SU-chef: The hands-free kitchen helper

Natural Language Processing
CS230 Project Final Paper
Stephanie Schneider, schneids

For this project, I created a solution to a problem that I, and I imagine many other people, have had while cooking: your hands are wet or dirty, and you just need to know the next step of a recipe. These days, most everyone accesses recipes on their phone or computer, and it seems like the screen always times out right when you're ready to look for the next ingredient. Instead of wiping your hands off to enter a password or swipe a fingerprint, what if you could just ask your device, for example, "how many cups of sugar do I add?" or "I just mixed the dry ingredients, what's next?" - and it gave a response! Even if you are an effortlessly tidy chef, we've all struggled with scrolling past endless life stories on a blog page just to get to the recipe. My project helps to eliminate all annoying practicalities of using online recipes; the user can simply upload a webpage, and begin asking their electronic sous-chef for everything else.

Using existing speech-to-text and text-to-speech implementations, the only missing ingredient in this operation is interpreting the question, and scanning the webpage for the answer. Deep learning has already been proven successful for similar reading comprehension tasks (see CS224N lecture on deep learning for NLP). I am using the Stanford Question-Answer Dataset (SQuAD) v1.1, which excels in teaching networks "extractive question-answering," in conjunction with the RecipeQA dataset, which focuses on question-answering in the context of online recipes. In the type of NLP exemplified by SQuAD, the final result provides answers composed of a subsequence of the passage, rather than counting or yes/no type answers. This is the exact type of question-answer needed for my application.

Modeling and Implementation

I used the SQuAD v1.1 dataset, which includes 100,000+ questions with extracted answers. I chose not to use the updated SQuAD v2.0 (adds 50,000+ questions where the answer does not appear in the passage) in order to simplify the problem, but also to make the SQuAD dataset more similar to the multiple-choice structure of RecipeQA, where an answer always exists in the passage [1,2].

I used a QANet deep network structure for my baseline. The QANet architecture uses convolution and self-attention to model local and global details, respectively. By using these techniques in conjunction, this implementation avoids the reliance on RNNs, which tend to greatly increase both training and prediction time. QANet takes in two sets of words, a context paragraph and a question, and outputs a span, which in this case is a direct subset of the original context. The SQuAD dataset is already formatted in a compatible way; I acquired both SQuAD and QANet from https://github.com/NLPLearn/QANet [4].

The basic structure of QANet follows nearly all existing reading comprehension models with a five-layer system: an input embedding layer, an embedding encoder layer, a query-to-context attention layer, an encoder layer, and an output layer. The novel difference in this model is its use of convolution and self-attention in the embedding and encoder layers, instead of RNNs. The specifics of the model can be read about in [4].

For Milestone 2, I introduced a novel dataset. The RecipeQA dataset is published in a different format than SQuAD, so some work needed to be done in order to merge the two. In particular, I deconstructed the multiple-choice, fill-in-the-blank format of RecipeQA to coalesce with open-ended SQuAD queries. Additionally, the RecipeQA examples include multimedia questions (e.g. using pictures of different stages of the recipe), which were filtered out.

The combined training dataset was constructed by adding the filtered RecipeQA train questions to the SQuAD train questions. Since my application deals exclusively with recipe data, I used only samples from RecipeQA for the dev and test sets (this deviates from my method in Milestone 2). It is important to note that while the training dataset differs, the dev and test set still come from the same distribution; this ensures that the model is built to hit the same target we care about in testing (recipe-specific questions), as taught in the “Structuring Machine Learning Projects” Coursera course.

The second significant change to the dataset since Milestone 2 deals with RecipeQA inference answers that aren’t strictly extractive spans of the context paragraph. Previously, I had been eliminating all such questions. However, this resulted in RecipeQA questions only making up about 2% of the combined training dataset (~2000 questions from RecipeQA and 100,000+ from SQuAD). My solution for these non-extractive questions was to use common substrings between the RecipeQA multiple-choice answer and the context paragraph as the extractive span.
I tested several different heuristic approaches to determine if the resulting overlap was of significance, such as total string length and length of individual words within the

string. All dataset modifications are implemented in the file `parse_json.py` in the submitted code.

Results

After exploring several hyperparameter choices (e.g. dropout rate, learning rate, and batch size), I found that the parameters for QANet in [4] were slightly superior, but more often the difference was insignificant. This makes intuitive sense: since SQuAD makes up the vast majority of the training set, and the RecipeQA questions have been modified specifically to match the format of SQuAD, we expect the hyperparameter tuning of QANet to be optimal for this dataset as well.

My iterative approach imposed overall time constraints, so I decreased the number of steps by a factor of ten (60000 to 6000). This will reduce overall accuracy, but still works to compare the architecture's performance on datasets constructed using different heuristics; in this sense, the comparative analysis is still meaningful. I evaluate the performance of the model using an exact match as well as F1 metric, which is what is used for the SQuAD leaderboard. By using F1, the evaluation strategy can ignore deviations in exact responses (e.g. "three," "three cups," and "three cups of sugar" could all be acceptable responses).

The datasets are defined by their RecipeQA heuristics.

- RecipeQA extractive: include only RecipeQA questions if the answer is an explicit span of the context paragraph
- RecipeQA 5-char span: includes RecipeQA extractive questions, plus questions whose answers had at least five-character common substrings with the context paragraph; use this substring as the answer
- RecipeQA 12-char span: includes RecipeQA extractive questions, plus questions whose answers had at least twelve-character common substrings with the context paragraph, stripping spaces and ignoring single-letter cut-off words (e.g. "d baking soda " →"baking soda"). Use this substring as the answer.
- RecipeQA significant words: includes RecipeQA extractive questions, plus questions whose answers shared a single word of at least eight characters. Use this single word as the answer.

Table 1 below shows the QANet performance for SQuAD data, both alone and including extractive RecipeQA questions, for reference.

Table 1: Reference performance on SQuAD data

Train Dataset	**Dev/Test Dataset**	**Exact match**	**F1**
SQuAD only	SQuAD only	7.71%	17.35%
87599 questions	10570 questions (dev=test)		
Combined (SQuAD + RecipeQA extractive)	Combined (SQuAD + RecipeQA extractive)	7.64%	17.15%
89062 questions	10770 / 10766 questions		

Since the QANet was optimized for the SQuAD testset, we need to isolate the test cases we care about, namely extractive recipe questions. The results in Table 2 show the results for training this network on several different versions of the novel SQuAD + RecipeQA dataset. All versions incorporated SQuAD questions in training, and used RecipeQA extractive dev and test sets, with 200 and 196 questions each, respectively.

Table 2: Performance for different training sets on RecipeQA extractive data

Train Dataset (+ SQuAD)	**Number of QA pairs**	**Exact match**	**F1**
RecipeQA extractive	89062	0%	3.5%
RecipeQA 5-char	94327	0.58%	1.59%
RecipeQA 12-char	90225	0.58%	3.36%
RecipeQA significant words	91078	0%	1.21%

These results suggest that none of the proposed heuristics improved upon a smaller and more specialized training set, though a modified 12-character overlap came the closest.

Conclusion

In order to encourage better performance on the combined SQuAD + RecipeQA dataset, I attempted to supplement the training set with examples representative of RecipeQA. The combined dataset draws roughly 2000 extractive questions from RecipeQA, but nearly 8000 more are being filtered out for the answers not being proper spans of the context paragraph. If we utilize these questions, the RecipeQA

representation jumps to 9% of the combined dataset. However, this project revealed that it is more valuable to have a smaller training set that is a better representation of the test data.

It's interesting that adding more samples *decreased* the accuracy of the predictions. In particular, I expected the "significant word" heuristic to improve upon extractive-only questions, but perhaps by just using words of a certain length, instead of language insight from CoreNLP or GloVe, this method effectively untrained the model to recognize significant words within a span.

The main takeaway from this project is the value of a good dataset. In future work (I still want to use this app), I will look into auto-generating more questions from each context paragraph in RecipeQA (as per SQuAD), or even auto-generating queries and answers from independent online recipes.

References

[1] Rajpurkar, Pranav, Robin Jia, and Percy Liang. "Know what you don't know: Unanswerable questions for SQuAD." arXiv preprint arXiv:1806.03822 (2018).
[2] Rajpurkar, Pranav, et al. "Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).
[3] Clark, Christopher, and Matt Gardner. "Simple and effective multi-paragraph reading comprehension." arXiv preprint arXiv:1710.10723 (2017).
[4] Yu, Adams Wei, et al. "Qanet: Combining local convolution with global self-attention for reading comprehension." arXiv preprint arXiv:1804.09541 (2018).