Assignment 2, NLP

Dhruv Ahlawat, CS1210556

May 9, 2024

1 Model details

Query Encoding ($encoding_q$) - Output of LSTM over the question.
Column name Encoding - one encoding for each column name, created from output of an LSTM over the column names in each cell individually.
$encoding_{c_i}$ = encoding of ith column name.
Column Model - MLP over concatenation of column name encoding and query encoding, the max output was chosen as the target column.

Row cell encoding - LSTM over the contents of the cell of a row.
$encoding_{row_{i,j}}$ = encoding of the jth cell in the ith row.

2 Notable techniques

2.1 custom attention

Made a custom attention pipeline. The Column names were given scores (attention) based on the question encoding, similar to how the column target model works.

$$\overline{attn_i} = MLP(concat(encoding_{c_i}, encoding_q))$$

$$attn = softmax(\overline{attn})$$

Used these attention values to generate an encoding for each row by the following weighted sum.

$$encoding_{row_i} = \sum_{j=0}^{n} attn_j \times encoding_{row_{i,j}}$$

Then finally, assigned scores to each row using an MLP over the concatenation of row encoding and question encoding, similar to column model.

$$score_{row_i} = MLP(concat(encoding_{row_i}, encoding_q))$$

The difficulties in this method were mainly in terms of computation speed and batching since an LSTM is required to run on each individual cell of the table.

After a lot of programming I was able to batch an entire table in one pass (an LSTM ran over each cell in the table, with over 2000 rows so basically around 10000-16000 LSTM computations's were able to be batched, mostly because each cell had only a short sequence)
Further improvements I could have made would be to use Multi-head attention. The above method was able to overfit on the training data and got 52% over the validation set. The reason might be because of unseen and out of vocabulary words, since I trained the wordvectors on only the words of the training dataset.

2.2 detecting multiple outputs

I had now the custom attention function which I could use to learn what the question must be asking about. On generating an output, if

$$attn_i \geq M \times attn_j, \forall j$$

, where M is some fine-tuned constant, then the query must be related to the ith column. (Note that it is not the target column, but the one the question attends to most. For eg- for a query "which day was event X held?", the target column would be day but the way to detect answer rows would be the value in the "**event**" column)
on finding such an attention value, I went through all the rows and added rows to my solution if they had the **same value in the ith column as my selected row** and also had a high enough softmax score.
This resulted in mis-classifying certain single solution examples as well, but it increased accuracy overall by classifying more multi-labelled ones correctly.

Another way to detect multiple outputs would simply be to compare the softmax scores of the rows and if there are any contenders to the top then we predict them as well. I found that this did not perform very well and ended up reducing accuracy.