Recursive Tree-Structured Self-Attention for Answer Sentence Selection

Khalil Mrini, Emilia Farcas, and Ndapa Nakashole

University of California, San Diego, La Jolla, CA 92093

{khalil, efarcas, nnakashole}@ucsd.edu

Abstract

Syntactic structure is an important component of natural language text. Recent top-performing models in Answer Sentence Selection (AS2) use self-attention and transfer learning, but not syntactic structure. Tree structures have shown strong performance in tasks with sentence pair input like semantic relatedness. We investigate whether tree structures can boost performance in AS2. We introduce the *Tree Aggregation Transformer*: a novel recursive, tree-structured self-attention model for AS2. The recursive nature of our model is able to represent all levels of syntactic parse trees with only one additional self-attention layer. Without transfer learning, we establish a new state of the art on the popular TrecQA and WikiQA benchmark datasets. Additionally, we evaluate our method on four Community Question Answering datasets, and find that tree-structured representations have limitations with noisy user-generated text. We conduct probing experiments to evaluate how our models leverage tree structures across datasets. Our findings show that the ability of tree-structured models to successfully absorb syntactic information is strongly correlated with a higher performance in AS2.

1 Introduction

Motivation. Natural language text is characterized by structure. For instance, syntactic parse trees decompose a sentence into syntactic groups, which in turn are decomposed recursively until we get to single-word spans. Therefore, syntactic parse trees have a varying number of levels that can be accurately represented by recursive model architectures.

Tree-structured LSTM networks (Tai et al., 2015) are the recursive extension of LSTM networks (Hochreiter and Schmidhuber, 1997), and allow for syntactic trees to be represented hierarchically. Tree-LSTMs and bidirectional Tree-LSTMs

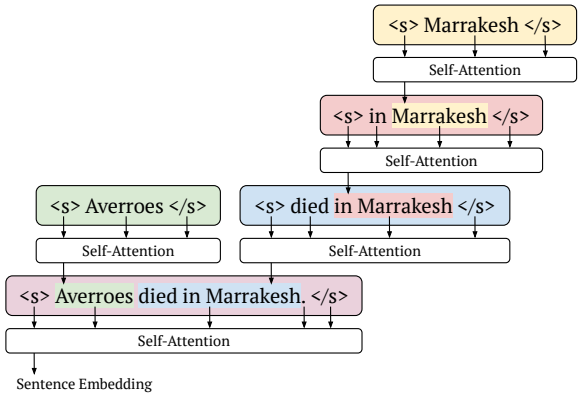

Figure 1: Embedding a sentence with our proposed recursive tree-structured self-attention using the corresponding constituency parse tree. There is only one set of parameters for the recursive self-attention.

(Teng and Zhang, 2017) do not represent sequence position information, whereas the hybrid neural inference networks (Chen et al., 2017a) represent sequence position information separately from tree-structured hierarchical information.

Tree-structured models have been applied to the tasks of natural language inference (Chen et al., 2017a), sentence pair similarity (Tai et al., 2015), dependency parsing (Kiperwasser and Goldberg, 2016), and text embeddings (Mrini et al., 2019). In this paper, we consider the problem of Answer Sentence Selection (AS2), where the goal is to predict for a question-sentence pair whether the sentence contains an answer to the question. Given that tree-structured models have performed strongly on a task that takes a sentence pair as input – sentence pair similarity, we hypothesize that tree structures can help in AS2, another sentence pair task.

The most recent top-performing model architectures for Answer Sentence Selection have been based on the self-attention transformer architecture (Vaswani et al., 2017). Three of them (Lai et al., 2019; Garg et al., 2019; Tran et al., 2020)

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 4651–4661
August 1–6, 2021.

use transfer learning on large AS2 datasets; another one (Laskar et al., 2020) uses direct fine-tuning on pre-trained transformer-based language encoders, whereas all three use pre-trained BERT (Devlin et al., 2019) and/or RoBERTa embeddings (Liu et al., 2019).

Contribution. We investigate whether tree structures are useful for AS2. We introduce the *Tree Aggregation Transformer*: a novel *recursive and tree-structured self-attention model* for Answer Sentence Selection. We use the syntactic parse trees of questions and candidate answer sentences to model them in a tree-structured way. We then form representations for questions and candidate answers using one additional self-attention layer in a recursive, bottom-up fashion, as shown in Figure 1. We learn syntactic embeddings to represent hierarchical order and phrase-level syntactic information. We find in an ablation study that our learned syntactic embeddings improve performance.

Without using AS2 datasets for transfer learning, our model establishes a new state of the art for the clean versions of TrecQA and WikiQA, two widely used benchmark datasets in question answering and AS2. Our tree-structured self-attention matches or exceeds the state of the art – which is fine-tuning on RoBERTa – on 2 out of 4 Community Question Answering (CQA) datasets. We conduct experiments for 3 probing tasks to establish what information our models leverage to increase performance, and likewise what they fail to leverage when they do not exceed baselines. We find that tree-structured representations that successfully absorb the provided syntactic information consistently perform better than baselines. Our probing task results suggest that there is more work to be done for tree structures to adapt to noisy user-generated text.

2 Related Work

Tree-structured Transformers. To the best of our knowledge, our method is the first to introduce tree self-attention to Answer Sentence Selection. There is a growing body of work incorporating tree structures in self-attention for a range of other NLP tasks.

Nguyen et al. (2019) introduce a transformer-based encoder-decoder that incorporates tree-structured attention. The tree-structured attention is accumulated hierarchically. A token in the tree has as many representations as overall children, therefore it is first accumulated in a bottom-up fashion (vertically), and then horizontally to compute a token's representation. Their model is not recursive and uses different parameters for each level. The authors evaluate their model in machine translation and text classification.

Sun et al. (2020) develop a tree-structured transformer encoder-decoder architecture for code generation. Here, the tree structure is based on the code syntax. The model uses character-level embeddings as input.

Harer et al. (2019) introduce Tree-Transformer: a model with a tree convolution block for correction of code and grammar. Wang et al. (2019) propose a model of the same name, where the model learns syntactic parse trees in an unsupervised manner. The model uses up to 12 layers of non-recursive self-attention on top of a pre-trained BERT.

Ahmed et al. (2019) introduce Constituency and Dependency Tree Transformer models, largely inspired by the Constituency and Dependency Tree-LSTM models (Tai et al., 2015) and RvNN models (Socher et al., 2011, 2012, 2013). On 4 datasets of semantic relatedness, natural language inference and paraphrase identification, their transformer models achieve performance on par with Tree-LSTM models, and do not set a new state of the art. The authors use two convolution layers to form a parent representation from the corresponding children. Their model does not learn an explicit syntactic representation, and the authors do not analyze the fluctuating results.

Answer Sentence Selection (AS2). The recent state-of-the-art models in the AS2 task all use transfer learning from large-scale datasets, and do not incorporate syntactic information. All of them use a standard linear (or sequential) input format, where the first input sentence is the question and the second is the candidate answer.

Lai et al. (2019) introduce the Gated Self-Attention Memory Network (GSAMN). It combines gated attention (Dhingra et al., 2017; Tran et al., 2017), memory networks (Sukhbaatar et al., 2015) and self-attention (Vaswani et al., 2017) in one model. The authors use transfer learning with their Stack Exchange QA dataset.

Garg et al. (2019) propose the *TandA* method: Transfer and Adapt. The method is simply fine-tuning directly on a pre-trained BERT or RoBERTa model. The *transfer* step is transfer learning: fine-tuning a large pre-trained BERT or RoBERTa on the ASNQ dataset: a large-scale answer sentence

selection dataset extracted from Google's Natural Questions (Kwiatkowski et al., 2019). The second step is to *adapt* the language model fine-tuned for answer sentence selection to the smaller, target benchmarks TrecQA and WikiQA.

Tran et al. (2020) build upon the work of Lai et al. (2019). They propose to use a neural Turing machine (Graves et al., 2014) as a controller for the memory network, instead of the gated attention that Lai et al. (2019) use. Like Garg et al. (2019), they use the ASNQ dataset for transfer learning.

Laskar et al. (2020) achieve state-of-the-art results on a wide range of QA and CQA datasets by directly fine-tuning on the target datasets, without transfer learning from an external large-scale dataset. They show results for two methods: the first trains a self-attention layer while freezing pre-trained language model layers, and the second directly fine-tunes on the language model.

3 Tree Aggregation Transformer for Answer Sentence Selection

In the AS2 task, the input is a pair of sentences, where the first one is the question and the second is a candidate answer. This is a binary classification problem on whether or not the candidate answer sentence contains an answer to the question. We therefore design our model to form a representation of the question and a representation of the candidate answer, in a bottom-up tree aggregation fashion.

Semantic and Syntactic Representation. We define a token embedding in our input representation as the concatenation of a semantic embedding and a syntactic embedding. The semantic embedding is a projection of the token embedding from a given pre-trained language model, whereas the syntactic embedding contains information from part-of-speech tags, syntactic categories, and the level within the syntactic parse tree.

The syntactic embedding is the sum of three learned embeddings. The first embedding represents the token's tag – a part-of-speech tag if the token is a word, or a syntactic category if the token is a classification or separator token. The second embedding represents the token's level within the tree, inherited from the head of the token's constituent span. Our recursive model allows to represent sentences with as many tree levels as the corresponding syntax tree has. The third embedding represents the position of a token within the constituent span, as seen in the example in Figure 2. This position embedding puts the token within its span context, whereas the position embedding of the semantic (language model) embedding puts the token within the context of the question-sentence pair.

More formally, given a token t, its language model embedding $\mathbf{x_t}$, its position index p_t, its part-of-speech tag or syntactic category s_t, and its tree level l_t, the token's semantic embedding $\mathbf{e_t}$ and syntactic embedding $\mathbf{n_t}$ are as follows:

$$\mathbf{e_t} = \mathbf{W_1} * \mathbf{x_t} + \mathbf{b_1} \tag{1}$$

$$\mathbf{n_t} = \mathbf{W_2}\left[\mathbf{E^s}\left[s_t\right] + \mathbf{E^p}\left[p_t\right] + \mathbf{E^l}\left[l_t\right]\right] + \mathbf{b_2} \tag{2}$$

where $\mathbf{W_1}$, $\mathbf{W_2}$, $\mathbf{b_1}$, $\mathbf{b_2}$ are learned, and $\mathbf{E^s}$, $\mathbf{E^p}$ and $\mathbf{E^l}$ are learned embedding layers, respectively for the part-of-speech tag or syntactic category, the position index, and the tree level.

Recursive Self-Attention. We add 1 layer of recursive self-attention layer on top of the language model layers. The recursive self-attention layer has separate attention distributions $\mathbf{a_t^e}$ and $\mathbf{a_t^n}$ for the semantic embedding $\mathbf{e_t}$ and syntactic embedding $\mathbf{n_t}$:

$$\mathbf{a_t^e} = \text{softmax}\left(\frac{\mathbf{q_t^e} * \mathbf{K^e}}{\sqrt{d_e}}\right) \tag{3}$$

$$\mathbf{a_t^n} = \text{softmax}\left(\frac{\mathbf{q_t^n} * \mathbf{K^n}}{\sqrt{d_n}}\right) \tag{4}$$

where d_n and d_e are the dimensions of the query and key vectors for the semantic and syntactic embeddings respectively, and $\mathbf{K^e}$ and $\mathbf{K^n}$ are the learned matrices of key vectors of input tokens. $\mathbf{q_t^e}$ and $\mathbf{q_t^n}$ are the query vectors for the token t, such that:

$$\mathbf{q_t^e} = \mathbf{W^{Q,e}} * \mathbf{e_t} \tag{5}$$

$$\mathbf{q_t^n} = \mathbf{W^{Q,n}} * \mathbf{n_t} \tag{6}$$

where $\mathbf{W^{Q,e}}$ and $\mathbf{W^{Q,n}}$ are learned.

The resulting vectors $\mathbf{o_t^e}$ and $\mathbf{o_t^n}$ are computed as:

$$\mathbf{o_t^e} = \mathbf{e_t} + \mathbf{W^{O,e}} * (\mathbf{a_t^e} * \mathbf{V^e}) + \mathbf{b^{O,e}} \tag{7}$$

$$\mathbf{o_t^n} = \mathbf{n_t} + \mathbf{W^{O,n}} * (\mathbf{a_t^n} * \mathbf{V^n}) + \mathbf{b^{O,n}} \tag{8}$$

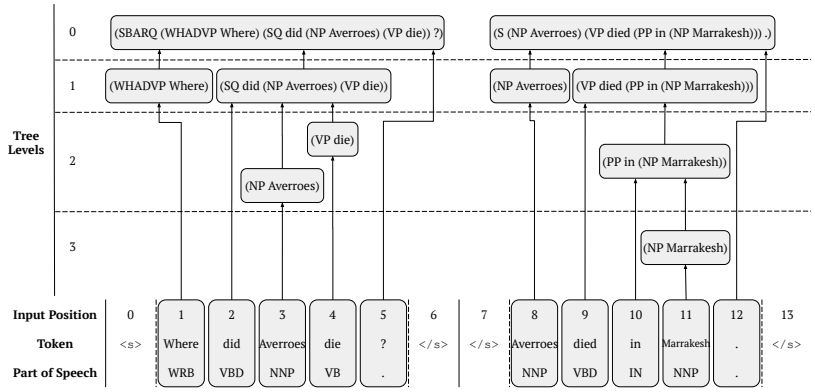

Figure 2: Input representation of an example question-sentence pair using RoBERTa.

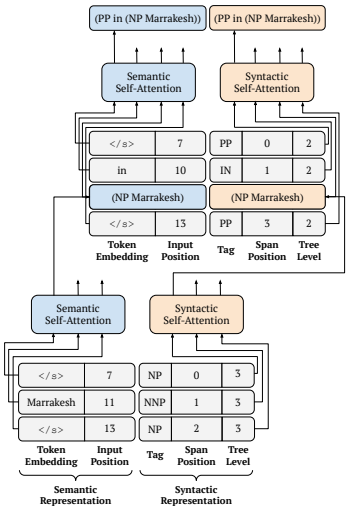

Figure 3: Detailed example of recursive tree aggregation.

where $\mathbf{V^e}$ and $\mathbf{V^n}$ are the value vectors for the input tokens, and $\mathbf{W^{O,e}}$, $\mathbf{W^{O,n}}$, $\mathbf{b^{O,e}}$, $\mathbf{b^{O,n}}$ are learned. Finally, we apply separate position-wise feed-forward layers to these output vectors.

Usually, self-attention includes residual dropout over the attention-weighted value vectors. We found in preliminary experiments that the performance on the dev set improved when we omitted dropout regularization. We omit dropout in both self-attention and position-wise feed-forward layer.

The recursiveness of the self-attention allows the model to re-use the same sets of parameters across each tree level, instead of training new ones as in previous work (Nguyen et al., 2019; Wang et al., 2019).

Constituent Span Embedding. Each input sentence is represented in a tree-structured fashion using its constituency parse tree. We use a pre-trained parser, whose parameters are fixed, to produce the trees before training time.

The constituent span is fed to the recursive self-attention as a matrix of token vectors. This matrix includes the embeddings of the words of the constituent span, preceded by a first, start-of-sentence embedding, and followed by an end-of-sentence embedding. The start-of-sentence token is the classification token if the span is part of the question, or a separator token if the span is part of the candidate sentence. Figure 3 shows how we compose a constituent span embedding for RoBERTa models.

The constituent span embedding is the output embedding of the first token. The first token embedding obtains through the recursive self-attention an attention-weighted sum of all of the span's token embeddings. This creates a span-specific embedding, conscious of the entire question-sentence pair input as a result of the language model layers, but focused on the tokens of a span as a result of the recursive self-attention.

In using only one layer of recursive self-attention, the first token embedding gets an attention-weighted sum of value vectors that contains token embeddings that did not go through a layer of self-attention, and syntactic embeddings that came directly out of the embedding layers.

Efficient Tree Aggregation. To obtain an aggregate sentence embedding, we proceed by embedding from the deepest level of the tree (the leaves) to the root, as shown in Figure 3. The computations are done on the same two sets of self-attention parameters.

To reduce training time, we compute the constituent span embeddings one level at a time. For instance, in Figure 2, we compute the NP, VP and PP groups at once when computing the span embeddings at tree level 2.

We efficiently compute all span embeddings only once, and keep all computed span embeddings, as they will be used in the next level.

The sentence embedding is obtained from the first token output of the computation at the root of the tree, as shown in Figure 1.

Prediction. Finally, we concatenate the aggregate embeddings for the question-sentence input pair. Given the question's aggregate semantic embedding $\mathbf{w_q^e}$ and aggregate syntactic embedding

$\mathbf{w_q^n}$, and the sentence's aggregate semantic embedding $\mathbf{w_s^e}$ and aggregate syntactic embedding $\mathbf{w_s^n}$, we obtain the prediction values as follows:

$$\mathbf{p(s|q)} = \text{softmax}\left(\mathbf{W} * \tanh\left[\mathbf{w_q^e}; \mathbf{w_q^n}; \mathbf{w_s^e}; \mathbf{w_s^n}\right] + \mathbf{b}\right) \tag{9}$$

where $\mathbf{W}$ and $\mathbf{b}$ are learned. We use binary cross-entropy as our loss function.

Our model can optionally include a residual connection, by adding the classification token embedding output of the language model to the beginning of the question-sentence pair vector. This residual connection does not contain syntactic information, and the classification token embedding is not projected in this case.

4 Experiments

4.1 Datasets

We evaluate our proposed *Tree Aggregation Transformer* on six English-language benchmark datasets for answer sentence selection. The first two – TrecQA and WikiQA – are widely used benchmarks in Question Answering (QA). The other four – YahooCQA and SemEval 2015, 2016 and 2017 – are all from the Community Question Answering (CQA) domain. We show the statistics of these six datasets in Table 1.

TrecQA (Wang et al., 2007) is collected from labeled sentences of the QA track of the Text REtrieval Conference (TREC). Over time, the dataset has evolved into two versions: the raw version includes all question-sentence pairs, whereas the clean version excludes questions with only non-relevant or only relevant candidate answers.

WikiQA (Yang et al., 2015) contains questions originally sampled from Bing query logs, and matched with candidate answer sentences from the first paragraph of relevant Wikipedia articles. Likewise, it also has a raw and a clean version. Following Lai et al. (2019); Tran et al. (2020), we evaluate our method on the clean versions of TrecQA and WikiQA.

YahooCQA (Tay et al., 2017) is a filtered and pre-processed subset of the large-scale *Yahoo! Answers Manner Questions* dataset (Surdeanu et al., 2008). The latter is based on the *Yahoo! Answers* online forum.

SemEval 2015 CQA (Nakov et al., 2015) is the challenge dataset of Subtask A of Task 3 of SemEval 2015. It is based on the Qatar Living online forum, and the goal is to predict the relevance scores of candidate answers given a question. The original subtask divides labels into three categories: definitely relevant, potentially useful, and irrelevant. Following previous work (Sha et al., 2018; Laskar et al., 2020), only definitely relevant candidate answers are marked as relevant in our binary classification setting.

Dataset		Number of Questions			Number of Answers		
		Train	Dev	Test	Train	Dev	Test
TrecQA Clean		1,229	65	68	53,417	1,117	1,442
WikiQA Clean		873	126	243	8,672	1,130	2,351
YahooCQA		50,112	6,289	6,283	253,440	31,680	31,680
SemEval	2015	2,600	300	329	16,541	1,645	1,976
	2016	4,879	244	327	36,198	2,440	3,270
	2017	4,879	244	293	36,198	2,440	2,930

Table 1: Statistics of the six benchmark datasets.

SemEval 2016 CQA (Nakov et al., 2016) corresponds as well to Subtask A of Task 3 of SemEval 2016, about question-comment similarity. It is a new dataset also based on the Qatar Living online forum. The training set includes the training, development and testing sets of the SemEval 2015 CQA, and two new training sets. The authors of the dataset have described the first one as highly reliable, and the second one as noisier.

SemEval 2017 CQA (Nakov et al., 2017) is the latest version of the community question answering task. The training and development sets are the same as the 2016 version, but the testing set is different.

In Figure 2, we show an example of question-sentence pairs for a QA dataset and a CQA dataset. The aim is to illustrate the difference in style and length between formal (QA) and informal (CQA) text.

4.2 Setup

The standard evaluation metrics in answer sentence selection are Mean Average Precision (MAP) and Mean Reciprocical Rank (MRR). Both metrics are widely used in Information Retrieval (IR) and are averaged per query – in this case per question. Our model produces relevance scores going from 0 (irrelevant) to 1 (relevant) for each candidate answer, and therefore produces a list of candidate answers that can be ranked by relevance. Whereas MRR scores how early a first relevant answer appears in that candidate list, MAP scores the order in which all candidate answers are listed for each question.

To produce parse trees, we use the NLTK part-of-speech tagger (Loper and Bird, 2002) trained on the part-of-speech tagset of the English Penn Tree-

Dataset	Question	Answer
WikiQA	how are glacier caves formed ?	A glacier cave is a cave formed within the ice of a glacier .
SemEval 2016-2017	Why people are crossing red signals on Doha Roads? I think signals are changing quickly than on Dubai roads and its hard for the motorists to control their vehicles? Moreover; motorists are bit panic fearing the penalties as per the new traffic law.	also i traffic lights here does not have standard options. some have blinking green light; some chage to yellow right away then red. several times alredy i found my self driving in the middle of the crossing in red light luckily at the moment no fines. hehehe :) pykester

Table 2: Samples of question-sentence pairs from the training sets of WikiQA and SemEval 2016-2017 (both years share the same training dataset). Here, the sentence contains an answer to the question.

Representation	TrecQA		WikiQA		YahooCQA		SemEval CQA			
							2015		2016-2017	
	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR
Semantic Only	0.932	0.958	0.892	0.901	0.929	0.929	**0.947**	0.959	0.911	0.950
Semantic + Syntactic	**0.946**	**0.961**	**0.898**	**0.912**	**0.933**	**0.933**	0.945	**0.962**	**0.914**	**0.957**

Table 3: Ablation study on syntactic representations: Results for our Tree Aggregation Transformer with and without learned syntactic embeddings for all of our benchmark dev sets, on RoBERTa Large.

bank (PTB) (Marcus et al., 1994), and the English-language parser of Mrini et al. (2020), which is the state of the art on the parse trees of the PTB.

4.3 Training Parameters

We use 1 layer of recursive self-attention for all datasets. We use the residual connection described in §3 for TrecQA only. For all our models, we use either BERT large or RoBERTa large, so as to match our baselines. Our recursive self-attention layers have: 16 attention heads, a feed-forward dimension of 4096, and a hidden dimension of 2048. We use half of the dimensions to encode semantic information, and the rest to encode syntactic information.

4.4 Ablation Study on Syntactic Embeddings

We perform an ablation study by removing the syntactic embedding part of the input representation. In this experiment, we are quantifying the added value of the learned syntactic embeddings for span position, part-of-speech tags and syntactic categories, and tree levels.

Our results on the dev sets are in Table 3. SemEval 2016 and 2017 results are the same since both have the same dev set. Across all AS2 datasets, we notice that there is an advantage to learning syntactic embeddings, as the sum of MRR and MAP scores are higher for the variant that includes learned syntactic embeddings. The advantage is clearer for QA datasets, suggesting that formal language tends to benefit more from learned syntactic information. We use syntactic embeddings in our next experiments.

4.5 Baselines

We conside five strong baselines, described in §2:

Model	TrecQA		WikiQA	
	MAP	MRR	MAP	MRR
Chen et al. (2017b)	0.781	0.851	0.721	0.731
Bian et al. (2017)	0.821	0.899	0.754	0.764
Tay et al. (2018)	0.784	0.865	0.712	0.727
Chen et al. (2018a)	0.823	0.889	0.736	0.745
Chen et al. (2018b)	0.841	0.917	0.730	0.743
Sha et al. (2018)	-	-	0.746	0.758
Madabushi et al. (2018)	0.865	0.904	-	-
Tymoshenko and Moschitti (2018)	-	-	0.762	0.776
Kamath et al. (2019)	-	-	0.700	0.716
Models using BERT Large				
GSAMN (Lai et al., 2019)*	0.914	0.957	**0.857**	**0.872**
TandA (Garg et al., 2019)*	0.912	**0.967**	-	-
Reg. Self-Attention (Laskar et al., 2020)	0.789	0.887	0.714	0.731
Direct Fine-tuning (Laskar et al., 2020)	0.905	**0.967**	0.843	0.857
Our Tree Aggregation Transformer	**0.917**	0.961	0.851	0.868
Models using RoBERTa Large				
TandA (Garg et al., 2019)*	0.943	0.974	-	-
Direct Fine-tuning (Laskar et al., 2020)	0.936	0.978	0.900	0.915
Our Tree Aggregation Transformer	**0.950**	**0.985**	**0.906**	**0.920**
Models using RoBERTa Large and Evidence Memory				
Evidence Memory (Tran et al., 2020)*	0.961	0.993	0.936	0.952
Our Tree Aggregation Transformer	**0.970**	**0.995**	**0.941**	**0.958**

Table 4: Our results in comparison with recent work on the TrecQA and WikiQA benchmark datasets. * indicates use of transfer learning on large-scale datasets.

(1) GSAMN (Lai et al., 2019): Gated Self-Attention Memory Networks.
(2) TandA (Garg et al., 2019): the two-step Transfer and Adapt method.
(3) Regular Self-Attention (Laskar et al., 2020): a self-attention layer fine-tuned over frozen BERT Large embeddings.
(4) Direct Fine-tuning (Laskar et al., 2020): directly fine-tuning on a pre-trained language model.
(5) Evidence Memory (Tran et al., 2020): the neural Turing machine as memory controller.

Baselines 1, 2, and 5 are available only on TrecQA and/or WikiQA, whereas baselines 3 and 4 use the exact same datasets as we do.

4.6 Results and Discussion

The results of our experiments with the QA datasets are in Table 4, and the results of our experiments

with CQA datasets are in Table 5.

4.6.1 State of the Art in QA datasets

Our results in Table 4 establish a new state of the art in TrecQA and WikiQA, two widely used benchmark datasets in answer sentence selection.

In TrecQA, our average of MAP and MRR scores matches the one for *TandA* (Garg et al., 2019) in BERT, without any transfer learning on a large dataset. This shows that our model is able to leverage the tree structure to increase performance on relatively small datasets.

For the RoBERTa results in WikiQA, the added value between the direct fine-tuning and our recursive self-attention confirms that our model is beneficial to formally written text, such as the one found in Wikipedia.

The increase in performance compared to the Evidence Memory models (Tran et al., 2020) when we add our tree representation shows that our tree aggregation method brings about a consistent and robust added value for the QA datasets.

4.6.2 Limitations in CQA datasets

As shown in Table 5, our *Tree Aggregation Transformer* is able to establish a new state of the art in SemEval 2015, and our BERT-based version exceeds other BERT-based baselines. However, our method scores below the state of the art in YahooCQA and SemEval 2016, and only manages to match the MRR – but not the MAP – of the state of the art in SemEval 2017.

Therefore, there is a contrast in the performance of our recursive tree-structured self-attention between the QA and the CQA datasets. The difference lies in the style of the datasets, as questions and sentences can be much longer in QA datasets than in CQA datasets. On average, a training set pair in QA has 32 words for WikiQA, and 39 words in TrecQA, whereas a training set pair in CQA has 78 words for SemEval 2015, 85 words for SemEval 2016-2017, and 40 words for YahooCQA. As shown in the example, CQA pairs may also have spelling mistakes or lack coherent structure. Thus, the informal writing style and larger text length of CQA datasets may be decreasing the ability of our model to leverage tree structures. Accordingly, we see that our model achieves very competitive scores for YahooCQA, and that it has a text length that is very close to the QA datasets. The SemEval 2015 exception could be explained by the fact that the 2015 training dataset is less noisy than the 2016-2017 training dataset, as pointed out by the authors of the SemEval CQA datasets.

4.7 Do Tree Structures Improve Performance?

We investigate how tree structures are leveraged in the Answer Sentence Selection task across the different datasets. We evaluate our tree-structured representations and compare them with the corresponding sequential representations, using three probing tasks from Conneau et al. (2018).

4.7.1 Probing Tasks

The three probing tasks are as follows:

(1) Top Constituent Prediction. This task looks to predict the top constituent sequence of the question-sentence pair: the sequence of syntactic categories immediately below the `S` (Sentence) syntactic category. Following Conneau et al. (2018), we define this task as a 20-way classification problem, where the first 19 classes are the 19 most popular top constituent sequences, and the last category is for all the remaining top constituent sequences.

(2) Tree Depth Prediction. The tree depth is the number of hops from the root node of the syntactic tree to the lowest-level leaf nodes.

(3) Input Length Regression. This tasks investigates whether the embedding is aware of how many words it contains. The length of the question-sentence pair input is defined as the number of its tokens – full words and punctuation symbols.

The first two tasks are syntactic, and investigate whether our tree-structured representations absorbed the syntactic category information that we fed it – respectively syntactic categories and tree levels – and whether that information was already present in the sequential representations.

4.7.2 Probing Experiment Setup

In our probing experiments, we consider all six datasets used both in our work and in Laskar et al. (2020). We consider the sequential representation of a question-answer pair to be the classification token embedding used for prediction in the RoBERTa-based models of Laskar et al. (2020). We take our own RoBERTa-based tree-structured models (without evidence memory), where we consider the tree-structured representation to be the classification token embedding fed to the prediction layer. The tree-structured and sequential representations have the same number of dimensions.

Model	YahooCQA		SemEval CQA					
			2015		2016		2017	
	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR
Nakov et al. (2017)	-	-	-	-	-	-	0.884	0.928
Tay et al. (2018)	0.801	0.801	-	-	-	-	-	-
Sha et al. (2018)	-	-	-	-	0.801	0.872	-	-
Models using BERT Large								
Regular Self-Attention (Laskar et al., 2020)	0.778	0.778	0.883	0.923	0.765	0.831	0.867	0.922
Direct Fine-tuning (Laskar et al., 2020)	**0.951**	**0.951**	0.935	0.961	**0.866**	**0.927**	**0.921**	**0.963**
Our Tree Aggregation Transformer	0.946	0.946	**0.946**	**0.972**	0.844	0.900	0.902	0.955
Models using RoBERTa Large								
Direct Fine-tuning (Laskar et al., 2020)	**0.955**	**0.955**	0.947	0.970	**0.888**	**0.938**	**0.943**	**0.974**
Our Tree Aggregation Transformer	0.949	0.949	**0.961**	**0.981**	0.863	0.918	0.926	**0.974**

Table 5: Our results in comparison with recent work on the YahooCQA and SemEvalCQA benchmark datasets.

Probing Task	Representation	TrecQA	WikiQA	YahooCQA	SemEval			Spearman's ρ	
					2015	2016	2017	MAP	MRR
Top Constituent Prediction (F1 score)	**Tree-Structured**	0.1573	0.1949	0.0354	0.2058	0.0674	0.1151	0.8214	0.9550
	Sequential	0.0475	0.0463	0.0364	0.0434	0.0505	0.0483		
Tree Depth Prediction (F1 score)	**Tree-Structured**	0.1568	0.1638	0.0354	0.1682	0.0621	0.1340	0.8214	0.9550
	Sequential	0.0481	0.0476	0.0354	0.0451	0.0523	0.0481		
Input Length Regression (MSE)	**Tree-Structured**	0.0266	0.0273	4.51e-06	0.0652	0.0989	0.0416	-0.0360	0.1429
	Sequential	0.0822	0.1200	4.14e-06	0.2915	0.3338	0.1484		

Table 6: Results for three probing tasks comparing sequential (Laskar et al., 2020) and tree-structured (ours) representations. In the last two columns, we show the Spearman correlation of the probing task and the AS2 performance differences between the tree-structured and sequential representations.

The probing model architecture is a simple MLP with a layer of the same size as the input embeddings, a ReLU activation, and a prediction layer. We train 36 probing models for each of the 36 combinations of a probing task, a dataset and a representation type. The input embeddings are frozen, so that the training does not change the weights of the pre-trained AS2 models. All experiments are trained for the same number of epochs, and use the same train/dev/splits as AS2 experiments.

4.7.3 Probing Results and Discussion

Our probing experiment results are shown in Table 6. We compute the Spearman correlations of the added values of the tree-structured representations compared to the sequential representations in each probing task with the same added value in the AS2 task. We compute the added value of the tree representation in a given task by subtracting the performance of the sequential representations (Laskar et al., 2020) from the performance of the tree-structured representations (ours).

For the syntactic probing tasks (the first two), the tree-structured representation gets an F1 score about 3 to 4 times higher than the one obtained by the sequential representation in 4 datasets: TrecQA, WikiQA, and SemEval 2015 and 2017. These 4 datasets correspond to the ones in which our tree-structured AS2 models set a new state of the art or matched the performance of the fine-tuning baseline of Laskar et al. (2020). In the other datasets, the tree-structured representation's F1 score is just slightly higher than the sequential representation's F1 score, if not about the same. This shows that when the tree-structured representations successfully absorb the syntactic information we fed it, there is a consistent increase in performance in the answer sentence selection task. The high correlation values for both MAP and MRR confirm that successfully absorbing syntactic information is associated with higher performance in AS2. The weakness of tree-structured representations in certain datasets may be due to the lack of generalization of syntactic parsers trained on the Penn Treebank.

In the input length probing experiment, we observe that the mean-squared error (MSE) of the tree-structured representations is consistently and significantly lower than the one of the sequential representations, except for YahooCQA. This shows that the recursion of our tree-structured AS2 model makes representations aware of the length of their question-sentence pair, but the correlation values show that this information does not necessarily help in the AS2 task.

5 Conclusions

We introduce the *Tree Aggregation Transformer*: a novel, recursive and tree-structured self-attention model for AS2. Our method embeds sentences by aggregating word representations following the corresponding parse tree. We show that our model leverages tree structure and, through an ablation study, that its learned syntactic embeddings increase performance. Our method establishes a new state of the art in the TrecQA and WikiQA benchmark datasets with only one additional self-attention layer. Our tree-structured self-attention exceeds or matches the state of the art in 2 out of 4 CQA datasets, where text is informal and longer. To investigate this mixed performance, we devise 3 probing tasks to examine what our tree-structured representations learn compared to their sequential counterparts. We find that there is a strong correlation between a tree-structured model's ability to absorb syntactic information and its ability to increase performance in the AS2 task compared to baselines. Our findings suggest that there is more work to be done for tree-structured representations to adapt to noisy user-generated text.

Acknowledgements

We gratefully acknowledge the award from NIH/NIA grant R56AG067393. This work is part of the VOLI project (Mrini et al., 2021; Johnson et al., 2020). We thank the anonymous reviewers for their feedback.

References

Mahtab Ahmed, Muhammad Rifayat Samee, and Robert E Mercer. 2019. You only need attention to traverse trees. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 316–322.

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 1987–1990, New York, NY, USA. Association for Computing Machinery.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017a. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, and Liang He. 2018a. Ca-rnn: using context-aligned recurrent neural networks for modeling sentence similarity. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, and Liang He. 2018b. Can: Enhancing sentence similarity modeling with collaborative and adversarial network. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 815–824.

Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, Liang He, and Weijie An. 2017b. Enhancing recurrent neural networks with positional attention for question answering. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 993–996.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846, Vancouver, Canada. Association for Computational Linguistics.

Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2019. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. *arXiv preprint arXiv:1911.04118*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Jacob Harer, Chris Reale, and Peter Chin. 2019. Tree-transformer: A transformer-based method for correction of tree-structured data. *arXiv preprint arXiv:1908.00449*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Janet Johnson, Khalil Mrini, Allison Moore, Emilia Farkas, Ndapa Nkashole, Michael Hogarth, and Nadir Weibel. 2020. Voice-based conversational

agents for older adults. In *Proceedings of the CHI 2020 Workshop on Conversational Agents for Health and Wellbeing, Honolulu, Hawaii*.

Sanjay Kamath, Brigitte Grau, and Yue Ma. 2019. Predicting and integrating expected answer types into a simple recurrent neural network model for answer sentence selection. In *20th International Conference on Computational Linguistics and Intelligent Text Processing*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Easy-first dependency parsing with hierarchical tree lstms. *Transactions of the Association for Computational Linguistics*, 4:445–461.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Tuan Lai, Quan Hung Tran, Trung Bui, and Daisuke Kihara. 2019. A gated self-attention memory network for answer selection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5955–5961.

Md Tahmid Rahman Laskar, Xiangji Huang, and Enamul Hoque. 2020. Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5505–5514.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, page 63–70, USA. Association for Computational Linguistics.

Harish Tayyar Madabushi, Mark Lee, and John Barnden. 2018. Integrating question classification and deep learning for improved answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3283–3294.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, page 114–119, USA. Association for Computational Linguistics.

Khalil Mrini, Chen Chen, Ndapa Nakashole, Nadir Weibel, and Emilia Farcas. 2021. Medical question understanding and answering for older adults. *The 3rd Southern California (SoCal) NLP Symposium*.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 731–742, Online. Association for Computational Linguistics.

Khalil Mrini, Claudiu Musat, Michael Baeriswyl, and Martin Jaggi. 2019. Structure tree-lstm: Structure-aware attentional document encoders. *arXiv preprint arXiv:1902.09713*.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. SemEval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 269–281, Denver, Colorado. Association for Computational Linguistics.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545.

Xuan-Phi Nguyen, Shafiq Joty, Steven Hoi, and Richard Socher. 2019. Tree-structured attention with hierarchical accumulation. In *International Conference on Learning Representations*.

Lei Sha, Xiaodong Zhang, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. A multi-view fusion neural network for answer selection. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1201–1211.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y Ng, and Christopher D Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Zeyu Sun, Qihao Zhu, Yingfei Xiong, Yican Sun, Lili Mou, and Lu Zhang. 2020. Treegen: A tree-based transformer architecture for code generation. In *AAAI*, pages 8984–8991.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online qa collections. In *Proceedings of ACL-08: HLT*, pages 719–727.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 695–704.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 583–591.

Zhiyang Teng and Yue Zhang. 2017. Head-lexicalized bidirectional tree lstms. *Transactions of the Association for Computational Linguistics*, 5:163–177.

Quan Hung Tran, Nhan Dam, Tuan Lai, Franck Dernoncourt, Trung Le, Nham Le, and Dinh Phung. 2020. Explain by evidence: An explainable memory-based neural network for question answering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5205–5210.

Quan Hung Tran, Gholamreza Haffari, and Ingrid Zukerman. 2017. A generative attentional neural network model for dialogue act classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 524–529, Vancouver, Canada. Association for Computational Linguistics.

Kateryna Tymoshenko and Alessandro Moschitti. 2018. Cross-pair text representations for answer sentence selection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2162–2173.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1060–1070.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.