Tree LSTMs for Quiz Bowl Question Answering

Bryan Anenberg
Stanford University
Stanford, CA
anenberg@stanford.edu

Albert Chen
Stanford University
Stanford, CA
acc2015@stanford.edu

Abstract

Automatic question answering is a difficult task in NLP, because the model must learn to understand not only words but phrases. For this reason, bag-of-words models do not work so well, but recursive neural networks based on sentence parse trees have shown more promise. Here we extend the work of Iyyer et al.[5], who used a dependency tree RNN model (called QANTA) to answer quiz bowl questions. We apply the recently developed Tree LSTM model to the task, and show that it achieves better accuracy by 3% on history and literature questions. Furthermore, we introduce a Tree LSTM Tensor model to flexibly incorporate dependency relation information, and find that although it has better performance than QANTA on the literature set, the Tree LSTM model has the best performance overall.

1 Introduction

This paper presents and evaluates the performance of a system trained to answer quiz bowl questions. High school and college students compete every year in quiz bowl tournaments, in which teams race to answer questions about a variety of trivia topics. A quiz bowl question consists of four to six sentences of clues that progressively reveal more well-known and uniquely identifying facts about a single answer. For example, a question about an author may begin by describing a scene in an lesser known work, and end by giving the title of the author's most famous novel. A moderator reads the paragraph of clues, and the first team to answer correctly wins the point. In answering quiz bowl questions, a good model must learn to understand phrases in addition to words, and then combine information across sentences. This necessity to reason across syntactic structure and combine semantic meaning is common to a wide variety of NLP tasks. QANTA [5] takes a recursive neural network approach to the problem, using the dependency parse tree of a sentence as the scaffold. Here we apply a different model, Tree LSTMs, to the task. This model extends the usual linear LSTM to a tree structure. Each node of the tree stores a memory cell, which allows the model to conditionally preserve information from lower nodes in the tree. The Tree LSTM model has worked well for sentiment analysis [6], beating previous benchmarks set by recursive neural tensor networks. Here we will evaluate its effectiveness for question answering along with a few variants on the Tree LSTM model.

2 Background

Prior literature has taken a variety of approaches toward factoid question answering. For example, Boyd-Graber et al. [1] apply a generative bag-of-words approach to answer questions by incrementally revealing sentences. The most relevant prior work try to model compositional vector spaces with neural networks. Hermann et al. [4] and demonstrate that recursive neural networks which incorporate syntax can capture the compositional structure of sentences.

Motivated by the success of recursive neural networks, Iyyer et al. [5] proposed QANTA, a recursive neural network structured using the dependency parse tree of sentences. QANTA trains answer vectors in the same space as each node's hidden vectors, and combines information across sentences to produce better predictions in the end. QANTA outperforms Bag-of-Words and Information Retrieval baselines by understanding the compositional structure and semantics of the sentence, rather than simply recognizing named entities.

The Tree-LSTM model first introduced in Tai et al.[6] was shown to perform well on semantic analysis tasks. In an analogous manner to the usual LSTM, each node of the Tree-LSTM contains a memory which it uses to selectively incorporate information from each child. The Tree-LSTM possesses the flexibility to emphasizes different relations in the dependency parse tree. For example, the Tree-LSTM could be trained to emphasize the verb phrase as opposed to the noun phrase in a sentence. We expect that this type of structure, with memory and parent-child forget gates, will be useful in answering quiz bowl questions, in which some parts of a question provide more useful clues than other parts.

3 Approach

3.1 QANTA model

We first we describe the QANTA model, after which we will show the modifications needed to form our Tree LSTM model. The QANTA model is a recursive neural network based on the dependency parse tree of sentences. A dependency parse tree is a way to represent the grammatical relationships in a sentence. Each word of the sentence corresponds to a node, and the edges in the tree correspond to dependency relations such as 'direct object' or 'nominal subject.' So each node j of the tree corresponds to a word w, and is associated with a word vector x_w and a hidden vector h_j. For each leaf node of the tree, the hidden vector is computed directly from its word vector x_w via $h_{leaf} = f(W_v \cdot x_w + b)$. W_v applies a linear transformation to the word vectors and f is a nonlinearity, which in the QANTA model is the normalized `tanh` function.

As we move up the tree, the hidden vector for a parent node additionally incorporates the hidden vectors of its children. They are linearly transformed by a matrix corresponding to the relationship between that parent and child. In the equation below, $K(j)$ denotes the children of node j, and $R(n, k)$ is the type of dependency relation between node j and child k:

$$h_j = f(W \cdot x_j + b + \sum_{k \in K(j)} U_{R(j,k)} \cdot h_k).$$

This allows us to compute the hidden vector for the root node of each sentence, as well as all intermediate nodes. The parameters of the model are $\theta = (W_{r \in R}, W, W_e, b)$. W_e contains as its columns the word vectors x_w. Training this allows the model to learn word representations of the answers themselves in the context of question answering, and to take advantage of the appearance of answers words in questions for other answers. To train the model, a max-margin loss function is used, in which we encourage the answers word vectors to be close to the hidden vectors at nodes of their clues: $C(S, \theta) = \sum_{s \in S} \sum_{z \in Z} L(rank(c, s, Z)) \max(0, 1 - x_c \cdot h_s + x_z \cdot h_s)$.

In this equation, the cost for a sentence S is the sum over the cost at each node s. The cost at a node is a scaled max-margin loss of the correct word x_c against an incorrect word x_z, so we encourage the dot product between the hidden vector h_s and the correct answer to be higher than the dot product with any other answer. Ideally, Z would be the set of all incorrect answers, but this is too expensive to compute. Instead, it is a subset of randomly sampled incorrect answers. The term $rank(c, s, Z)$ is an approximation of the rank of the correct answer as scored by $x_w \cdot h_s$ over all words w. $L(r)$ scales the penalty to increase with r according to $L(r) = \sum_{i=1}^{r} 1/i$.

After training the neural network, predictions are made using multiclass logistic regression, taking the hidden vectors and word vectors as features. To make a prediction given the first k sentences of a question, we first compute the average x_w and h_j over all nodes for a sentence. The averages are then averaged across the k sentences and fed as the feature input for a multiclass logistic regression classifier.

3.2 Tree LSTM model

We are interested in improving upon the core composition equations of the QANTA model. We can apply Tree LSTMs in much a similar way as the QANTA model, using the same dependency tree structure, loss function, and prediction model, but changing the way we propagate information up the tree. Rather than including a matrix for each each dependency relation, we'll use the following structure reminiscent of the usual linear LSTM, but accounting for each child node:

$$\tilde{h}_j = \sum_{k \in C_j} h_k, \tag{1}$$

$$i_j = \sigma(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}), \tag{2}$$

$$f_{jk} = \sigma(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}), \tag{3}$$

$$o_j = \sigma(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}), \tag{4}$$

$$u_j = \texttt{tanh}(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}), \tag{5}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \tag{6}$$

$$h_j = o_j \odot \texttt{normtanh}(c_j), \tag{7}$$

$$\tag{8}$$

We will refer to this as the 'LSTM' model. Here $x_j \in W_e$ is the word vector for the word at node j. This is the Child-Sum Tree-LSTM model described in Tai et al.[6] The parameters are now $\theta = (W_e, \{W^{(x)}, U^{(x)}, b^{(x)}\}_{x \in \{i,f,o,u\}})$. We choose to train W_e for the same reasons as in QANTA. For initialization, we will take a similar approach as in QANTA, initializing W_e with word vectors from word2vec.

While there are many similarities in the framework of QANTA and Tree LSTMs, the key difference lies in how information is propagated up the tree. QANTA has the advantage of treating the various types of dependency relations differently. The Tree LSTMs inherit the forget property of LSTMs, in which a parent can choose to take input from certain children and forget the input of others. Furthermore, by associating each node with a memory cell, it has the ability to remember information many steps up the tree.

3.3 Tree LSTM variants

3.3.1 LSTMrel

We would like to allow the Tree LSTM to leverage the various types of dependency relations. This information can be included naturally by additionally indexing each U matrix by $R(j,k)$ so that the composition $U^{(x)}_{R(j,k)} h$ depends on the type of parent-child dependency relation. So $U^{(x)} \tilde{h}_j = \sum_{k \in C_j} U^{(x)} h_k$ becomes $\sum_{k \in C_j} U^{(x)}_{R(j,k)} h_k$. We call this the 'LSTMrel' model.

3.3.2 LSTMtensor

Finally, each relation may not require its own $U^{(x)}_{R(j,k)}$. That is, many dependency relations are similar and should be handled with similar U, and it is not efficient to model them separately. So we propose the 'LSTMtensor' model in which a variable number of latent relation matrices is learned for each type of gate. The key idea is to map each relation to a vector of length r, where r is smaller than the total number of relations. This length-r vector specifies the weights to combine the layers of a $r \times d \times d$ tensor to give the final $d \times d$ matrix for composition with the hidden vector. Each of the $d \times d$ layers in the tensor is considered a latent relation matrix. (r is a tuning parameter for the number of layers in the tensor, and d corresponds to the hidden vector dimension). Specifically, instead of $\sum_{k \in C_j} U^{(x)}_{R(j,k)} h_k$, the model uses

$$\sum_{k \in C_j} U^{(x)} \cdot R^{(x)}[R(j,k)] \cdot h_k$$

	Train (sentences)	Dev	Test	Vocab size	Unique answers	Unique relations
History	6780	472	897	19753	409	45
Literature	8502	650	1415	24384	445	45

Table 1: Data summary statistics.

	Non-vocab parameters		
	General	$r = 45, d = 100$	Comparison
QANTA	$(r+1)d^2 + d$	460100	100%
LSTM	$8d^2 + 4d$	80400	17.5%
LSTMrel	$4(r+1)d^2 + 4d$	1840400	400%
LSTMtensor	$4(d^2+d) + (r+d^2)\sum_{x\in\{i,f,o,u\}} r^{(x)}$	412065 (large) 261390 (small)	89.6% 56.8%

Table 2: Comparison of model sizes relative to QANTA. The number of relations is r, and the vector dimension is d. As used later in this paper, LSTMtensor large is with $(r^{(i)}, r^{(f)}, r^{(o)}, r^{(u)}) = (10, 5, 2, 20)$ and LSTMtensor small is with $(5, 5, 2, 10)$. Given our parameter settings, the LSTM model has about 1/6 as many non-vocab parameters as QANTA, and LSTMrel has 4 times as many. LSTMtensor trades off in between.

where $x \in \{i, f, o, u\}$. $U^{(x)}$ is the $r^{(x)} \times d \times d$ tensor. The $R^{(x)}$ matrix has dimension $r^{(x)} \times$ (# relations), and relation $R(j, k)$ indexes into one of the columns of $R^{(x)}$.

Note that LSTM and LSTMrel are special cases of this LSTMtensor model where the number of matrices is fixed at 1 and 45 and the $R^{(x)}$ matrix is fixed appropriately. We can think of the LSTMtensor model as an interpolation of the other two models, which is no more powerful than the LSTMrel model, but may be a useful reduction of parameters to a more natural model.

All of the models were implemented in Python, and training was parallelized across sentences. We used adagrad and backpropagation through structure to train the models.

4 Experiments

4.1 Data

Quiz bowl questions were provided by Iyyer et al.[5]. Table 1 shows an overview of the data summary statistics. Questions fall into two categories, literature and history. As in the QANTA paper, we train separate models for each category, because a history question is unlikely to provide much information to help identify a novel, and vice versa. There are more literature questions than history questions, and we use a 82/6/12 train/dev/test split. Note that the data size is in the thousands, which is smaller than those on which neural networks typically perform well. Nevertheless, the QANTA model still significantly outperforms the BOW baselines, so we will be able to draw meaningful comparisons to QANTA.

There are 19753 vocabulary words for history and 24384 for literature, and because we train them in the model, each one contributes d parameters, where d is the word vector and hidden vector dimension. These constitute the majority of the parameters. Table 2 compares the model sizes of the rest of the parameters. The LSTM model is the most economical, only a sixth the size of the QANTA model, whereas LSTMrel is four times as large. LSTMtensor trades off in between.

4.2 Evaluation

We evaluate our models based on accuracy, the number of correct predictions over total predictions, which is the metric that was optimized in the QANTA paper. When quiz bowl is played, there is an incentive to answer early in order to beat the other team to the buzzer. However, if the wrong answer is given, there is a point penalty and the other team has a chance to hear the rest of the question uncontested. Therefore, we will calculate the model accuracy given just the first sentence, the first two, and the entire question. We will also examine the model accuracy averaged over all sentence

Model	Dataset	λU	λW	λx	Dimension d	Sentence accuracy (%)	Unregularized accuracy (%)
LSTM	History	1e-4	1e-5	0	90	51.3	50.4
	Literature	0	1e-4	0	100	53.8	52.9
LSTMrel	History	1e-5	0	0	100	52.5	51.9
	Literature	0	1e-5	0	100	53.6	52.5

Table 3: Optimal hyperparameters. λ denotes the regularization for each respective parameter. Sentence accuracy is the accuracy of each sentence treated independently, using the hidden vector at the root of each sentence. Unregularized accuracy is the accuracy of the model with $d = 100$ and all λ values set to 0.

positions, to get a big picture of the 'full' model accuracy. Each of these metrics captures a slightly different aspect of each model, and is useful in understanding how each one behaves. For example, a model which has poor accuracy given the first sentence may be able to combine information across sentences well to attain high question-level accuracy.

4.3 Baselines

We use bag-of-words models and QANTA as our baselines. The 'BOW' model computes unigram indicators as uses them as features for logistic regression. The 'BOW-DT' model additionally incorporates indicators for the dependency relations in the dependency parse tree of a sentence. Neither of these models is expected to perform well, because at best, they associate key words with answers but miss the nuances in syntactic structure. We are most interested in drawing comparisons to the QANTA model, which uses a full recursive neural network infrastructure just as our LSTM-based models do.

4.4 Model tuning

4.4.1 LSTM and LSTMrel

There are quite a few hyperparameters to tune in LSTM and LSTMrel, but most important are the regularization constants and the hidden vector dimension d. The hidden vector dimension equals the word vector dimension. Greff et al.[3] argue that it is sufficient to vary each parameter one at a time when tuning an LSTM because higher order interactions are less important. Thus, we optimized the parameters separately. Search for λs was done over the set $\{0, 1e\text{-}6, 1e\text{-}5, 1e\text{-}4\}$. Search for word and hidden vector dimension was done over $\{60, 70, 80, 90, 100\}$. Next, we trained another model that used the optimal settings for each parameter. Table 3 shows the optimal hyperparameters and their performance on the dev set. Finally, we selected the model with the highest sentence level accuracy for evaluation on the test set. In contrast to the results reported in Iyyer et al. [5], regularization can improve model performance slightly, by about 1% on the dev set over the unregularized model. However, this only holds up to a point; regularization greater than 1e-3 drastically hurt performance on both train and dev sets.

We observe that it is beneficial to regularize U and W, but not the word vectors x. This is interesting since we would expect that regularizing the word vectors would prevent the model from overfitting to the small training data. Perhaps using large λ shrinks toward 0 too aggressively, away from their meaningful initialization. It may also too strongly prevent the word vectors from training. One justification for training the word vectors rather preserving the default word2vec initialization is that quizbowl questions contain specialized jargon, in which words are used differently than in normal conversation. Furthermore, because answer vectors are themselves words, we are able to learn better representations of the answer vector from other questions which contain the answer word.

4.4.2 LSTMtensor

To tune the LSTMtensor model, we were mainly concerned with picking the tensor dimension $r^{(x)}$ for each type of gate. We set $d = 100$ and all λs to 0, and tried a low and high setting for the tensor dimensionality. The settings were chosen by analyzing the U matrices learned by the LSTMrel

LSTMtensor	$r^{(i)}$	$r^{(f)}$	$r^{(o)}$	$r^{(u)}$	Sentence accuracy (%)
History	10	5	2	20	44.7
	5	5	2	10	49.8
Literature	10	5	2	20	44.8
	5	5	2	10	47.8

Table 4: LSTMtensor hyperparameter tuning. The tensor model achieves worse dev error than LSTM and LSTMrel.

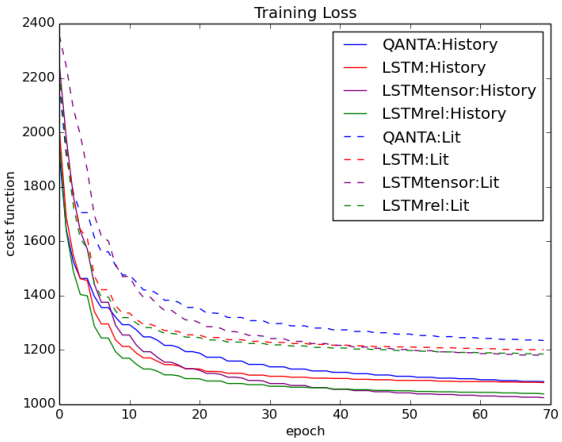

Figure 1: The minimum training cost so far as a function of epochs. All four models train very quickly on both data sets, converging after 30 epochs. LSTMrel appears to be the most powerful model, closely followed by LSTM, as they achieve the lowest cost values.

model, in an attempt to pick a natural tensor dimension for each type of gate. See Figure 3 in Section 5 for more details.

Table 4 shows the results on the dev set. Surprisingly, we achieve worse accuracy than both LSTM and LSTMrel. The lower-dimensional model achieves better performance on the dev set, but it still gets under half the sentences correct. The difference in dev error between the small and large models is quite large (3-5%), so the performance is sensitive to tuning the dimensions. We expect that additional parameter tuning in the low range will yield better results.

4.4.3 Training cost and error

Figure 1 is a plot of the minimum training cost (max margin loss + regularization) for the four models at their optimal hyperparameters. Note that LSTM and LSTMrel include regularization cost, but the overall trend is clear: the models learn quickly, with cost sharply decreasing until epoch 10, and not changing much after epoch 50. QANTA is the least powerful model, because it trains the slowest and has the highest cost for both history and literature. In general, the literature cost is higher than the history cost, indicating that literature is a harder subject.

Figure 2 shows the model error (100%-sentence level accuracy) against training epochs. All four models train quickly, reaching close to optimal dev error after about 30 epochs. The trends are similar for history and for literature. QANTA trains the slowest and converges to higher dev error. LSTM and LSTMrel train much faster, and attain lower dev error. The LSTMtensor model falls somewhere in between. The error on the train set is close to 0, indicating that all the models are powerful enough to overfit.

5 Results

Table 5 shows the final model accuracies, evaluated on the held out test set. We observe that in general, the LSTM model performs the best, beating QANTA by more than 3% over all sentences (the 'full' column) for both history and literature questions. In fact, the LSTM model also beats QANTA given just the first sentence, the second sentence, or the entire question. LSTM is the best model overall.

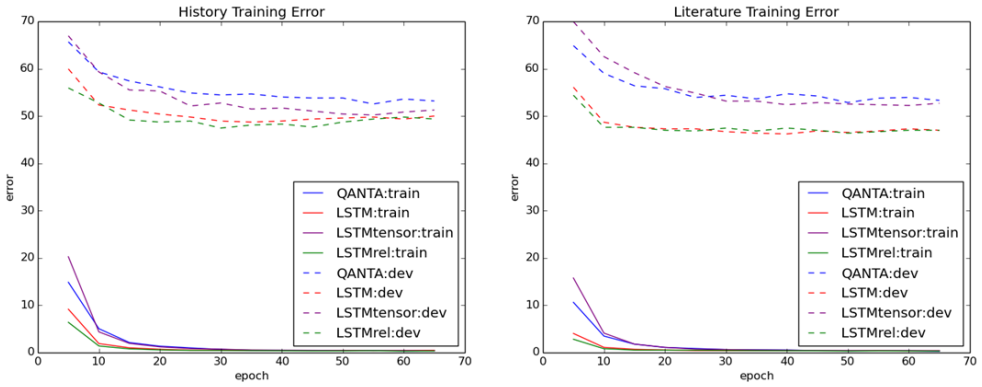

Figure 2: Sentence level error on train and dev set, plotted against training epochs. All three models attain nearly 0 train error after 30 epochs. The LSTM and LSTMrel models generalize best to the dev set.

	History				Literature			
Model	Pos 1	Pos 2	Full	Question	Pos 1	Pos 2	Full	Question
BoW	15.6	29.7	35.8	50.0	12.6	24.3	30.2	46.6
BoW-DT	20.3	37.0	47.2	66.6	15.9	34.3	42.9	71.2
QANTA	32.8	**53.1**	60.1	81.8	21.4	40.1	52.3	84.1
LSTM	33.3	**53.1**	**63.2**	**85.9**	**25.9**	**45.3**	**55.7**	85.8
LSTMrel	**34.4**	48.4	61.3	83.9	**25.9**	42.7	55.3	**87.1**
LSTMtensor	33.3	49.0	59.3	81.3	22.7	41.4	54.8	84.5

Table 5: Final model accuracies on the test set. Accuracy at the question level, after one sentence, two sentences, each sentence, and entire question are revealed. 'Full' refers to the average accuracy over all sentences. BOW is logistic regression on unigram indicators. BOW-DT is with unigram and dependency relation indicators. LSTM achieves a 3% gain in full accuracy over QANTA. Surprisingly, neither LSTMrel nor LSTMtensor improves upon LSTM.

Interestingly, the LSTMrel model does well given just the first sentence, and attains the highest question-level accuracy for literature questions (87.1%). It performs slightly worse than LSTM in general, but still beats QANTA, especially on the more difficult literature set. Iyyer et al.[5] observe that literature questions are harder to answer than history questions, because they often require knowledge of plot events or lines of poems which never appear in the training data. It is fitting that the more powerful LSTM-based models would show more improvement on these harder questions. Curiously, the models attain better accuracy for literature than history when given the entire question. Even BOW-DT performs 5% better on literature than history when the full question is revealed. We suspect that this is because many literature questions name the author's most famous work in the last sentence, whereas there is no similarly canonical clue to reveal for history questions.

One surprising observation from the table is that LSTMtensor performs worse than LSTMrel, which it was designed to improve. It also performs worse than QANTA on history questions, but better on literature. As it is the most flexible model, we suspect that with additional tuning of the tensor dimensions $r^{(x)}$, it can achieve better results.

5.1 Qualitative analysis

To understand how the LSTMrel model works, we examine the trained U matrices for each of the gates. For each type of gate, Figure 3 shows the pairwise distances between the 45 relation matrices. Each of the four plots has a unique pattern, indicating that the gates leverage the dependency relations differently. From the upper right plot, we observe that nearly all output matrices are the same, and that the one for DET ([2] determiner, i.e. words like 'the' and 'which') is most different from

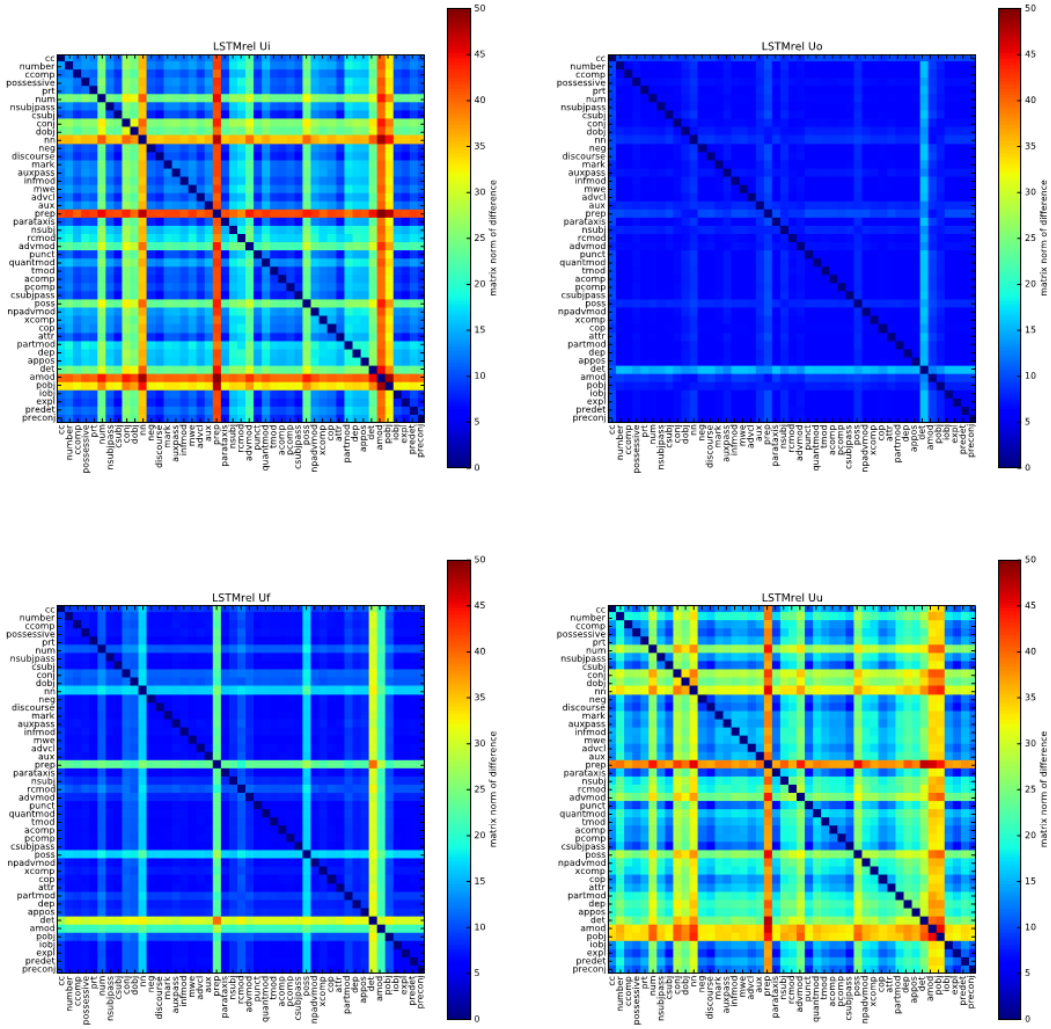

Figure 3: Pairwise distances between relation matrices U in the LSTMrel history model. The distance between A and B is defined as norm$(A - B)$. Note that there are essentially only two kinds of output matrices, whereas input and memory matrices are more diverse. This plot motivates the LSTMtensor model, in which we learn a restricted number of matrices for each type of gate. These plots also confirm our intuition about which types of dependency relations contain relevant information. The plots for the LSTMrel literature model are similar.

the rest. The similarity between the $U^{(o)}$ relation matrices suggests that the children should all be treated equally when computing the output gate.

In contrast, the forget matrix distances reveals that children related by POSS (possession modifier), PREP (prepositional modifier), NN (noun compound modifier), AMOD (adjectival modifier), and DET (determiner) need to be treated differently. The first four are common types of modifiers which likely do not bear too much meaning on their own, but contribute to the meaning of their context. We observe that the input and memory cell matrices show the most variation.

These matrix distance plots motivate the LSTMtensor model as a technique to reduce the number of unique relation matrices to learn. The tensor dimension $r^{(x)}$ reflects the number of basis latent matrices required to express the 45 relation matrices for each type of gate ($x \in \{i, f, o, u\}$). Setting $r^{(x)} < 45$ asserts that there exists fewer than 45 unique latent matrices to describe the 45 relations. This is especially evident in the case of the output matrices $U^{(o)}$.

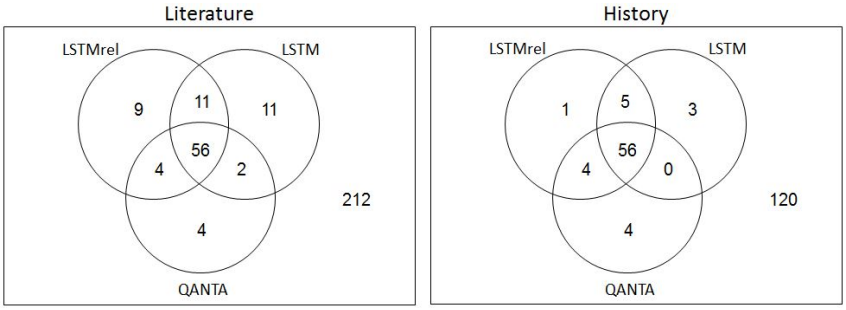

Figure 4: Venn diagram of correct model predictions based on the first sentence of a question. Wrong predictions are outside the circles. In general, the models make the same predictions. Note that LSTMrel is more similar to QANTA than LSTM is, presumably because both rely on dependency relation information.

Q1	he wrote about his experiences in the spanish civil war in beneath your clear shadow and other poems
A	**octavio paz**, carlos fuentes, gabriela mistral
Q2	he penned a dramatic poem about a sick knight who can only be cured by a maiden sacrificing her life for him the golden legend
A	lysistrata, **henry wadsworth longfellow**, william wordsworth
Q3	frederick nicolai rewrote this work to give it a happy ending prompting its author to write a poem where nicolai defecates on the grave of its main character
A	**the sorrows of young werther**, **the sorrows of young werther**, ben jonson
Q4	poems like hibiscus on the sleeping shores and le monocle de mon oncle are typical of this poet 's early work characterized by an intricate style and tropical imagery also seen in the poem sunday morning
A	william wordsworth, alexander pope, **wallace stevens**
Q5	one resident of this state served as secretary of defense for most of bill clinton 's second term
A	james buchanan, franklin pierce, **maine**
Q6	character in this play won a diana trophy for her archery prowess during college
A	ethan frome, death of a salesman, miss julie
Q7	it opens at noon when the sun was accustomed to awaken both lap dogs and lovers
A	arthur rimbaud, arthur rimbaud, arthur rimbaud

Table 6: Example questions and model predictions. Predictions of LSTM, LSTMrel, and QANTA are shown in order from left to right, with the correct answer highlighted in blue. The correct answers are impressive. Q3 and Q7 reveal the difficulty of discerning whether the answer is an author or a work, and Q5 relies on correctly understanding the meaning of 'state' as a noun. Other questions show the reliance on specialized information, which the model is sometimes able to pick up on.

Figure 4 illustrates similarity in correct predictions between the models QANTA, LSTM, and LSTMrel. The models generally predict the same answers, but vary on a significant fraction of questions. The large intersection between LSTM and LSTMrel indicate that they correctly predict many answers that QANTA does not. Furthermore, LSTMrel has a larger intersection with QANTA than LSTM does, implying that they are more similar. We believe this is because both LSTMrel and QANTA leverage dependency relation information.

Table 6 examines the first sentences of a illustrative questions and compares the predictions of QANTA, LSTM, and LSTMrel. Notice that QANTA is misled in Q3 to predict an author rather than a work, but in Q5 the LSTM-based models fail to recognize that the answer is a U.S. state. The word 'state' has multiple meanings but is represented by only one word vector, so we don't expect LSTM, which doesn't have relation information, to be able to disambiguate the meanings.

Next we'd like to understand how the models are making predictions. In Figure 5, we examine the dot product of each node's hidden vector with the correct answer for a literature question and

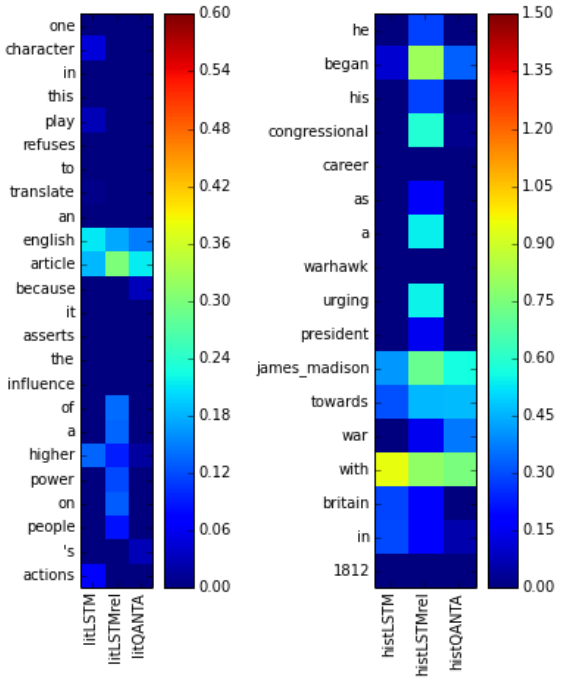

Figure 5: Dot product of each node's hidden vector with the correct answer. On the left, LSTM and LSTMrel correctly predict 'An Enemy of the People', but QANTA guesses 'Catch 22.' This question is difficult because it contains no named entities, but the LSTM-based models are using the plot summary to arrive at a correct guess. On the right, LSTMrel correctly predicts 'Henry Clay', but the other models guess 'James Monroe.' This is reflected in the dot products with the correct answer vector.

a history question. The literature question on the left shows that both LSTM and LSTMrel, which make correct predictions, are able to infer the name of a play from a description of a scene that contains no named entities. Note that none of the models has high activation with the word 'play', because simply knowing that answer is a play should not strongly indicate the answer. Yet each model predicts the name of a play and not a playwright. This suggests that models are not answering as a human would, by first narrowing down to plays, and then thinking of plays that match the plot description. Instead, they learn to recognize the answer directly by how the question is asked.

For the plot on the right in Figure 5, the LSTMrel model makes the correct prediction of 'Henry Clay,' whereas the other models guess 'James Monroe.' While Monroe did serve an early term in Congress and he succeeded James Madison as president, it is Clay who is better known as a congressman and prominent War Hawk. The LSTMrel model shows strong syntactic reasoning abilities beyond what Bag-of-Words models could do.

6 Conclusion

We present a successful application of Tree-LSTMs to factoid question answering that outperforms the bag-of-words and recursive neural network (QANTA) baselines. The Tree-LSTM improves upon how information propagates from the leaf nodes to the root node in each sentence by storing a memory cell and scaling the influence of the input and children at each node. We find that of the three proposed models (Tree-LSTM, Tree-LSTMrel, and Tree-LSTMtensor) the Tree-LSTM yields the best performance: a 3% improvement upon QANTA in both history and literature.

Future directions of this work include training the models on a larger data set with more quiz bowl questions and optimizing the choice of the tensor dimension $r^{(x)}$ for each type of gate by cross validation. We expect that the more powerful Tree-LSTMrel and Tree-LSTMtensor models will scale the best to larger datasets. Furthermore, we are interested in tuning other aspects of the model, beyond the composition equation framework. We could develop a zero shot learning variant of the system to predict answers which the model was not trained on. This extension is feasible since answers exist in the same vector space as the sentence representations. It would also be fun to build a competition-ready version of this model, with the ability to answer questions across multiple categories and decide when to 'buzz-in,' so that it could compete against real human players.

References

[1] Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daumé, III. Besting the quiz master: Crowdsourcing incremental classification games. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1290–1301, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[2] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

[3] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *CoRR*, abs/1503.04069, 2015.

[4] Karl Moritz Hermann and Phil Blunsom. The Role of Syntax in Vector Space Models of Compositional Semantics. In *Proceedings of ACL*, August 2013.

[5] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Empirical Methods in Natural Language Processing*, 2014.

[6] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015.