Question Answering Project Milestone

Stephanie Uduji (suduji@stanford.edu)

Hujia Yu (hujiay@stanford.edu)

Project Goals

As we mentioned in our literature review, our project will focus on question answering (QA) via relation extraction and validation with supporting textual evidence. We envision our end-product to be something like a chat interface where users will be able to manually type out questions in standard, colloquial English and receive answers from our QA machine. Our specific goals in this project are to explore 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, and to combine with external resource as textual evidence (such as Wikipedia) to achieve higher accuracy during the relation extraction process. As a reach goal, if a question requires a complicated answer, we are also interested in diving into module networks, where we can try and implement an episodic memory module and an answer module on top of our relation extractor.

Prior Approaches

During our exploration of literature reviews, we found the following approaches to be inspiring to our current approach, our discussion of prior approaches will be divided into the approaches on QA systems and those on information retrieval/relation extraction, since both parts are equally important in building a QA model.

Firstly, 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 structured 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 seem 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.

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.

In addition to pure QA systems, 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.

Current Approach

Our current approach is to empower a knowledge-based question answering system that is built using entity linking and relation extraction and integrated with additional evidence from Wikipedia to give more accurate and expressive answers. Following the model of QA via relation extraction and validation with supporting textual evidence in Xu, et. al's paper [4], this approach fully recognizes the capability of using knowledge bases like Freebase to capture real world facts, yet the model tries to improve the accuracy by incorporating web resources to validate or support these facts. The idea is simply to answer our question using KB relation extractor first, and to filter out wrong answers using external web resources. This novel method for question answering which infers both on structured and unstructured data is then to be evaluated on a benchmark dataset WebQuestions in the end.

Figure 1. on the right-hand side gives an overview of our method for the example question "Who did shaq first play for? ". There are two main steps: (1) inference on Freebase (KB-QA box); and (2) further inference on Wikipedia (Answer Refinement box). Furthermore, there are three steps in our step 1 (KB-QA box), and they are entity linking, relation extraction, and joint inference. More specifically, given a question, we first divide it to sub questions if needed, and then perform entity linking to identify a top entity in the sub question and its possible Freebase entities. Next, we use a relation extractor to predict the potential Freebase relations that could exist between the linked entities from the question and the (potential) answer entities. After that, a joint inference is performed to step over the entity linking and relation extraction results to find the best entity-relation pairs which will produce a list of candidate answer entities. This list of answer pairs is then pruned to filter out wrong answers in step 2 using the answer refinement model, which takes the Wikipedia page of each answer entity pair and looks for validation.

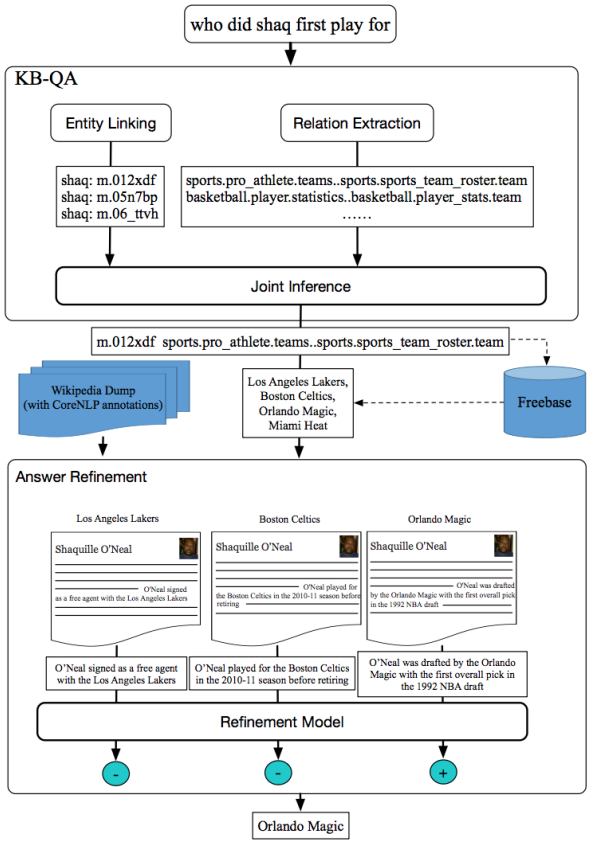
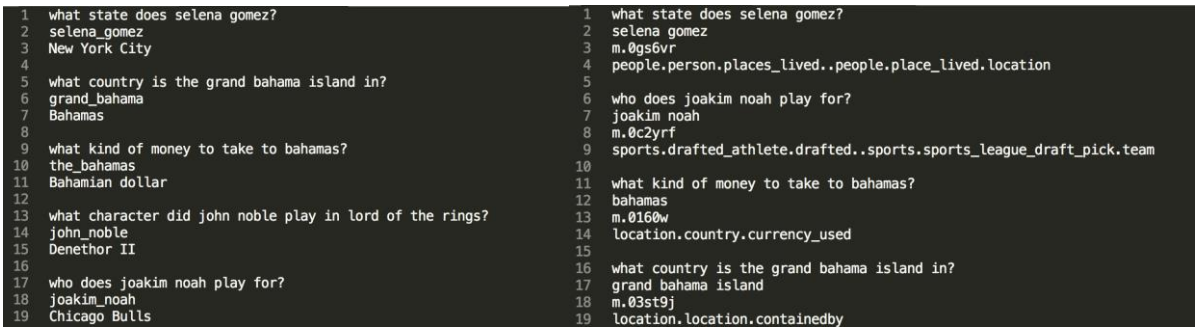


Figure 1: An illustration of our method to find answers for the given question *who did shaq first play for*.



```
what state does selena gomez?
selena_gomez
New York City

what country is the grand bahama island in?
grand_bahama
Bahamas

what kind of money to take to bahamas?
the_bahamas
Bahamian dollar

what character did john noble play in lord of the rings?
john_noble
Denethor II

who does joakim noah play for?
joakim_noah
Chicago Bulls
```

```
what state does selena gomez?
selena gomez
m.0gs6vr
people.person.places_lived..people.place_lived.location

who does joakim noah play for?
joakim noah
m.0c2yrf
sports.drafted_athlete.drafted..sports.sports_league_draft_pick.team

what kind of money to take to bahamas?
bahamas
m.0160w
location.country.currency_used

what country is the grand bahama island in?
grand bahama island
m.03st9j
location.location.containedby
```

Figure 2. Training KB Relation Data and Question Answer Pairs. (Left) Question answer pairs with entity. (Right) Question Answer Pairs with KB relations.

We also made sure our dataset contains sufficient information for training and is easy to use. Figure 2(left and right). gives snapshots of two files of our training/dev dataset. Specifically, our dataset is formatted in question-answer pairs with annotated question entities. Moreover, in the KB-QA training part, the dataset also gives the KB relations corresponding to each question entities to train the relation extractor.

Current Progress

We have implemented a baseline that encapsulates a simplified version of the model. We have implemented major steps of entity linking and relation extraction in our KB-QA box (Step 1) and we are finishing up implementations and evaluations of the inference based on KB part. The next step in the process we need to implement is Answer Refinement Box (Step 2 above), which has validation via a source like Wikipedia to rank top candidate answers and then return the top answer. Specific implementation details are explained below.

Inference on Freebase

This is the part one of our model and is also the most important part. Namely the inference based on freebase can itself be a standalone QA system, and the second part (answer refinement box) are just some extra steps to improve accuracy. Therefore, a correct and effective implementation of inference based on Freebase is key to success of the entire model. This inference part involves three major steps: entity linking, relation extraction, joint entity linking and prediction. Below we explain our implementations for each step we have completed.

Entity linking

The goal of this part is that when given a question in raw text form with standard English, we identify possible entities by using part of speech tagging. Question words (i.e. "who", "what", "where", etc. with a POS tag of "WP") and nouns with POS tags shown in Table 1. below

NN	noun, singular or mass	table
NNS	noun plural	tables
NP	proper noun, singular	John
NPS	proper noun, plural	Vikings

Table 1. Part of Speech Tagging used in Entity Linking.

are considered as possible entities. Our baseline diverges from the original implementation that includes an additional step of linking these mentions to entities in the Freebase with the entity linking tool, S-MART. This tool is not available for usage in python and the process of porting the tool would be extremely time consuming. So, in our future work, we will consider using other linking tools such as wikilinks/nel (on GitHub). For now, we use the mentions extracted with POS tagging to find matches in the KB based on substrings (i.e. if we've extracted "smith," we will check for "Sam Smith," "Adam Smith," etc. in the KB). Exact matches of the mentions will receive a count ranking of float('inf'). Other matches will receive count rankings based on the number of times the entity is found to be a match based on the mention list. We retrieve up to five entities for each mention with this ranking scheme. The output is a dictionary of mentions as keys and the top five entities as values.

So far, this entity linking model has been able to extract and rank the correct entity by searching the KB for a small training set of questions. If there is no match and an entity can't be found, the mention itself is treated as an entity. The next consideration we need to make is how to distinguish between mentions and other parts of a question. For example, given a question: "who stars in the great buck howard?", the only entity mentions should be "who" and "the great buck howard" (referring to the movie). However, our current model will also extract "stars" because it is tagged as a plural noun rather than a verb. We will need consider when to make this distinction as it might not necessarily be important to do so at the entity linking step.

Relation Extraction

After we implement entity linking, which can identify possible entities given a raw text question, we now want to infer the Freebase relation corresponding to question entities and the answer entities, which is the goal of relation extraction part.

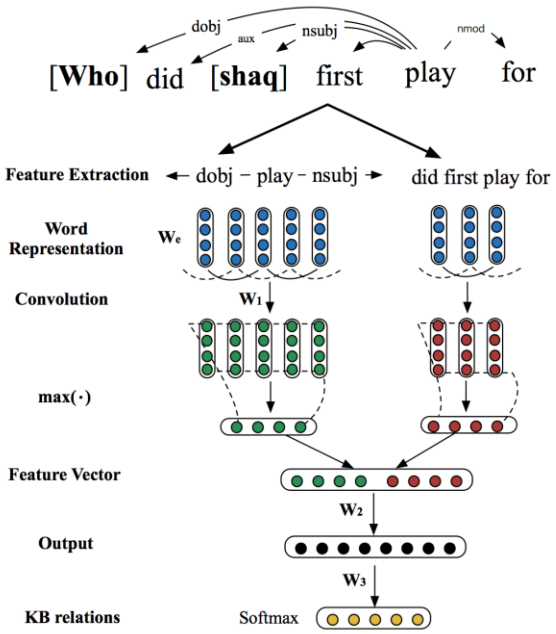


Figure 3. Overview of the Multi-channel Convolutional Neural Network for Relation Extraction. Sentence is first parsed into features using dependency parser and the shortest path between the question word and entity are found and concatenated to feed into model.

We implemented a Multi-Channel Convolutional Neural Network (MCCNN) for relation extraction. Figure 3 above gives an overview of this neural network. Specifically, two types of features are first extracted from a given question: syntactic features and sentential features. Syntactic features are extracted using Stanford CoreNLP dependency parser [11] to get the shortest path between the question word and an entity word in the question, which is represented using a concatenation of dependencies between words, non-entity words and path arrows for any two given pairs of question word and the question entity. On the other hand, sentential features are simply words in the questions excluding the entities themselves. These two set of features are then both represented using a pre-trained 50d GloVe model and fed into a convolutional neural network separately. The output is then concatenated to be one fixed-length feature vector after max pooling, which is then used to predict KB relations.

One modification that we made during this part of implementation is that when pre-trained GloVe model is used to represent the dependencies features extracted from a given question, some of the dependency terms are not found in the GloVe vocabulary. For example, dependencies like 'dobj', or 'nsubj' are not found in GloVe. As a result, vector representations of 'object' and 'subject' are used to represent those dependencies. It is unclear at this point how much impact this change will have on the overall model performance, but we could always allow for back propagation into the word vector representations to allow for some training to improve the representative power of feature vectors. Other than this modification, the rest of the implementation of this part strictly follows the state-of-art method discussed in the original paper [1].

Conclusion

We are currently in the process of finetuning / evaluating our relation extractor neural network model, we have the pipeline set up from entity linking to feature extraction and concatenation given an incoming question, yet the convolutional network takes time to train and our relation extractor is prone to error propagation from previous steps by design. Therefore, we are currently conducting unit testing and model evaluation to minimize intermediate errors and to make sure every step is implemented correctly before putting them together to begin the joint entity linking and prediction step. There are some parts of the process that we are still experimenting with as well (i.e. using an entity linking tool, like S-MART, as in the paper rather than our own, etc.). We hope to use these variances from the paper as a means of comparison with their results as well as an opportunity to further explore this variant of QA so that it isn't limited by the coverage of the Freebase data set.

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] Stanford Question Answering Dataset (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

[11] https://nlp.stanford.edu/software/lex-parser.shtml