therefore, in addition to having the model go through multiple iterations to converge, multiple passes over the same inputs should also be widely implemented in NLP tasks in general. This point makes great sense. The need for multiple passes of the input could be unique to NLP problems due to its complex input embeddings. Therefore, passing through the input text multiple times allows input representation to converge to a more stable state that incorporates more meanings.

The first paper frequently mentions, and bench marks its results against models introduced by Weston, et al. [7] Therefore, the next paper I will review is the paper by him, named *Towards AI-complete question answering: A set of prerequisite toy tasks*. This paper takes a quite different approach from DMN model above, instead of building a complicated model, this paper introduced a unique perspective of evaluating QA methods, which is by creating a large variety of toy tasks to see if the QA system has the corresponding "skillsets" to perform well on those categories of toy examples.

Instead of viewing a variety of NLP tasks as QA problems like the previous paper does, this paper views different types of QA tasks as specific skillsets. A better way to put it is that, just like humans, the models can be strong at some tasks and weaker in others. The goal of this paper is to identify the weaknesses of current models and help motivate new algorithm designs.

Based on this motivation, this paper started with conceiving a collection of much simpler QA tasks, with the objective of observing strengths and weaknesses of the model. What's mind-blowing to me is that this paper further relates this idea to the way software testing is done in computer science, explaining that the QA test cases lie at the bottom of a tree and are supposed to be independent of each other. Test cases on leaf nodes indicate simple unit tests, and parents of leaf nodes are more complicated QA tasks that is integration of the bottom skillsets. This makes strong intuitive sense to me and I strongly agree with this approach of testing QA methods.

More specifically, the paper divides the QA tasks to be of the following categories: single supporting face, two or three supporting facts, two or three argument relations, yes/no questions and counting and lists/sets questions. The last type of QA questions are questions such as "How many objects is Daniel holding?", which is essentially different from "What is Daniel holding? ".

Interestingly, all the tasks are generated with a simulation which behaves like a text game. The paper explains that the idea behind generating text within this simulation is that it allows them to apply constraints to the text and to limit the language to a coherent and controlled world. Although we do not completely agree with this approach, we think there is still value to evaluating models on simulated world to have better control of the data set and to understand the limits of the model better.

Having built the simulator and gathered the simulated tasks, the paper evaluates model performance of the following methods: N-gram classifier baseline, LSTMs, Memory Networks (MemNNs) (Weston et el 2014), a structured SVM that incorporates external labeled data for existing NLP tasks. The model that we are most interested in though, is the structured support vector machine (SVM) mentioned as a baseline in the above evaluation process.

In the results of the above evaluation, the structured SVM method performs better than vanilla MemNNs, which is impressive since MemNNs are a multi-layer neural networks model, whereas SVM is a rather straightforward classification model with external resources. This result strengthened my interest in the structured SVM approach as our primary approach to building and testing our QA system.

Another paper we were really inspired by is from Kun Xu et al. Their paper named *Question Answering on Freebase via Relation Extraction and Textual Evidence*. Unlike previous papers that builds QA system based on raw text inputs (stories), this paper introduces knowledge-based question answering systems, which is a more classic way of performing QA. This paper first retrieves candidate answers from Freebase using relation extractor, and then infer its validity using external resources (Wikipedia).

What might not be intuitive at first is how this paper combines the two sources. It turns out that the model first relies on knowledge bases to capture real world facts and then relies on Wikipedia to validate these facts. If we are to answer a question "What mountain is the highest in California?", relation extraction methods would usually identify entities in the question, which are "mountain", "California", and would fail to account for the relationship 'highest' and would therefore likely output all mountains in California.

Because 'highest' is a rather sophisticated representation that incorporates mathematical meanings, which would require the QA system to retrieve first all mountains in California, then the heights of those mountains. It then need to sort the heights in descending order and finally pick the first entry and output. The method proposed by the paper joins the external resource with internal knowledge base and can help filter out mountains that are known not to be the highest by looking into Wikipedia, and thus pick the correct one. This model makes great sense and is shown to outperform existing state-of-art models in 2016.

What we also find intuitive is that when the model extracts entities' relations, it follows a dependency tree-based method to divide one question into sub questions that can located directly in the KB, and the final output to the original question is obtained by taking the intersection of the answers to the sub-questions. This approach again makes great sense and is a heuristic for a simple classification problem.

What we like about this paper is that it sheds light on an approach that is drastically different from the previous two papers, and it introduces novelty based on a rather classic way of doing QA, which is knowledge-based relation extraction. Yet the approach is reasonably straightforward, and it also has shown to perform at state-of-art.

Information Retrieval

The next few reviews will be based on papers that are more related to information retrieval, which is also the foundation of building QA systems.

In QA, an essential step is relation or information extraction. The goal of information extraction is to create structured data from unstructured data. For example, one particular task might be determining the relationship between two people (if any) mentioned in block of text. In *Speech and Language Processing Chapter 21* [1], by Jurafsky and Martin (2017), a general overview of modern techniques in Information Extraction provides a good starting point for our project. The paper cites five main classes of algorithms for relation extraction: hand-written patterns, supervised learning, semi-supervised learning, distant supervised learning, and unsupervised learning.

The general method for information extraction follows a set-up of, first, finding named entities (named entity recognition) which is possible with the use of open source tools such as Stanford NER. Then, after labeling the training data these entity tags, create feature vectors based on the sentence/text that the entities occur in. The most commonly used features are word shape, short word shape, part of speech tags, whether an entity contains a prefix (all prefix lengths <= 4), whether an entity contains a suffix (all suffix lengths <= 4), the presence of a hyphen, the presence of words/entities in a gazetteer (i.e. list of place names). Training a classifier on these features should result in an output of a predicted relation between the entities.

The first of the four main classes of algorithms—hand-written patterns—heavily relies on regular expressions to find sentences that match the defined patterns. For example, the lexico-syntactic pattern of [X *such as* Y] is a common form for the hyponym relation. Modern versions of pattern matching don't only use the raw text, but also include the entity tags as constraints. This method yields high precision as the patterns can be tailored to specific domains, but suffers from low recall and involves a lot of time dedicated to defining these specific patterns. Given this limitation and the lack of generalization, this is the least likely method we will employ in our project.

The fourth method mentioned, distant supervised learning, is a combination of bootstrapping (semi-supervised) with supervised learning. A common, successful theme that we've seen in the papers is the tactic of combining effective methods to utilize the advantages while attempting to mitigate the weaknesses of each. In distant supervision, the process involves starting off with a large training set and running a supervised classifier. Some effective features for the learning step include entity tags, bag of words between each entity in a sentence, and dependency paths between entities. Because this method involves a large training set, the features can be rich (i.e. combination of features) without running the risk of over specification. Additionally, it is important to have a large enough dataset to avoid noisy information.

As mentioned before, we don't need to worry about data availability for performing these QA tasks. Similarly, with information retrieval related tasks, we don't have cause for concern about the availability of large datasets as the paper also points to potential sources that we might use for this task. These include: Wikipedia Infoboxes (set of relations), Wikipedida DBpedia (over 2 billion RDF triples), Freebase (set of relations), and WordNet (set of relations). The distant supervised learning method utilizes a supervised classifier, so we are able to evaluate against a gold standard.

For the future work, we plan to start approaching and exploring QA tasks with a knowledge-based relation extractor first, using easy-to-understand-and-implement dependency parsing trees to parse questions into sub questions and find relations between input entities. Once that part is done, we can then add more flavor to it by incorporating models with better predictive power, such as RNN structure or structured SVM mentioned in [5], and possibly combine with external resource like that mentioned in [4] to achieve higher accuracy during relation extraction process. The above method only works for questions that require some words as their answers. If the question requires more complicated answer, we are interested to dive into module networks introduced in [3], where we can try and implement an episodic memory module and an answer module on top of our relation extractor.

References:

[1] Information Extraction: Jurafsky, Daniel and Martin, James H. *Speech and Language Processing Chapter 21*. http://web.stanford.edu/~jurafsky/slp3/21.pdf

[2] Question Answering: Jurafsky, Daniel and Martin, James H. *Speech and Language Processing Chapter 28*. http://web.stanford.edu/~jurafsky/slp3/28.pdf

[3] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus and Richard Socher. 2016. *Ask Me Anything: Dynamic Memory Networks for Natural Language Processing.* arXiv:1506.07285

[4] Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. *Question Answering on Freebase via Relation Extraction and Textual Evidence.* In Proceedings of the Association for Computational Linguistics (ACL 2016).

[5] Weston, J., Bordes, A., Chopra, S. & Mikolov, T. *Towards AI-complete question answering: a set of prerequisite toy tasks.* http://arxiv.org/abs/1502.05698 (2015).

[6] Graves, A., Wayne, G. & Danihelka, I. *Neural Turing machines*. http://arxiv.org/abs/1410.5401 (2014).

[7] Weston, J., Chopra, S., and Bordes, A. *Memory networks.* In ICLR, 2015b.

[8] **S**tanford **Qu**estion **A**nswering **D**ataset (SQuAD) https://rajpurkar.github.io/SQuAD-explorer/

[9] *The bAbI project* https://research.fb.com/downloads/babi/

[10] Su, Miller, Zhu, et. al. *Solving the Prerequisites: Improving Question Answering on the bAbI Dataset* http://cs229.stanford.edu/proj2015/333_report.pdf