SQuAD
11-632 Fall Semester Report

Griffin Adams, Tian Qin, Yifan Yang, Matthias Grabmair
December 14, 2017

Introduction

The majority of the world's knowledge is encoded in text [2]. For machines to decode the knowledge and maximize its usefulness, they must achieve reading comprehension on par with that of humans. Human-level comprehension of natural language would jumpstart a host of vital artificial applications, such as Live QA, machine summarization, and intelligent information retrieval. It would also signal a dramatic step toward literate artificial intelligence. The proliferation of large human-annotated/cloze-style QA datasets [1][2][3][4] has spurred advancement in the state of the art for machine comprehension. For the Stanford Question Answering Dataset (SQuAD) [3], researchers have achieved state of the art performance with a combination of language modeling with recurrent neural networks, question-context attention mechanisms, dynamic answer extractors, and ensemble scoring methods [6][7]. While the above approaches have closed the overall gap vis-a-vis human performance: 87.8% versus 91.2% (F1 span overlap), the deficiency gap is pronounced for questions which involve reasoning, inference, common-sense, and contain lexical or syntactic divergence [1][2][5].

Hypothesis / Project Goal Our problem covers Span Question Answering: given a passage and a question about it, extract the contiguous span of text which best answers the question. Recurrent models suffer from the problem of early summarization, where they are forced to memorize the entire context into a fixed length vector. Self-attention models and Tree-LSTM BiDAF models address this early summarization issue head on by allowing potentially distant, but related words, to directly interact. We want to develop, evaluate, and improve upon the use of self-attention and tree-structured recursive models to the task of Span QA answer extraction.

Fall Development Goals Our milestones before the first midterm were implementing the Self-Attention model and implementing the Tree-LSTM. We met both milestones since they were just baseline models. Our milestones before the second midterm were to add BiDAF to the tree model. For the final, we added bi-LSTMs into the model layer/answer extractor, added a modeling layer for the Tree LSTM, and conducted extensive error analysis. All the milestones are completed. We could have left more time for hyper-parameter tuning since that can improve accuracy a lot. Progress was not monotonically improving and we remained flexible.

Experiment/System Design Overview

We design and implement two models: the Self-Attention model and the Tree-LSTM BiDAF model. The Self-Attention model is inspired by Transformer model [3]. Unlike traditional

sequential models, which uses RNN or CNN, the self-attention model is solely based on attention mechanisms, dispensing with RNN and CNN entirely. It learns the question and context representation using self-attention. Then it merges question and context information using another attention function. The answer extractor predicts the start index and the end index. The Tree-LSTM BiDAF model is composed of four main steps: generate a constituent parse tree, encode it recursively bottom-up, classify each node's relevance with respect to a query (question), and select the max node. We outline each in the following sections.

Experimental Design

Self-Attention Model

See Appendix 1 for an graphical overview of the Self-Attention Model.

Question: The question word embedding are initialized using Glove embeddings [5]. Positional embeddings are added into word embedding to capture word order information. It is defined as: $PE(pos,\ 2i)\ =\ sin(pos/10000^{2i/d_{model}}),\ PE(pos,\ 2i+1)\ =\ sin(pos/10000^{2i/d_{model}}),$
In the equation above, pos is the position, i is the dimension and $d_{model} = 128$ is the output dimension. After the embedding, there are N identical layers. Each layer consists of Multi-Head Attention and Position-wise Feed-Forward Networks (Explain below). Residual connections and layer normalization are added inside of each layer.
Context: Similar to question section, it contains word embedding, positional embedding. The first sub-module and third module of the N identical layers stay the same. The second sub-module merges question information and context information through Multi-Head Attention.
Answer Extraction: It generates the probabilities of start index S and end index E for the answer span through linear and softmax layer. We try to minimize the loss: $-logS(i)\ -logE(j)$
Attention: The dot-product attention defines as: $Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$. It combines Query, Key, Value of dimension d_k . We project Q, K, V into different spaces to form Multi-Head Attention: $MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O;\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$
Position-wise Feed-Forward Networks: It contains two linear transformations with RELU in the middle: $FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$

Tree-LSTM BiDAF Model

The following sections detail the steps for the Tree-LSTM BiDAF Model.
Generate a parse tree: We use nltk's interface to the Java Stanford Parser to generate parse trees. We choose the unlexicalized PCFG parser [8] for n-array constituency trees. We chose constituency trees over dependency trees because constituency trees preserve word order.
Word-level Modeling: We model both context and question separately through a Bi-LSTM with shared weights. We use the concatenation of the final and first hidden states of the forward and backward passes, respectively, as summary representations of the context and question.

Recursive bottom-up encoding: Tree-LSTMs [9] have demonstrated the most success among recursive composition functions and can handle arbitrary number of children. We choose the Child-Sum variant. To compute a parent representation (both hidden state and cell state), the LSTM function receives a list of the child hidden and cell states. The forget gate is computed separately for each child, allowing it to selectively ignore irrelevant subtrees.
Leaf nodes [w_1, … w_n] are represented as:

$$(h_1, ..., h_n), (c_1, ..., c_n) = Bi-LSTM(W_{emb}(w_1), ..., W_{emb}(w_n))$$

Non-terminal node states, p_i, are computed as

$$h_i, c_i = ChildSumTreeLSTM((h_j, c_j) \forall j \in C(h_i))$$

Where $C(h_i)$ represents the children of node i. The Child-Sum Tree LSTM equations are:

$$\widehat{h} = \sum_j^C h_j \qquad i = \sigma(W^i \widehat{h} + b^i) \qquad o = \sigma(W^o \widehat{h} + b^o) \qquad f_j = \sigma(W^f h_j + b^f) \forall j \in C(h_i))$$

$$c_i = i \odot u + \sum_j^C f_j \odot c_j \qquad h_i = o \odot tanh(c_i)$$

As can be seen above, the model assigns each constituent equal weight (the hidden states are added $\widehat{h} = \sum_j^C h_j$. Yet, headwords (both lexical and semantic) should be given more preference in their parent's encoding. To accomplish this, we produce the child sum hidden state according to the following equation $\widehat{h} = \sum_j^C a_j h_j$ where [a_1,...,a_C] is a learnt distribution over child nodes. They represent the relative importance of each child to its parent. We learn *a* according to the dot product of the concatenated meta embeddings of the child and parent, a learned weight vector, and a bias. The scores produced for each child separately are passed through a softmax to make $\overline{a}$ sum to 1. We experiment with adding in the question summary and child hidden states in order to be able to define task-specific relevance but we find that this does not help the model much. Learning syntactic and structural relevance proves enough to improve the parent's representational power. This tree *attention* method is inspired by the work of Zhou et al [6], who apply task-specific tree encoding to paraphrase detection and True-False QA.

Constituent MetaData Embeddings: Each constituent contains rich syntactic information which can be very discriminative when it comes to answer extraction. We use 5 categories of structural variables to learn a latent *structural* space for each node in the tree. The variables are part of speech, parent part of speech, height in the tree, length of spanning subtree, and relative position within its parent rule. We learn 32 dimensional embeddings for each and concatenate them together.

Attention Layer: We employ a BiDirectional Attention Flow [6] layer as defined by Seo, Minjoon, et al over question word-level represents and tree node representations (the output of the tree LSTM). The authors of the paper describe BiDAF as a means to encode context-question aware representations "without early summarization" [6]. The term bi-directional comes from the fact that both query-to-context representations, as well as context-to-query representations are produced. We first calculate a similarity matrix between

question hidden states and tree node hidden states by a dimension-normalized dot product: $\overline{c} \cdot \overline{q} / \sqrt{model_{dim}}$. Then we calculate a Context2Query (c2q) vector for each context tree node as the weighted average of each question word, with weights defined by a softmax over each question words' similarity to that context node. This represents the relevance of each question word to each context word. Query2Context (q2c) attention, conversely, represents the weighted relevance of the context words to each question word. We calculate q2c by first calculating for each tree node, the maximum similarity score over the question sequence. Then we take the softmax over this vector to produce a weighted relevance score for each context node. We take a weighted average of the node representations with these weights and repeated across the dimension of the flattened tree. We produce the final state vector for each tree node as the concatenation of the following vectors: $[h;\ c2q;\ c2q \circ h\ ;\ q2c \circ h\ ]$ where h represents the hidden state representation of the tree node and $\circ$ is element-wise multiplication.

Modeling Layer: We learn interactions between attended tree nodes through a modeling layer. The modeling layer passes the bidaf output through a bi-directional LSTM. To preserve linearity, we flatten the tree according to an in-order traversal which is commensurate to a left-right tracking LSTM through structure. We optionally include each node's metadata as part of the input to the modeling layer since it contains rich syntactic information which can help smooth over idiosyncrasies from the in-order flattening. We find this does not add much value, however.

Classification: We concatenate the bidaf output, the modeling layer output, the meta embeddings, as well as the meta modeled embeddings to a Multi-Layer Perceptron. The output of the model is a Softmax layer over each node. We define the loss criterion for answer *span* as

$$-log(\sum_{t}^{T} p_t F1(t,\ span)\ +\ \varepsilon)$$

which amounts to the negative log expected F1. We add a constant ε chosen to be 1e-5 to avoid numerical underflow from the log.

Dataset and Evaluation Metrics

SQuAD is a dataset for machine comprehension task, which is based on Wikipedia articles. Each dataset instance contains a question, a context and answers, which are always spans in the context. The training data consists of 90k questions and we evaluate our Self-Attention model and Tree-LSTM BiDAF model on 10k questions development dataset.
We measure our system's performance by two main metrics: F1 score and exact match score (EM). F1 reflects the average ratio of overlap between predicted span and true answer span. EM reflects whether predicted span is exactly equal to true answer span.

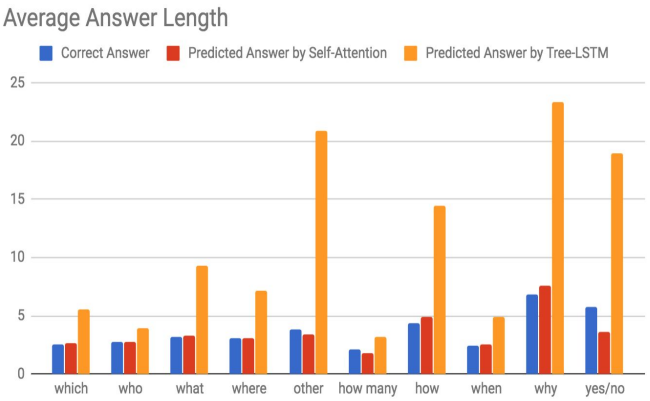
Results

The following tables compares the state-of-art model BiDAF with our two models.

	EM	F1
BiDAF [6]	0.68	0.77
Self-Attention Model	0.56	0.68
Tree LSTM Model	0.27	0.54

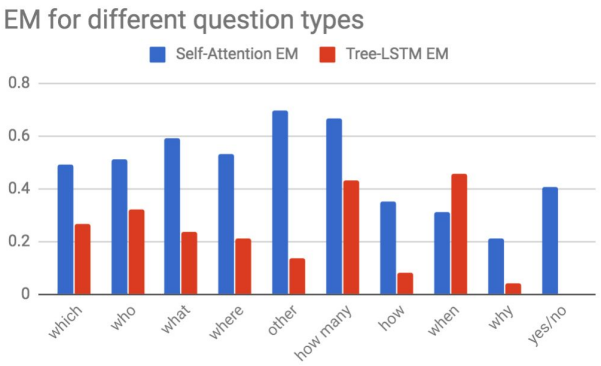
Error Analysis

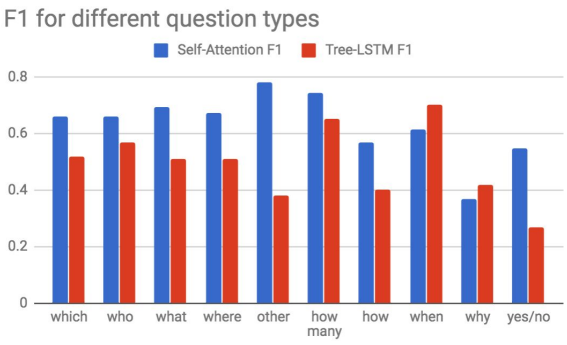
Question Type Analysis



We classify the questions into ten categories using first two words of the question. The following figure illustrates the average length of correct answers and predicted answers produced by our two models. The Self-Attention model generates reasonable answer lengths, while the Tree-LSTM BiDAF generates much longer lengths. The Tree-LSTM BiDAF model chooses maximum constituents so when the model is unsure about an answer, sometimes it reverts to an entire sentence. A better model and an answer length cap will improve this, yet the model demonstrates it acts conservatively under uncertainty.

The following two figures shows the F1 and exact match score for two models. Our two models produce similar F1 scores. But there is a gap between Self-Attention model and Tree-LSTM BiDAF model in exact match score. The Tree-LSTM model is capped in terms of EM by the EM of the best scoring node. Heuristic analysis we have done shows this is often an off by one error, where the constituent includes a determiner which the true answer does not have. We think this poor EM match is a result of not being allowed to overfit to the SQUAD data





Attention Analysis

In both Self-Attention model and Tree-LSTM BiDAF model, we use attention to fuse information from context and query. Context2Query attention measures which words in query are most relevant with respect to each context word. Conversely, Query2Context attention measures which words in context are most relevant with respect to each question word. In this section, we present the top attended words and their attention values with two examples.

An example from Self-Attention Model

Context: tesla served as a *vice president* of the american institute of electrical engineers , the forerunner (along with the institute of radio engineers) of the modern - day ieee , from 1892 to 1894 .
Query: what position did tesla hold in the american institute of electrical engineers ?
Correct answer: *vice president*

Query2Context Attention

Query words	Context words with top attention values
position	vice=0.04, institute=0.04, 1892=0.04
tesla	tesla=0.05 1894=0.03 american=0.03
hold	a=0.05, as=0.03, “,”=0.03, served=0.03

Context2Query Attention

Context words	Query words with top attention values
served	did=0.15, hold,=0.09, ?=0.08
vice	position=0.15, in=0.08, what=0.08
president	position=0.13, ?=0.08, what=0.08

Multi-Head Attention

CONTEXT: super bowl 50 was an american football game to determine the champion of the national football league (nfl) for the 2015 season . the american football conference (afc) champion denver -UNK- defeated the national football conference (-UNK-) champion carolina panthers 24 – 10 to earn their third super bowl title . the game was played on february 7 , 2016 , at levi -UNK- ' s stadium in the san francisco bay area at santa clara , california . as this was the 50th super bowl , the league emphasized the -UNK- " -UNK- golden anniversary -UNK- " -UNK- with various gold - themed initiatives , as well as temporarily suspending the tradition of naming each super bowl game with roman numerals (under which the game would have been known as -UNK- " -UNK- super bowl l -UNK- " -UNK-) , so that the logo could prominently feature the arabic numerals 50 .
Query: what color was used to emphasize the 50th anniversary of the super bowl ?
Correct Answer: *golden*

Attention Head	Top context words for query ‘color’	Possible Interpretation
#1	50=0.0117, ‘,’=0.0116, 50=0.0116	Number

#2	national=0.0085, league=0.0085, football=0.0085	Football
#3	area=0.0125, at=0.0110, arabic=0.0107	Location
#4	UNK=0.0131, UNK=0.0131, UNK=0.0131	Unknown words
#5	***gold=0.0129, santa=0.0123, golden=0.0117***	***color***
#6	UNK=0.0126, UNK=0.0126, UNK=0.0126	Unknown words
#7	themed=0.0124, conference=0.0124, initiatives=0.0123	???
#8	francisco=0.0102, levi=0.0102, american=0.0102	Location

An example from Tree-LSTM Model

Context: the victorian alps in the northeast are the coldest part of victoria . the alps are part of the great dividing range mountain system extending *east west* through the centre of victoria . average temperatures are less than 9 °c 48 °f in winter and below 0 °c 32 °f in the highest parts of the ranges . the state 's lowest minimum temperature of <unk> °c 10.9 °f was recorded at omeo on 13 june 1965 , and again at falls creek on 3 july 1970 . temperature extremes for the state are listed in the table below
Query: in what direction does the mountain system extend ?
Correct Answer: *east west*

Query2Context Attention

Query words	Context constituents with top attention values
direction	the northeast=0.0152 extending east west through the centre of victoria=0.0142 east west=0.0141
extend	extending=0.0307 extending east west through the centre of victoria=0.0206 system=0.0138
mountain	mountain=0.0328 the great dividing range mountain system=0.0274 again at falls creek on 3=0.0197

Context2Query Attention

Context words/phrases	Query words with top attention values
east west	direction=0.1601, mountain=0.1521, extend=0.1281, system=0.1158
the great dividing range mountain system	system=0.2559, mountain=0.2283, the=0.1321

Observations

1) In most cases, the self-attention model's multi-head attention can only do word-matching. It only assigns high attention values when two words are similar, without taking into account context. Also, it usually assigns high attention value to punctuations. This suggests that self-attention model's multi-head attention lacks effective feature learning.
2) Multi-Head attention is hard to interpret because it's composed of multiple layers and each layer contains multiple attention head. How each attention head works and how they collectively contribute to downstream task is difficult to understand. In the above example we visualize the 8 attention heads in 4th self attention layer in decoder. The 5th attention head attends to color related words. The 3rd and 8th attention heads attend to location related words. And the 4th and 6th heads attend to unknown words. Currently, there is no explicit way to enforce different heads to learn different attention perspectives.
3) The attention distribution is not as sparse as we expected. This phenomena is extremely severe in the self-attention model. Ideally, each word should only attend to few other words and extract most relevant information from long text. If the attention distribution is flat, then the model won't be able to locate and focus on important information. [18] discusses how to enforce sparsity on attention, which should be included in future work.
4) One advantage of Tree-lstm model is that it's able to form attention over constituents instead of just words. Constituents contain higher level representation than words, which encourage attention over an structural feature space, instead of just word-matching.

Tree-LSTM Constituent Analysis

Each tree contains roughly 200-300 nodes on average. We note that it is very adept at narrowing the selection down within its top k list, even if its top guess is incorrect. We present results from choosing the highest scoring node (by F1) among the k-best list.

Top K	F1	Recall	Precision
Top 1	0.55	0.82	0.66
Top 3	0.73	0.82	0.66
Top 10	0.88	0.97	0.80

Results shown here are metrics with respect to the highest scoring node as defined by the tree. These are within +/- 1% of the true results. We see that the model is fairly robust given that just including the top 3 candidate spans improves results materially. Thus, even when the model makes errors, it more often than not has the correct answer a few ranks below. We attribute this to the ability of the model to isolate the syntactic constraints provided by the question and only select spans which fit the bill. Here is a positive example where each answer is possible:

Question: between which two streets along kearney boulevard were wealthy african americans at one time residing ?
True: fresno street and thorne ave (68-73)
Top 5 Ranked Predictions:
Rank=1 fresno street and thorne ave
Rank=2 kearney boulevard between fresno street and thorne ave
Rank=3 kearney boulevard

Rank=4 fresno
Rank=5 the roughly half mile stretch of kearney boulevard between fresno street

Oftentimes it overpredicts the answer by choosing a parent node. Better isolation of the desired syntactic unit sought by the question would help prune the answer tree. We plan on implementing a Tree-LSTM on the question and performing BiDAF over tree2tree constituents to achieve this. An example of the over-prediction is:

Question: how are pharmacists regulated in most jurisdictions ?
True: separately from physicians (12-15)
Guessed: in most jurisdictions such as the united states , pharmacists are regulated separately from physicians .

Predominantly, the top k list includes many overlapping subtrees related to the question. This means that the remaining challenge of the model is not region identification but better answer extraction. We thought that encoding questions with a tree would help point the model to the exact constituent sought by the question (more in future work). The error for the question-passage pair:

Question: other than the us and britain what was the other main country that tesla had patents granted ?
True: canada
Guessed: the united states , britain , and canada

suggests insufficient question modeling. It is thrown off by questions such as “other than...”, since it fails to model questions beyond word-level associations. We believe that heuristic answer extraction and tree question encoding will help appreciably. It should be able to isolate “what was the other main country” as the constituent that requires an answer and reason that “other than the us and britain” negates the potential for the US or Britain to be an answer.

Lessons Learned & Reflections

One of the major lessons learned over the course of the project is controlling the pace and scope of work. Trying too many configurations, new features, etc., can lead to a lot of technical debt and an inability to diagnose the successes/shortcomings of a model. Once you get a deep model to converge, it is very easy to incrementally add features, testing each time. Originally, our eyes were too big for our stomachs which led us to overstretch. Optimizing code to be memory and time efficient pays tremendous dividends down the road. We optimized the tree code to a point where we reduced training time by ~30x from over a day to under an hour. This involved tree pruning, implementing unary closures, a batch-wise greedy queue system for processing child compositions, using flattened structures whenever possible, pre-allocating GPU memory, bucketing batches according to input dimensions to ruthlessly minimize over-padding. This allowed us to experiment in an agile manner and afforded us to explore moonshot ideas without much time lost (only souls).

A related lesson is to never assume anything. Just because something worked in a highly cited paper, or because PyTorch says it will take the softmax, don’t assume it will. Test everything and use print statements liberally. This goes hand in hand with test your code early and often but it cannot be overstated. There is nothing worse than writing a lot of code and then finding

out your model's performance has worsened. Another major lesson is the use of Git. Establishing rules of engagement with Git early on and sticking to them will avoid a lot of headaches later on. Code reviews would help and enable teammates to learn from each other and be on the same page. Machine learning research can be non-monotonic - you can work really hard for a week and accomplish nothing (in terms of downstream objective), and sometimes you can change one line of code and halve your loss. You must be level-headed and work methodically to deal with the natural natural vicissitudes of ML.

Future Work

Attention analysis indicates Self-Attention model doesn't capture attentions between query and context. The generated attention values are flat and it is hard to distinguish the target words. One method to improve this is to decrease the temperature value, which is used to normalize the attention values. Moreover, we can try other attention algorithm such as MLP attention. Future work for the tree model would involve adding even more lexical features for the tree nodes. Choosing a node in a constituency tree provides a great deal of opportunity to introduce inductive, structural bias to the model and improve the syntactic coherence of the model. More heuristics could also be introduced for the answer span extraction. Statistics regarding when/where/why and how the model errs on span boundaries make it possible to alter the trees to have them better respect span syntactic boundaries. Rather than changing the learning objective (choosing the best scoring node) to fit the divergence between answer spans and constituents, co-learning task-specific structures that both respect answer boundaries and learn to identify answers would be a very very cool research direction! We have implemented a version of the code in which we pass the question through the same tree encoding as the context. This model, however, degrades performance. More analysis is needed on why hierarchical modeling of the question worsens results. It might due to parameter sharing between question and context which limits expressiveness. (We tried using separate Tree LSTM weights for the question and passage but this overfits as well. Further hyperparameter exploration is needed since tree question modeling introducers more parameters.)

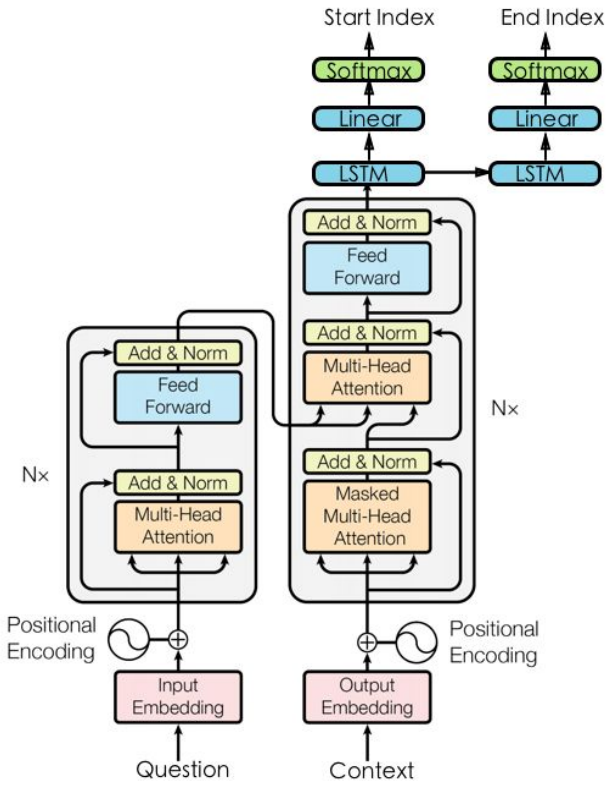
Conclusion

We build two models for the stanford question answering task: Self-Attention Model and Tree-LSTM BiDAF Model. These two models both aim at addressing the early summarization problem of existing recurrent models by allowing potentially distant, but related words, to directly interact. Our empirical experiments show that Self-Attention model has slightly better F1 score, but Tree-LSTM model has better structural feature learning and attention modeling ability. Both models are behind the state-of-the-art performance. Future work includes enforcing attention sparsity, better question topology, adding more lexical features, and an ensembled re-ranker.

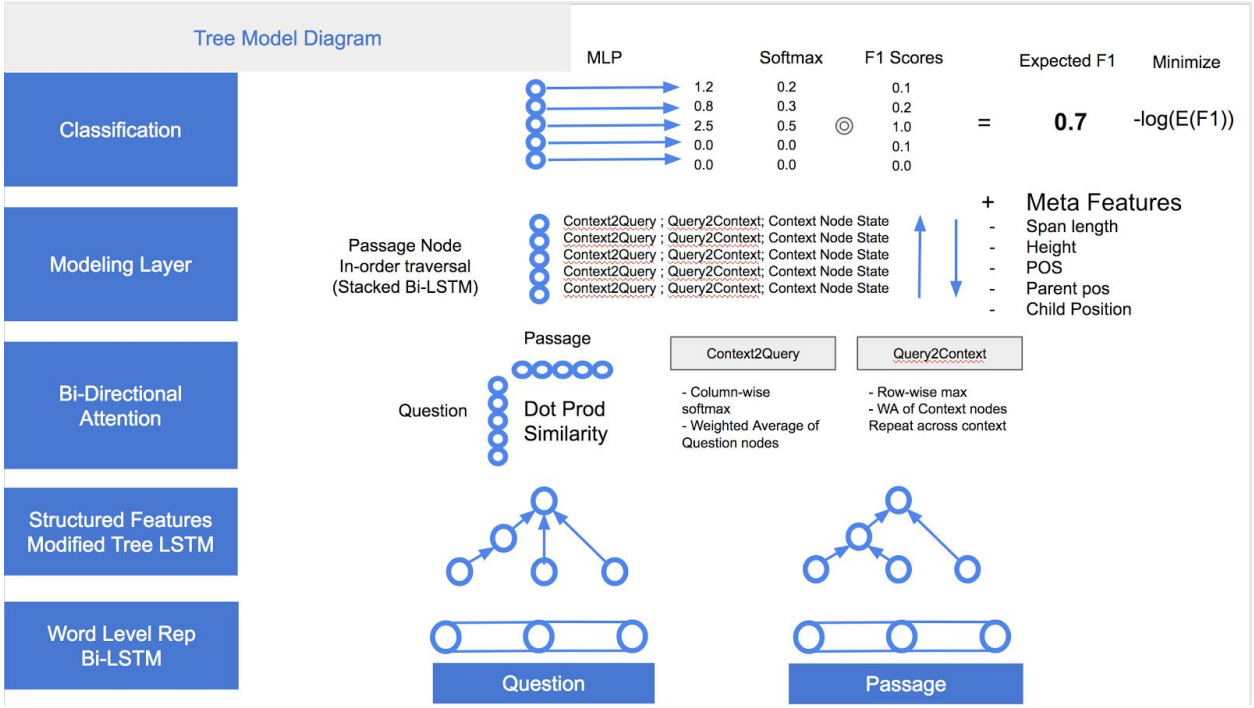
References

1. Hermann, Karl Moritz, et al. "Teaching machines to read and comprehend." Advances in Neural Information Processing Systems. 2015.
2. Trischler, Adam, et al. "NewsQA: A Machine Comprehension Dataset." arXiv preprint arXiv:1611.09830 (2016).
3. Rajpurkar, Pranav, et al. "Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).
4. Richardson, Matthew, Christopher JC Burges, and Erin Renshaw. "MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text." EMNLP. Vol. 3. 2013.
5. Chen, Danqi, Jason Bolton, and Christopher D. Manning. "A thorough examination of the cnn/daily mail reading comprehension task." arXiv preprint arXiv:1606.02858 (2016).
6. Seo, Minjoon, et al. "Bidirectional Attention Flow for Machine Comprehension." arXiv preprint arXiv:1611.01603 (2016).
7. Xiong, Caiming, Victor Zhong, and Richard Socher. "Dynamic Coattention Networks For Question Answering." arXiv preprint arXiv:1611.01604 (2016).
8. Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pp. 423-430.
9. Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. "Improved semantic representations from tree-structured long short-term memory networks." *arXiv preprint arXiv:1503.00075*(2015).
10. Kaiser, Lukasz, and Illia Polosukhin. "Attention is All You Need." arXiv preprint (2017).
11. Lin, Zhouhan, et al. "A structured self-attentive sentence embedding." arXiv preprint arXiv:1703.03130 (2017).
12. Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.
13. Zhou, Yao, Cong Liu, and Yan Pan. "Modelling Sentence Pairs with Tree-structured Attentive Encoder." arXiv preprint arXiv:1610.02806 (2016).
14. Socher, Richard, et al. "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection." Advances in Neural Information Processing Systems. 2011.
15. Cheng, Jianpeng, and Dimitri Kartsaklis. "Syntax-aware multi-sense word embeddings for deep compositional models of meaning." arXiv preprint arXiv:1508.02354 (2015).
16. Iyyer, Mohit, et al. "A Neural Network for Factoid Question Answering over Paragraphs." EMNLP. 2014.
17. Li, Jiwei, et al. "When are tree structures necessary for deep learning of representations?." arXiv preprint arXiv:1503.00185(2015).
18. Lin, Zhouhan, et al. "A structured self-attentive sentence embedding." arXiv preprint arXiv:1703.03130 (2017).

Appendix 1 Overview of Self-Attention Model [10]



Appendix 2 Overview of Tree LSTM Model



Appendix 3 Error Analysis

The following table summarizes the modes of errors. We randomly select 50 EM-incorrect answers and classify them into 6 categories.

Error type	BiDAF (%)	Self-Attention Model (%)	Tree-LSTM (%)
Imprecise answer boundaries	50	54	45
Syntactic complications and ambiguities	28	34	14
Paraphrase problems	14	4	20
External knowledge	4	0	10
Multi-sentence	2	0	0
Incorrect preprocessing	2	8	8