Syntactically Informed Natural Language Inference

Deric Pang **Joshua Bean**
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA, USA
{dericp, jbean96}@cs.washington.edu

Abstract

We demonstrate the importance of syntactic information in semantic models by extending the decomposable attention model for natural language inference. By pipelining the hidden states of a neural syntax parser into the decomposable attention model, we achieve greater performance on the SciTail dataset than the ESIM model and DA + ELMo.

1 Introduction

Natural language inference (NLI) is the task of characterizing entailment and contradiction relationships between texts. In general, most NLI tasks are formulated as characterizing the relationship between a pair of sequences—a premise and a hypothesis. An NLI model should predict whether the hypothesis is entailed by the premise, contradicts the premise, or is neutral to the premise.

We extend the decomposable attention (DA) model from Parikh et al. (2016) by incorporating syntactic features. The DA model obtained state-of-the-art results on the SNLI (Bowman et al., 2015) dataset at its time of publication while using drastically fewer parameters than previous NLI models. We primarily evaluate on the domain-specific SciTail dataset (Khot et al., 2018) for both its smaller size and lack of annotation artifacts when compared to SNLI (Gururangan et al., 2018).

Our model, `syntail`, achieves 4.8% better test accuracy than the vanilla DA model and 1.9% better test accuracy than DA with ELMo (Peters et al., 2018) all while maintaining the core DA architecture.

2 Decomposable Attention Model

The DA model is a simple and easily parallelizable approach to NLI. We summarize it here so we can build upon it later. The model decomposes into attend, compare, and aggregate steps.

Let $\boldsymbol{p} = \langle p_1, \ldots, p_{\ell_p} \rangle$ and $\boldsymbol{h} = \langle h_1, \ldots, h_{\ell_h} \rangle$ where each $p_i, h_j \in \mathbb{R}^d$ is a d-dimensional word embedding vector.

Attend. Compute unnormalized attention weights e_{ij} with a feed-forward neural network F:

$$e_{ij} := F(p_i)^\mathsf{T} F(h_j). \tag{1}$$

Compute the aligned subphrases $\boldsymbol{H}i$ and $\boldsymbol{P}_j$:

$$\begin{aligned} \boldsymbol{H}_i &:= \sum_{j=1}^{\ell_h} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_h} \exp(e_{ik})} h_j \,, \\ \boldsymbol{P}_j &:= \sum_{i=1}^{\ell_p} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_h} \exp(e_{ik})} p_i \end{aligned} \tag{2}$$

where $\boldsymbol{H}_i$ is the weighted sum over $\boldsymbol{h}$ that aligns to p_i and vice versa for $\boldsymbol{P}_j$.

Compare. Compare each $(p_i, \boldsymbol{H}_i)$ and $(h_j, \boldsymbol{P}_j)$ pairs with another feed-forward neural network G:

$$\begin{aligned} \boldsymbol{v}_i^p &:= G([p_i, \boldsymbol{H}_i]) \quad \forall i \in [1, \ldots, \ell_p] \,, \\ \boldsymbol{v}_j^h &:= G([h_j, \boldsymbol{P}_j]) \quad \forall j \in [1, \ldots, \ell_h] \,. \end{aligned} \tag{3}$$

Aggregate. Aggregate each set of comparison vectors:

$$\boldsymbol{v}^p = \sum_{i=1}^{\ell_p} \boldsymbol{v}_i^p \qquad , \qquad \boldsymbol{v}^h = \sum_{j=1}^{\ell_h} \boldsymbol{v}_j^h \,. \tag{4}$$

and make a prediction with a final feed-forward layer H:

$$\hat{\boldsymbol{y}} = H([\boldsymbol{v}^p, \boldsymbol{v}^h]). \tag{5}$$

3 Our Model

The motivation for incorporating syntactic information in an NLI model can be demonstrated with a simple example. Consider the premise "Adrian is running and Alex is swimming." If we present the DA model trained on SNLI with the hypothesis "Adrian is swimming," it will confidently predict that this blatantly incorrect hypothesis is entailed by the premise. Observing any syntactic parse of the premise will make it obvious that Adrian is in fact running.

We experiment with features produced by two syntax parsers—the minimal span-based neural constituency parser (Stern et al., 2017) and the deep biaffine attention model for dependency parsing (Dozat and Manning, 2016). We also experiment with different methods of incorporating the syntactic features extracted from these parsers.

3.1 Extracting Syntactic Information

The minimal span-based neural constituency parser and the deep biaffine attention model for dependency parsing both encode the input sequence with an LSTM. Using this LSTM, we obtain syntactic features $\boldsymbol{s}^p = \langle s^p_1, \ldots, s^p_{\ell_p} \rangle$ and $\boldsymbol{s}^h = \langle s^h_1, \ldots, s^h_{\ell_h} \rangle$ from the premise and hypothesis, respectively. An s_i is either the final hidden state $\boldsymbol{r}_i$ of the LSTM at index i or a projected representation of $\boldsymbol{r}_i$.

3.2 Syntail Architectures

We will now describe the different model architectures we experimented with. In general, as the version number increases, the complexity of the model increases.

v1: Naive late fusion of syntactic features. The simplest way to incorporate $\boldsymbol{s}$ is to concatenate its final representation to $\boldsymbol{v}^p$ and $\boldsymbol{v}^h$ as shown in figure 1.

The prediction step in equation 5 then becomes:

$$\hat{\boldsymbol{y}} = H([\boldsymbol{v}^p, s^p_{\ell_p}, \boldsymbol{v}^h, s^h_{\ell_h}]) \tag{6}$$

where we concatenate the final hidden representation of the premise and hypothesis when passed through the neural syntax parser with the aggregrated comparision vectors. Everything else is identical to the DA model.

v2: Using syntactic features to attend. We wanted to experiment with incorporating syntactic features earlier in the model. Instead of learning weights on the syntactic features, it is possible to concatenate them to the projected input representations as show in figure 2.

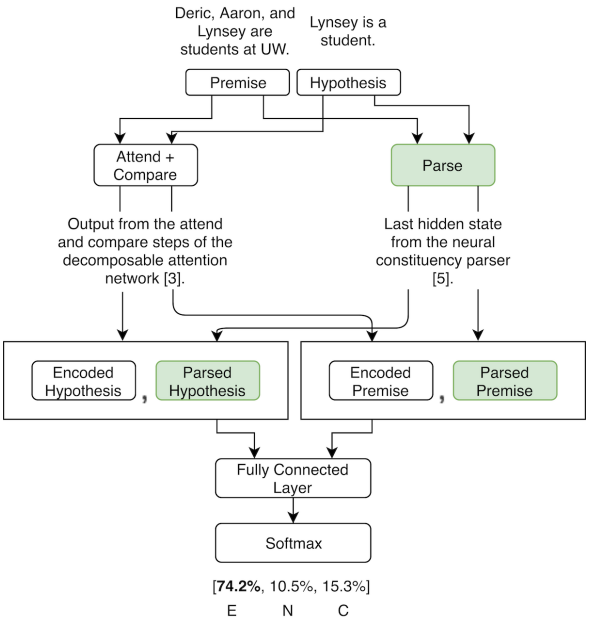

Figure 1: Syntail v1 architecture.

Computing the unnormalized attention weights in equation 1 then becomes:

$$e_{ij} := [F(p_i), s^p_i]^\mathsf{T} [F(h_j), s^h_i] \tag{7}$$

where instead of calculating the alignment using only the projected hypothesis and premise vectors, we now also use the final hidden states of the premise and hypothesis when passed through the neural syntax parser. This seems rather naive since this model will not learn any weights on the syntactic features, but in practice this model performs very well. Everything else is identical to the DA model.

v3: Using syntactic features to project the premise and hypothesis. We decided to try incorporating the syntactic features even earlier in the model. Instead of concatenating the final hidden states of the parsed premise and hypothesis with the projected inputs, we concatenate the final hidden states of the parsed premise and hypothesis with the inputs immediately before projecting them. This architecture is shown in figure 3.

Computing the unnormalized attention weights in equation 1 then becomes:

$$e_{ij} := F([p_i, s^p_i])^\mathsf{T} F([h_j, s^h_j]). \tag{8}$$

Everything else is identical to the DA model.

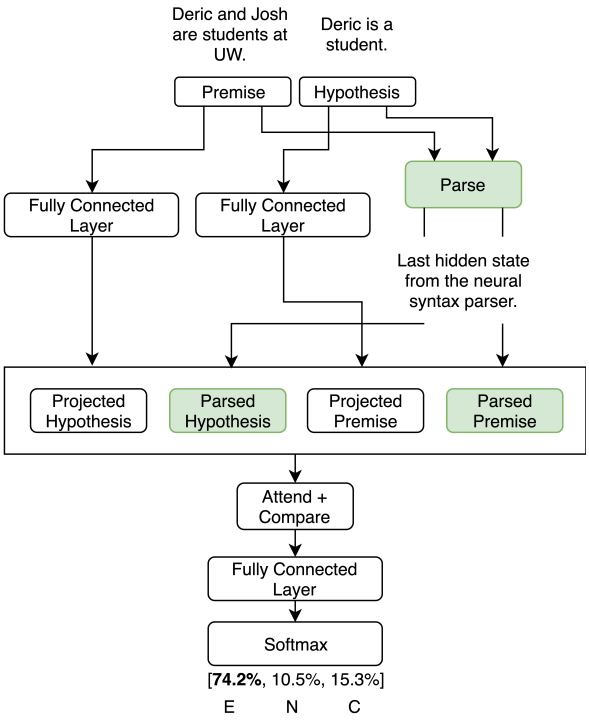

Figure 2: Syntail v2 architecture.

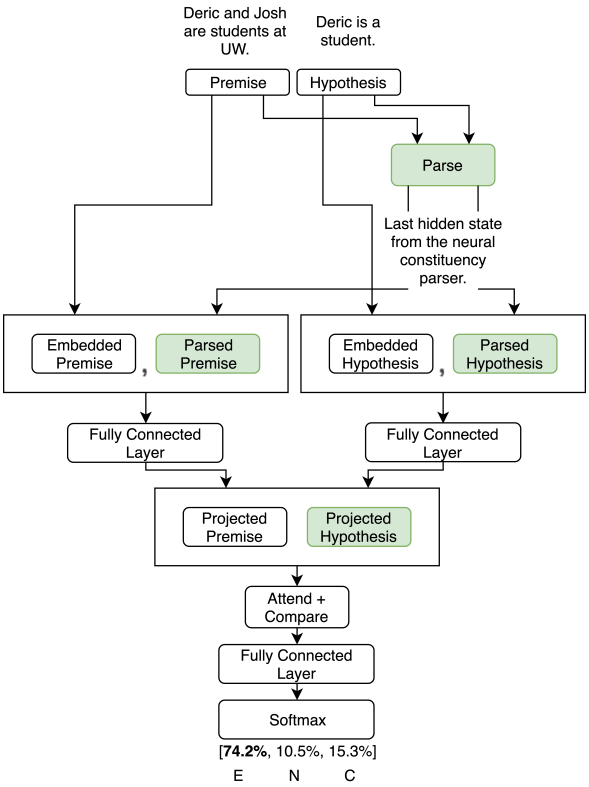

Figure 3: Syntail v3 architecture.

v4: Projecting the syntactic features before computing attention. After noticing a drop in performance from v2 to v3, we hypothesized that it was due to not allowing our model to transform the syntactic features enough before calculating attention. We decided to learn weights to project the final hidden states of the parsed premise and hypothesis by passing them through an additional fully-connected layer before concatenating them to the inputs. This architecture is shown in figure 4

Computing the unnormalized attention weights in equation 1 then becomes:

$$e_{ij} := F([p_i, J(s_i^p)])^\mathsf{T} F([h_j, J(s_j^h)]) \quad (9)$$

where J is a feed-forward neural network. Everything else is identical to the DA model.

3.3 Implementation Details

3.3.1 Word Embeddings

Due to resource constraints, we limited our models to only use 300-dimensional 6B GloVe word embeddings (**?**). Additionally, contextual embeddings are known to capture syntax in some capacity, so limiting our models to only use GloVe vectors made it easier to experiment with the syntactic features in isolation.

3.3.2 Neural Syntax Parser

We initially used a constituency parser in syntail. However, after some experimentation, we found that we were achieving better results with a dependency parser. For this reason, we only report results using the deep biaffine attention model for dependency parsing. Dependency parsers focus on encoding the the relationships between words as opposed to determining constituents. Intuitively, if we consider our example sentence, "Adrian is running and Alex is swimming," the dependency arcs will be more informative of who is doing what.

3.4 Hyper-parameters

We use 200-dimensional projections of the word embeddings. Our feed-forward neural networks are 2 layered and have hidden dimensions of 200. We train with both Adam (**?**) and Adagrad (**?**) and report the better result.

4 Experiments

We compare our model's performance to the decomposable attention model (Parikh et al., 2016) and the ESIM model (**?**). Early on in our experiments, we noticed that features from a dependency parser seemed to perform better than those from a

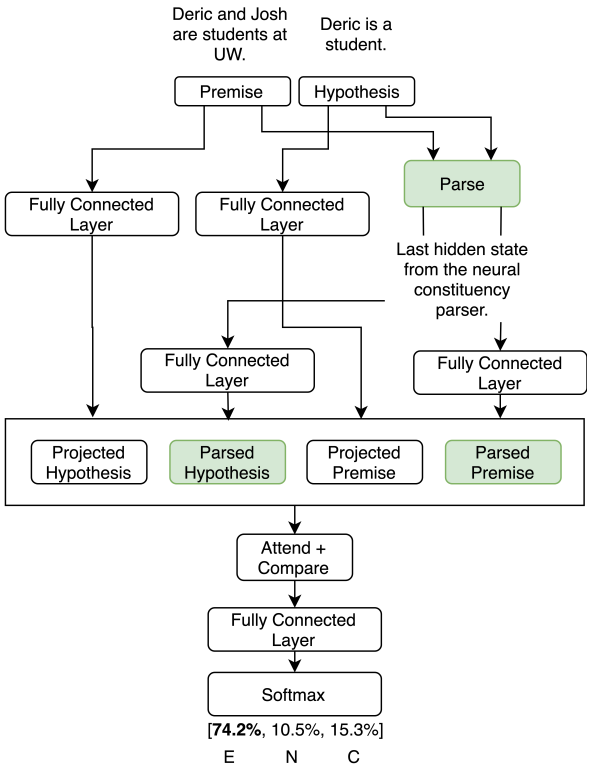

Figure 4: Syntail v4 architecture.

constituency parser. Therefore, all of our models use the deep biaffine attention model for dependency parsing (Dozat and Manning, 2016).

Datasets. For practical reasons, we only experiment on the SciTail dataset. SciTail (27k pairs) is significantly smaller than SNLI (570k pairs) (Bowman et al., 2015) or MultiNLI (433k pairs) (Williams et al., 2017) which makes it feasible to try many different architectures given limited resources.

Additionally, Gururangan et al. (2018) showed that SNLI and MultiNLI have significant annotation artifacts whereas SciTail is the first entailment set to be completely derived from sentences that exist "in the wild."

Baselines. We list baseline performance reported by Khot et al. (2018) of a majority class predictor, n-gram model, and ESIM (**?**) in table 1. We reimplemented and retrained the DA model and obtained similar test accuracies as reported by **?**. Adding ELMo to the DA model improved test accuracy performance by approximately 3%.

4.1 Results

Our experimental results can be found in table 1. All of our syntail models achieve better test accuracy than the baseline models. This demonstrates that the syntax parser is providing features that are both useful for determining entailment relationships and difficult for the baseline models to learn on their own.

Model	Embeddings	Dev. Acc.	Test Acc.
Majority Class	n/a	63.3	60.3
N-Gram	n/a	65.0	70.6
ESIM	GloVe 840B 300d	70.5	70.6
DA	GloVe 6B 300d	70.4	72.6
DA	ELMo	79.1	75.5
Syntail v1	GloVe 6B 300d	78.4	**77.4**
Syntail v2	GloVe 6B 300d	**82.3**	**77.4**
Syntail v3	GloVe 6B 300d	79.0	76.1
Syntail v4	GloVe 6B 300d	81.3	77.0

Table 1: Dev and test accuracies on SciTail.

We were surprised that the simpler v1 and v2 models outperformed the more complicated v3 and v4 models. This is likely due to the syntactic information being fused later in v1 and v2. The later the features are fused, the closer they are to the loss function and the more the model can learn to use them.

It is also interesting that adding and additional transformation on the syntactic features in v4 decreases performance. Again, this is likely due to the early fusion problem. The additional power of the model is overwhelmed by the detrimental effects of the vanishing gradient problem. It is also possible that the randomly initialized weights of the feed-forward neural network struggle to learn how to summarize the syntactic features.

One downside of using a pretrained syntax parser is that the data SciTail is created from (science exam questions) is in a very different domain than the data used to create the Penn Treebank (news articles) (**?**). This likely significantly reduces the accuracy of the syntax parser on SciTail. Retraining the syntax parser on in-domain data should produce better syntactic features and further improve the performance of our model.

4.2 Analysis of Syntactic Features

We discuss a specific in-domain input example shown in table 2 to better understand the effects of syntactic features. The predictions on this input made by the DA and syntail v1 models are shown in table 3.

Syntail v1 correctly determines the relationship of this pair of sentences while the DA model does not. To investigate why, we visualize the attention

Premise	Hypothesis
The continental crust is less dense than the mantle but the oceanic crust is not.	The oceanic crust is more dense than the mantle.

Table 2: A pair of sentences where syntail performs particularly well. The hypothesis is entailed by the premise.

Model	Entails (correct)
DA	29.5%
Syntail v1	**67.9%**

Table 3: Entailment probabilities generated by the DA and syntail v1 models on the input from table 2.

weights in figures 5 and 6. We can see that the attention produced by the DA model is nonsensical and focuses solely on "is." It seems that by incorporating syntactic information, syntail v1 is forced to learn a more reliable attention function.

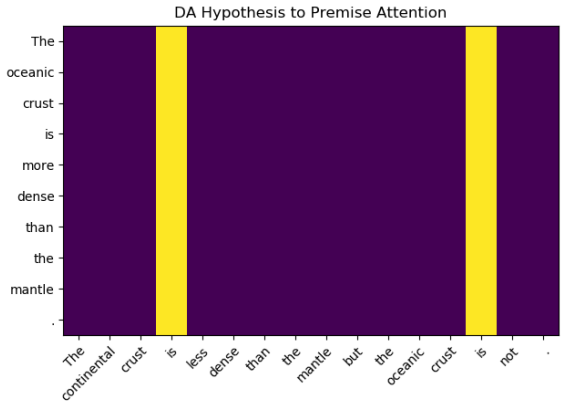

Figure 5: Hypothesis to premise attention weights of the DA model when processing the input from table 2.

5 Related Work

Previous works have experimented with incorporating syntactic features for semantic tasks. **?** use tree structures generated by a constituency parser to improve their ESIM model. They encode the phrase-structure trees with a Tree-LSTM and use its hidden states. Khot et al. (2018) incorporate syntactic features by proposing a model architecture which can use any graph-based syntactic/semantic structure.

Other works have experimented with incorporating syntactic learning objectives. **?** achieved state-of-the-art performance for a model using predicted predicates and standard word embeddings on CoNLL-2005 SRL by performing multi-task training on additional syntactic tasks. **?** use

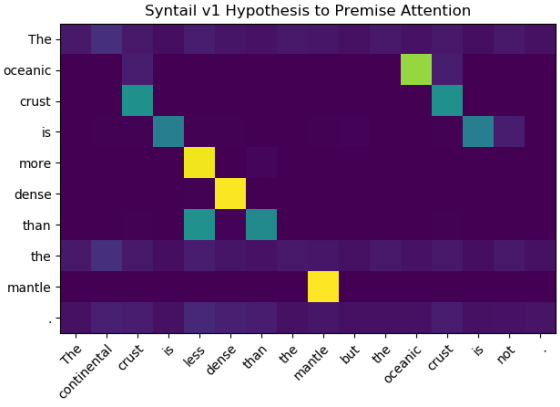

Figure 6: Hypothesis to premise attention weights of syntail v1 when processing the input from table 2.

hard-coded constituency and dependency parse features and a syntactic scaffold to extend the softmax margin segmental RNN (segRNN) model.

6 Conclusion

By making simple extensions to the DA model, our syntail models outperform ESIM and DA + ELMo model on SciTail. Our results demonstrate the importance of syntactic information in semantic models and motivate future research into syntactically informed models.