Question Answering with Match-LSTM and Boundary Pointers

Tal Stramer
Stanford University, Department of Computer Science

Background

- Reading comprehension long-standing problem in Natural Language Processing.
- Important for many tasks: Machine translation, sentiment analysis, question answering.
- Previous approaches to reading comprehension required a lot of manual feature engineering
- New models based on end-to-end neural architectures highly successful and require little to no feature engineering

Problem Statement

- **Problem**: Given a a question and answer pair, along with a context paragraph, predict a span within the context paragraph that answers the question
- **Evaluation metrics**
  - Exact match: the percentage of predictions that match the ground truth answers exactly
  - F1: the average overlap between the prediction and ground truth answers.

Dataset

- Stanford Question Answering Dataset (SQuAD) used for training and evaluation
- Training set of ~100,000 question-answer pairs, along with a context paragraph. 95% of the data used for training, 5% for validation.
- Test set of ~10,000 question-answer pairs
- Vector word representations based on Glove - trained on 'Common Crawl' dataset, 840 billion tokens and 2.2 million vocabulary size

Figure 1. SQuAD context lengths.

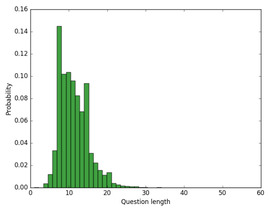
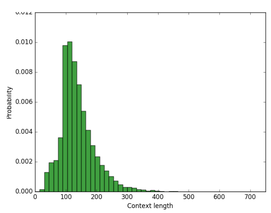


Figure 2. SQuAD question lengths.



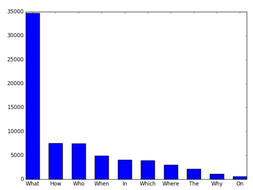


Figure 3. SQuAD question types.

Methods and Materials

- **Embedding layer**: Convert words in question and context into word embeddings
- **Encoder layer**: Encode questions and contexts using two separate bidirectional LSTMs
- **Attention layer**:
  - Option 1: Match LSTM to create a question-aware representation of the context
  - Option 2: Dynamic Co-attention to create co-dependent representation of the question and context
- **Prediction layer**: Use a pointer network to predict probabilities for each context word being the start or end index of the answer. Find most probable start and end index based on probabilities.
- **Cost function**: Cross-entropy loss over training examples

Experiments

- Experiment with various parameters
  - Word embedding size: 100, 200, 300
  - Word embedding source: Wikipedia, Common crawl
  - Hidden state size: 100, 200, 300
  - Co-attention encoder: Match LSTM, Dynamic Co-attention
- Run 10 epochs per experiment
- Evaluate F1 score and exact match score after each epoch on 100 examples from validation set. Choose model from epoch with highest F1 score.

Word Embed Size	Word Embed Source	Hidden State Size	Attention Encoder	F1 Score	Exact Match Score
100	Wiki	100	Match LSTM	0.41	0.32
200	Wiki	200	Match LSTM	0.55	0.38
200	Wiki	200	DCN	0.54	0.36
300	Common Crawl	200	Match LSTM	0.55	0.38
300	**Common Crawl**	**300**	**Match LSTM**	**0.62**	**0.48**
300	Common Crawl	300	DCN	0.60	0.44

Table 1. Experiment results.

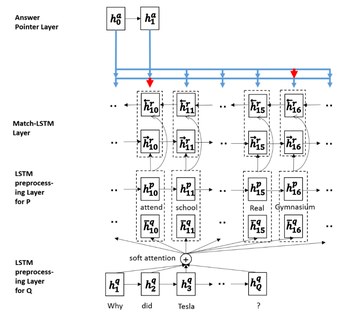


Figure 4. Illustration of neural architecture from Wang et al [1].

Discussion

- Prediction accuracy highly dependent on word embed size and hidden state size: too low and the neural network is not powerful enough to generalize
- Prediction accuracy highly dependent on answer length: If length is between 1-10 tokens, F1 score ~0.8 and EM score ~0.5. If between 11-20 tokens, F1 score ~0.4 and EM score ~0.1
- Not much of a difference between use Dynamic Co-attention and Match LSTM. Both encode a question-aware representation of the context.

Conclusions

- Relatively simple neural architecture can achieve high performance on question-answering task using SQuAD
- No manual feature engineering required is a huge benefit.
- Future work:
  - Better tokenization of question and context sentences using Stanford tokenizer
  - Integrate character-level embeddings into word representations
  - Build a more complex model that allows iteratively refining the question and context representation based on each other rather than one-shot approach
  - Analyze how model behaves on different types of question

References:
[1] Wang, Shuohang, and Jing Jiang. "Machine comprehension using match-lstm and answer pointer." arXiv preprint arXiv:1608.07905 (2016).