Corpus-based Question Answering with Neural Module Networks

Runpeng Liu
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA
rliu42@mit.edu

Liang Zhou
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA
zhoul@mit.edu

December 13, 2015

Abstract

One critical but challenging problem in natural language processing is to consistently understand and correctly answer general questions about text. In this paper, we tackle a subset of these questions– standardized multiple-choice science exam questions– using neural module networks. NMNs have recently be applied successfully in machine vision for simple visual Q&A tasks. Here, we extend some of the original modules (i.e. measure, combine, attend) to be able to process text corpora as well as images. We define and formulate our own application-specific modules that are finely tuned to the corpus of scienftic knowledge being analyzed, and to the particular task of answering multiple-choice questions correctly. To further improve accuracy, we also attempt tocompose trained NMNs with feature detection methods.

1 Introduction

Question-answering is fundamentally compositional in nature – to determine a plausible answer to a natural language question, it is necessary to semantically parse the query statement, formulate a logical representation of the parsed nodes on which NLP computations may be made, and then compose these representations together in a way that accurately models the intent of the original question.

For instance, in the question *"How many people are on the bus?"*, the phrase *"how many"* indicates that we need to attend to quantity, *"people"* alerts us that this attention must be focused on *people* (and not *plants, backpacks, books,* etc.), *"bus"* signifies the object we are considering, and *"on"* describes how we are considering that object. To formulate a proper understanding of the question (and in turn, give a plausible answer), it is necessary not only to parse and categorize each of the semantic parts, but also model how they connect and compose with one another.

This paper describes an application and relatively new approach to question-answering using *neural module networks* (NMNs). NMNs offer flexibility above other machine learning Q&A techniques – the natural language computation of answering a question is divided among multiple independent, yet composable modules, instead of within a single large system.

To implement these modules, we reference a recent paper in machine vision [1], which implements several basic modules, such as `attend`, `classify`, and `combine`. In this project, we adapt these modules for use in corpus-based question answering.

2 Dataset

The dataset is composed of roughly 2500 8th-grade standardized multiple-choice science exam questions provided by the Allen AI Institute. We used this dataset for several reasons:

1. The question statements are reasonably standardized in proper, technical English.
2. The complexity/difficulty of the questions ranges quite uniformly between recalling simple definitions to multi-level high-order reasoning tasks.
3. Each question has one distinct, unambiguous correct answer. This answer can usually be deduced from a relevant chapter in knowledge base of science textbooks that we constructed and reference.

The neural modules we implement in this project take several distinct steps to find the most relevant answer to the question, and to some extent, our model is optimized for our dataset. Since there wasn't training necessary, all the available data was used as testing data. To test the module implementation, we inspected several dozen manually to ensure the correctness of the code.

3 Knowledge base

In addition, we constructed a knowledge base consisting of 6 middle-school level science textbooks in the following sub-fields: Physics, Chemistry, Biology, Earth Science, Life Science, and Physical Science, also linked by the Kaggle challenge. These textbooks were then indexed by chapter, so that relevancy searches would return a single chapter of the desired textbook as the target document on which queries would be made.

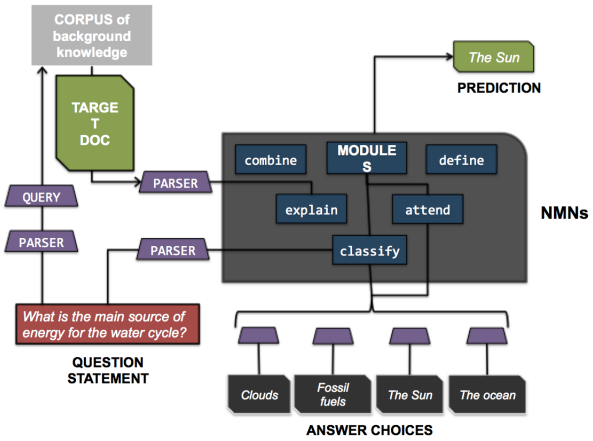

Figure 2: *The blueprint design for a neural module network. The parsed question and answer choices are passed through a series of independent, but jointly-trained modules that are composed to predict the answer.*

4 Neural Module Networks

Neural module networks themselves are fairly simple to explain. Instead of passing some preprocessed question through a large neural network, we use the information from preprocessing to find the relevant modules to apply to this particular question. For instance, "What is the age of the Earth?" might decide to use something like `define("Earth")`, but there is no need for the explain module at all. This allows for significantly more flexibility in the implementation as well as the application of these modules to question-answering.

In the preprocessing step, the question is passed through a semantic parser. In our particular implementation, we use the Python `nltk` [5] package along with the Stanford parser to break up the question. These sentence parts are then respectively analyzed to identify various dependencies, which can then be passed into our modules. In our implementa-

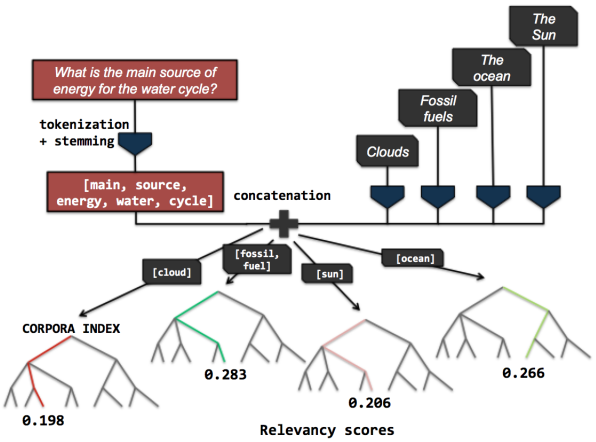

Figure 3: *A sample implementation of the Lucene baseline VSM. Questions and answers are only compared via relevancy in the textbook corpus, and the resulting probability distribution directly gives our chosen answer.*

tion, our modules are rather general but, again, also optimized for our model.

At the heart of all our modules is the relevancy module, which tests for how relevant a particular answer choice is. In order to actually combine our modules together, we pass the results of two modules into such a relevancy module, which finds a probability distribution and passes it up the module dependency tree.

5 Baselines

In order to establish some baselines for our results, we considered a few toy ML approaches as well as a more sophisticated VSM (Vector Space Model) pure relevancy approach. The main differences between these approaches and our neural network modules is the complexity to which the model understands the question – only the NMN actually breaks the question down into its component parts and attempts to choose an answer based on both the structure of the question and the answer choices given.

The first baseline we implemented used an aggregate boosting procedure to train weights for different models. These models were chosen to perform only slightly better than chance and then aggregated in a committee-style fashion to generate a final prediction. We chose intuitive features for each classifier used in the boosting procedure, such as string similarity of each of the answer choices to the question, `word2vec` distances, Levenshstein and Sorensen distances, etc. Due to the small size of the dataset, this pure ML approach is not likely to work well, as it forgoes true comprehension of the question. Indeed, this pure ML baseline performed only slightly better than chance, at 28.6%.

Our more interesting baseline made use of the Lucene search engine library [4] made available by Apache in order to search our corpus of 8th-grade science textbooks for the most relevant information. The so-called "Vector Space Model" (VSM) is an implementation based on what is called the *term frequency-inverse document frequency (tf-idf)*, although for this particular implementation, it was more of a *document frequency-inverse corpus frequency*.

The VSM searches only for the most relevant answer. First, it concatenates the question with each answer choice separately. The result is then tokenized and stemmed, and thus cleaned before the Lucene algorithm queries the corpus given the cleaned result. This query then returns a relevancy score, which is compared with the relevancy scores from the other answer choices. The answer choice with the highest relevancy score is then picked to be the proposed answer. An example of how it works is shown in Figure 2.

We see that relevancy searches perform substantially better than simple feature detection, but rele-

Baseline	# correct	% correct
Longest answer	602	25.6
Aggregate Boosting	727	28.6
Google Search hits	719	31.6
Vector Space Model (VSM)	832	35.4

Table 1: Baseline Comparison

vancy only brings us so far: no matter how much we train our VSM, it will fall short on higher-level reasoning tasks, simply because it doesn't take what the question is asking, or its context, into consideration. In order to understand the question itself, we need to break it down into its component parts, analyze each, and compose them together effectively.

6 Module Implementation

Compared to our baseline implementations, the neural module network actually builds upon and is implemented rather similarly to the VSM. The relevancy module is the key module used in our NMN, as we restrict ourselves to the corpus of science textbooks.

Our goal is to identify a set of modules that can be composed into the necessary configurations for the tasks that we'd like to use the NMN for. In this project, we identify mainly several simple, distinct modules that can break a rather simple query down into its components parts for use and analysis.

Here are some of the modules that we used [1]:

1. `attend`.
   Answer $\times$ Query $\rightarrow$ [(Document, Relevancy)]
   Every query uses the `attend` module in order to find the relevant section of text to evaluate. It finds the "attention" that a particular answer choice should have by taking in the answer choices with respect to the query and returning the relevant area of description or definition.

2. `define`.
   Document $\times$ Query $\rightarrow$ [(Attention, Relevancy)]
   Similarly to the `attend` module, the `define` module evaluates the relevancy of the input, but does so in the context of the document (corpus) instead of with the query. It defines and constrains a distribution within the text at the relevant area of attention.

3. `explain`.
   Attention $\times$ Query $\rightarrow$ [(Document, Attention)]
   It combines `attend` and `define` modules in response to trigger phrases such as "because", "describe", and "how does" by comparing what is being "explained" with the relevant area of the corpus. These distributions are then combined into a single probability distribution that is the output of the `explain` module.

4. `invert`.
   Document $\times$ Relevancy $\rightarrow$ Relevancy
   This module inverts the distribution over the answer choices. For instance, if the query contains "Choose all EXCEPT _____", then the module will reverse its probability predictions within its sub-modules (as part of the parsed query).

5. `combine`.
   Attention $\times$ Attention $\rightarrow$ Attention
   It acts as a placeholder to combine different modules in a compound context by averaging the likelihood distribution across its inputs. `combine` modules at the present are a bit overgeneralized, however, and don't consider what the actual link is between phrases.

6. `classify`.
   Document $\times$ Attention $\rightarrow$ [(Answer, Relevancy)]
   This module categorizes the final probability distribution as one of the labeled answers, de-

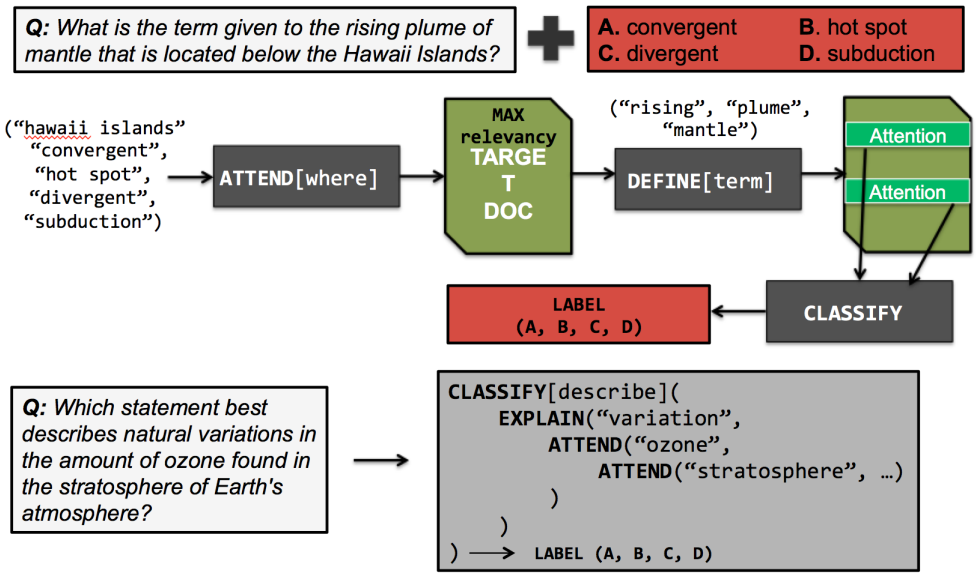

Figure 4: *An example of composition of modules in answering a question. The question is broken down into a series of modules that then work together to classify the answer to match one of the four given answer choices (labels).*

pending on the probability distribution it is given. With the same implementation it can also evaluate sub-module likelihoods not corresponding to the answer choices.

To compose our modules, we use the results of the parsing to pass phrases recursively into lower modules, which (mostly `attends` and `defines`) then pass their results back up via probability distributions on relevant areas of the text and on the answer choices. For instance, the word `is` is a clear indicator of the `define` module and is usually clearly separated in the query parsing step. Finally, once the results from all the modules have been received at the root, the `classify` module computes a final probability distribution, from which the prediction is obtained.

7 Results

We found that neural module networks performed substantially better than any of the baselines, but not as well as we had hoped – the accuracy was still under 50%, meaning most questions (generally the more complex ones) were answered incorrectly. We have included only the testing data results, as there were no parameters involved – simply algorithmic implementation.

Model	# correct	% correct
Aggregate Boosting	727	28.6
Google Search hits	719	31.6
VSM	832	35.4
NMN	1065	42.6

Table 2: NMN Accuracy

Pure machine learning approaches based simply on feature detection or aggregation of weak classifiers are unlikely to succeed – there are too many factors, and the underlying reasoning of the question cannot be model with a representation so naive. Instead, we learned that referencing a specialized knowledge base with relevancy searches, as well as decomposing the query, can help significantly in predicting the correct answer.

From Table 3, we observe that NMNs do indeed perform better than the Lucene VSM baseline on the same dataset. However, from seeing which questions were answered correctly and which weren't, we also note that NMNs were better at answering the same types of questions that the Lucene VSM usually got correct anyways – this makes sense, because simple questions are more likely to have answers similar to the results of a relevancy search. NMNs do much better on these because they actually manage to break the question down, but perform only slightly better on more complex queries.

	Lucene VSM	NMN
Correct	0.929	0.952
1st best	1.0	1.00
2nd best	0.879	0.892
3rd best	0.836	0.744
4th best	0.753	0.641

Table 3: Normalized Relevancy Scores

8 Limitations and Future Work

From the data that we've gathered, we can say that NMNs improve quite significantly upon VSMs and basic relevancy searches. They take the basic relevancy modules and compose them with higher-order modules to improve predictive power, especially on straightforward questions with easy-to-see module structures.

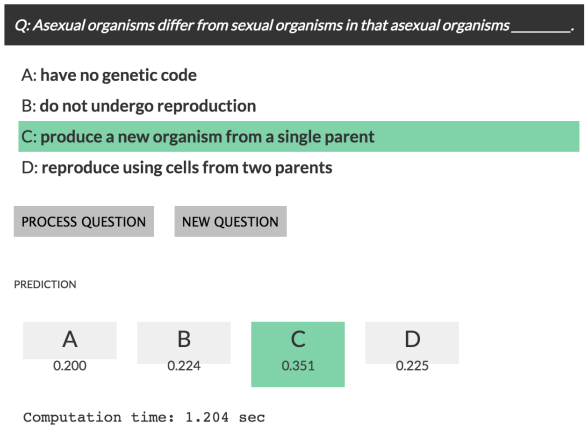

Figure 5: *An example of a relatively simple definition-based question that our model predicts correctly with high confidence.*

Nonetheless, in their current form NMNs can only analyze relatively simple sentences with clear module breakdowns. It is still rather difficult to optimize for complex questions that require multi-layer composability, or for example data that are natural for humans to reason about but difficult for computers to interpret. For instance, questions that ask for examples of a specific application, or compound questions referencing earlier parts of the problem, are still very difficult for our model to predict. The probability distribution over the answers shows this by being relatively even across all the answer choices.

On the other hand, having a specialized knowledge base compiled specifically for this standardized dataset was especially helpful – it performed considerably better relative to popular search engines, likely due to the lack of external noise and irrelevant information found in online sources. This is an aspect that obviously can't be applied to general question-answering, but in terms of standardizing what knowledge would be tested and what wouldn't,

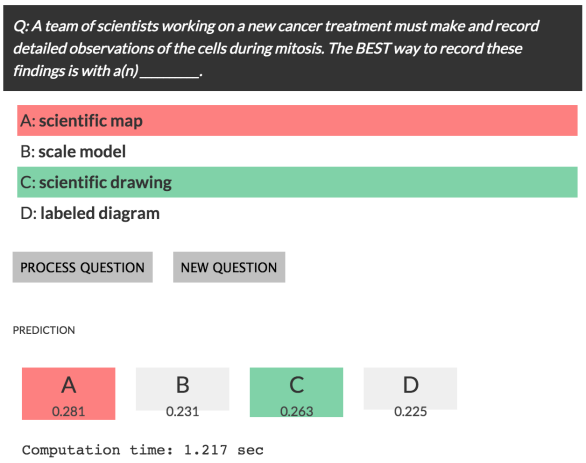

Figure 6: *An example of a longer query that is more difficult for our model to parse. Note that this requires more complex reasoning to answer correctly. The probability distribution over the answer choices is more even because the model is uncertain about them.*

limiting the scope improved performance quite significantly.

Future work includes (most pressingly) extending these modules to become more adaptive to more complex queries, as well as providing the option, instead of just answering questions, to allow users to specify which modules to use and when for each question. Furthermore, the given modules are relatively simple, and it is easy to imagine more complex modules that work better in particular situations, or replacements that generalize better.

9 Acknowledgements

We would especially like to thank Professor Regina Barzilay and Karthik Narasimhan for their consultation during this project. Meeting with them was very helpful in developing a preliminary framework for our proposed approach. We'd also like to thank the instructors and other TAs of 6.864 for teaching us the background knowledge necessary to complete the project.

10 Codebase

There is a link to the codebase at `https://github.mit.edu/rliu42/6864-project`.

References

[1] Andreas, Jacob, Rohrbach, Marcus, Darrell, Trevor, and Klein, Dan. *Deep Compositional Question Answering with Neural Module Networks.* arXiv:1511.02799, 2015.

[2] L. Ma and Z. L. Andiyyer Hang Li. *Learning to answer questions from image using convolutional neural network.* arXiv:1506.00333, 2015.

[3] M. Ren, R. Kiros, and R. Zemel. *Image question answering: A visual semantic embedding model and a new dataset.* NIPS, 2015.

[4] A. Biaecki, R. Muir, G. Ingersoll. *Apache Lucene 4.* SIGIR 2012.

[5] E. Loper, S. Bird *NLTK: The Natural Language Toolkit* arXiv:cs/0205028, 2002.