1. Intro

"How a rainbow come into being?"

This is a question an open-domain question answering (QA) system should be able to respond to. QA systems emulate how people look for information by reading the web to return answers to common questions. Machine learning can be used to improve the accuracy of these answers.

Existing natural language models have been focused on extracting answers from a short paragraph rather than reading an entire page of content for proper context. As a result, the responses can be complicated or lengthy. A good answer will be both succinct and relevant.

My goal is to predict short and long answer responses to real questions about Wikipedia articles. For each article + question pair, the network will predict / select long and short form answers to the question drawn directly from the article. - A long answer would be a longer section of text that answers the question - several sentences or a paragraph. - A short answer might be a sentence or phrase, or even in some cases a YES/NO.

2. Related Work

Decomposable Attention Model

This model is a simple neural architecture designed for natural language inference. Core idea of this approach is using of attention. It supposed that in natural language inference tasks there exist critical local texts in a sentence can be aligned with those in other sentences. This model achieved state of art result on the Stanford Natural Language Inference dataset with much fewer parameters than previous work.

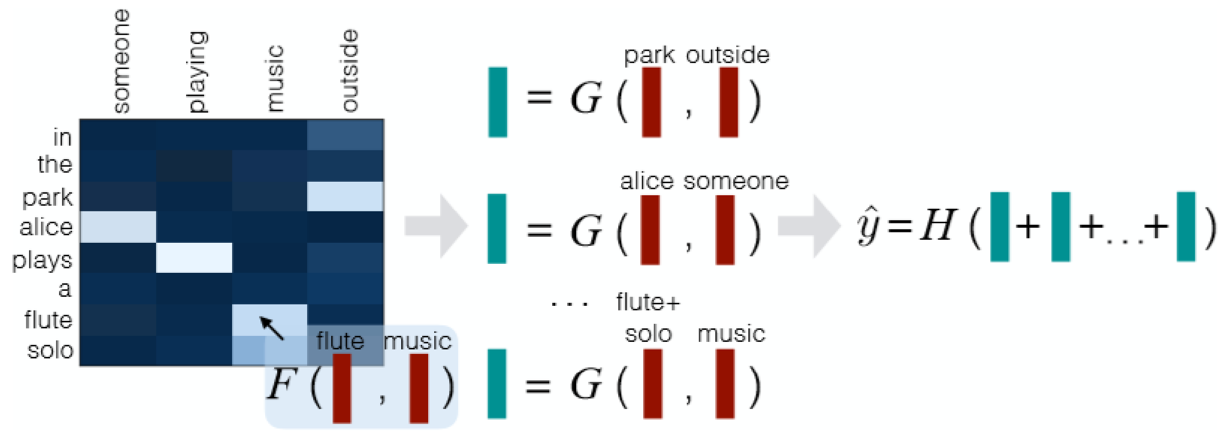

Figure 1. Pictoral overview of Decomposable Attention Model

In Google Research's work, they modified this model to adapt the long question answering task. Long answers candidates and the question are feed to the network. They expanded the output dim of

Decomposable Attention Model to 10 and gathered statistic parameter from data. They encoded long answer's position into a 10 dimensional trainable embedding, calculated the maximum number of a particular word shared by a question and corresponding context and the number of weighted shared words. Then, all these were concatenated as an input of a fully connected layer to give the final scores. But the result was not as good as in its original task. It only got 55% accuracy.

3. Methods

3.1 Long Answer Model

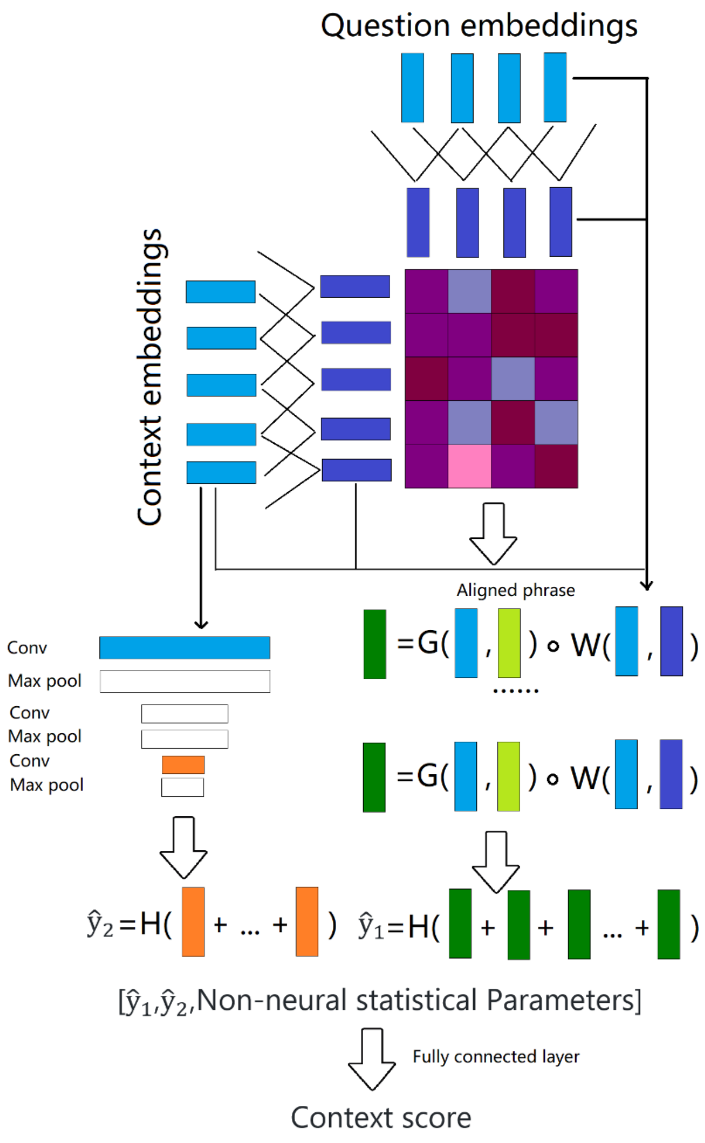
Question embeddings
Context embeddings
Aligned phrase
Conv
Max pool
Conv
Max pool
Conv
Max pool
$=G(\ ,\ )\circ W(\ ,\ )$
$=G(\ ,\ )\circ W(\ ,\ )$
$\hat{y}_2=H(\ +\ \dots\ +\ )$
$\hat{y}_1=H(\ +\ +\ \dots\ +\ )$
$[\hat{y}_1,\hat{y}_2$,Non-neural statistical Parameters]
Fully connected layer
Context score

Figure 1. Pictoral overview of Long Answer Model

Long answer model is improved upon Decomposable Attention Model. The input is a question and a long answer candidate pair. The output is context score for a particular long answer candidate.

Embed

The glove data is processed to be a dictionary so that embeddings can be generated directly from strings instead of index and look-up table. Hence, I am able to use full GolVe embedding while the baseline has a 40000 words limitation. Upper letters in the word are maintained. If dictionary does not contain this word, then all letters in this word are transferred to lower case letters. Out-of-Vocabulary words are deleted.

Attend

The unnormalized attention weights e_{ij} decomposes as:

$$e_{ij} = F(\bar{a}_i)^T F(\bar{b}_j)$$

The F function is a three layers 3*embed_dim convolution network sliding through words. Since adjective and adverb plus noun or verb may lead to different meaning. Local context information must be gathered to get a clear meaning.

Then aligned phases are calculate as follow:

$$\beta_i := \sum_{j=1}^{\ell_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_b} \exp(e_{ik})} \bar{b}_j \,,$$
$$\alpha_j := \sum_{i=1}^{\ell_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_a} \exp(e_{kj})} \bar{a}_i \,.$$

Scaler

Concatenate F and word embeddings. Pass it through two layer network to get scalar for every word.

After adjective or adverb aggregate with their neighboring word, they are useless. It is reasonable to block them.

Compare

Use G (a two layer fully connected network) to compare a word and its aligned phrase. Then times the scalar.

Aggregate

Since the questions need extra Wikipedia document to answer, this means the answer part of document has extremely low information entropy. So it can not be determined by merely comparing question and its aligned words from the document. Since aligning method can locate which paragraph has key features related to questions, the only way to determine answer is to judge the structural of the whole passage. So I designed a convolution network with 3 max pooling layers which can extract feature of each sentence of the passage. The output is summed up and average is calculated over words.

The results of comparation is also aggregated through summing.

These two vectors are concatenated to get context score.

3.2 Short Answer Model

Implement the Document Reader model (Chen et al., 2017)

4. Experiments

4.1 Overview

Google Natural Question Answering Dataset is used to evaluate the performance of different Networks. All networks are trained and tested on the same dataset through same epochs. The inputs is pair of a question in natural language , a corresponding Wikipedia document and long answer candidates. The long answer candidates are extracted from the Wikipedia document by dividing it using html tags. For long answer task, the output is index of long answer candidates. For short task, the output is index of start token and index of end token for short answer. In the meantime, the outputs can be Null cause some question does not have an answer. The accuracy of predicted long answers and short answers are evaluated.

4.2 Dataset

Natural Questions (NQ) contains real user questions issued to Google search, and answers found from Wikipedia by annotators. NQ is designed for the training and evaluation of automatic question answering systems. NQ contains 307,372 training examples, 7,830 examples for development, and a further 7,842 examples for testing. Each sample contain the following parts:

1.document_text - the text of the article in question (with some HTML tags to provide document structure). The text can be tokenized by splitting on whitespace.

2.question_text - the question to be answered

3.long_answer_candidates - a JSON array containing all of the plausible long answers.

annotations - a JSON array containing all of the correct long + short answers. Only provided for train.

4.document_url - the URL for the full article. Provided for informational purposes only. This is NOT the simplified version of the article so indices from this cannot be used directly. The content may also no longer match the html used to generate document_text. Only provided for train.

5.example_id - unique ID for the sample.

4.3 Evaluation Measures

Because this tasks require models to output Null for some questions, it is necessary to separately measure false positives, false negatives and accuracy free from the decision of whether or not to answer. The precision, recall, and free accuracy of model are defined as follow:

$$t\left(q^{(i)}, d^{(i)}, a, f_\theta\right) = h_\beta\left(a, f_\theta(q,d)\right)[[f_\theta(q,d) \neq NULL]]$$

$$R(f_\theta) = \frac{\sum_{i=1}^{n} t(\, q^{(i)}, d^{(i)}, a^{(i)}, f_\theta)}{\sum_{i=1}^{n} \chi_{[\beta,+\infty)}(g(a^{(i)}))}$$

$$P(f_\theta) = \frac{\sum_{i=1}^{n} t(\, q^{(i)}, d^{(i)}, a^{(i)}, f_\theta)}{\sum_{i=1}^{n}[[f_\theta(q^{(i)}, d^{(i)}) \neq NULL]]}$$

$$\mathrm{F1}(f_\theta) = \frac{h_\beta\left(a, f_\theta(q,d)\right)}{\sum_{i=1}^{n} \chi_{[\beta,+\infty)}(g(a^{(i)}))}$$

Where q is a question, d is a Wikipedia document, a is annotations and f_θ is a model with parameters θ, $g\left(a^{(i)}\right)$ is the number of annotations in $a^{(i)}$ that are non-null and h_β judges the correctness of model.

Definition of h_β. If$g(\mathrm{a}) \geq \beta$ and $f_\theta = \mathrm{a_j}$ for some j ∈ $\{1\ldots5\}$, Then = $h_\beta(\mathrm{a}, f_\theta)$ = 1; Else If$g(\mathrm{a}) < \beta$ and l = NULL, Then $h_\beta(\mathrm{a}, f_\theta)$ = 1 ; Else $h_\beta(\mathrm{a}, f_\theta)$ = 0.

4.4 Training

My model is trained for 30 epochs just as baselines. The negative log-likelihood of the correct answer is used as loss function:

$$-\sum_{i=1}^{n}(\log \frac{e^{z_{l^{(i)}}}}{\sum_l e^{z_l}}) \times (1\text{-}\eta[[l^{(i)} = \mathrm{NULL}]])$$

Other parameters are as following:

Batch size: 5

Learning rate: 0.05, 0.005, 0.0005

4.5 Results

	Long answer Test			Short answer Test		
	P	R	F1	P	R	F1
DocuentQA	47.5	44.7	46.1	38.6	33.2	35.7
DecAtt + DocReader	52.7	57.0	54.8	34.3	28.9	31.4
My method + DocReader	59.2	60.1	59.6	37.1	33.8	35.5

Table 1: Precision (P), recall (R), and the harmonic mean of these (F1) of all baselines and my method.

The results shows my method improved accuracy of long answer prediction for about 5%. It also improved the accuracy of short answer prediction, although the model of short answer part is the same as base line. This is because the short answer is always nested in the long answer produced by long answer model.

5. Conclusion

This work shows that introducing local context information for attention mechanism will help model get more accurate relation ship between sentences. And obtaining structure information of document is beneficial for question answering system extracts answer from the document.