Sentence Level Representation Learning

Fatima Al-Raisi
Carnegie Mellon University
Pittsburgh PA
fraisi@andrew.cmu.edu

Griffin Adams
Carnegie Mellon University
Pittsburgh PA
gta@andrew.cmu.edu

Sarguna Janani Padmanabhan
Carnegie Mellon University
Pittsburgh PA
sjpadman@andrew.cmu.edu

1. Introduction

Linguistic theorists generally agree that language is constructed recursively and according to learnt syntactic structure [27, 8, 17]. Yet, most state-of-the-art neural methods for sentence modeling do not make use of the generative topological structure of sentences. Recurrent neural networks (RNNs), the de facto standard for sentence modeling, treat language as a flat sequence. We want to explore the new area of compositional learning, where words are grouped together and recursively combined to form a singular semantic representation. The predominant approaches to this hierarchical modeling are Convolutional Neural Networks (CNNs) and Recursive Neural Networks (also known as tree RNNs). Tree RNNs define compositional units over a parse tree, whereas CNNs apply alternating local filter functions and pooling operations to progressively combine wider and wider contexts. We believe that if such approaches are able to properly incorporate long-range dependencies, as sequential RNNs can, that they can outperform sequential semantic models and provide rich general-use compositional representations. We formulate several weakly-supervised to unsupervised tasks with a number of datasets to guide models towards efficient summarization.

2. Task

The aim of the project is to explore and improve upon the task of learning suitable latent structures that better encode the semantics of a sentence than syntax-based parsers. We explore different downstream tasks to evaluate the quality of sentence embeddings including Natural Language Inference and the new Conceptual Dependency task. Typically, the models generating a sentence embedding are relatively complex and non-trivial while the classifier itself is a simple cosine-similarity threshold [21], Support Vector Machine (SVM) [16] or single layer Fully Connected Network (FCN) [6].

Dataset: For the Natural Language Inference task, the Stanford Natural Language Inference (SNLI) [29, 6, 33] dataset is a large and fairly balanced corpus with 550k pairs of sentences for Training and 10k pairs each for test and validation. For the new Conceptual Dependency task, we curate a new dataset from a collection of textual course descriptions and provided prerequisite constraints.

2.1. Tree-Based Models

Compositional models of language are well supported by linguistic theory and philosophy [8, 17, 5]. Recent neural approaches to building compositional representations can be delineated by those that rely on external parsers [28, 19, 24, 5, 36, 13], and those that learn compositional functions as a by-product of parsing [3, 15].

Parse Tree Provided: Approaches which assume a parse tree is provided typically generate a parse and encode a sentence recursively from the bottom-up. Socher et al. train a composition function as a recursive auto-encoder which learns to recreate its full spanning subtree (unfolding version) [24]. To perform paraphrase classification, they construct a similarity matrix over all nodes and use dynamic pooling to obtain a fixed size matrix for classification. In 2015, Tai et al. introduced a more sophisticated Tree-LSTM composition function [28]. In the Child-Sum variant, the parent's LSTM cell treats the sum of its children's hidden states as the input from the previous time step. The forget gate is separately parametrized for each child representation, which allows for selective filtering of child content. They achieve state of the art performance on the SLICK textual entailment dataset [20] and Stanford Sentiment Treebank [25]. More recent works have implemented variants of the above approaches by adding additional parameters and/or introducing more syntactic bias [5, 36].

Parse Tree Learned: Given the inefficiencies of relying on external parsers, recent work has focused on efficiently generating parses whilst learning compositional representations as a by-product. Neural dependency parsing is categorized into transition-based parsing and graph-based parsing. We have chosen to focus on transition-based approaches given their recent success [4, 9, 3, 1, 15, 35] and linear-time speed.

Chen et al. create a classifier based on learnt distributional features, deviating from prior approaches which relied on sparse, hand-crafted features [4]. At every step, they extract a fixed set of word, part of speech, and arc label embeddings to predict an action according to the arc standard algorithm [22]. By pre-computing features for common words, they achieved a dramatic speed-up over existing methods. Dyer et al. create a novel stack LSTM, allowing them to continuously consider the full configuration space when making shift-reduce decisions [9]. The stack LSTM augments the traditional LSTM with a stack pointer which determines the t-1 state for all newly added (pushed) cells. The stack for partial subtrees computes a nonlinear transformation of head-word, dependent, and relation, which can be used as standalone encodings. They train and test their model on the Stanford Dependency Treebank [7] and achieve state of the art results. Bowman et al. build a fast batch-able parser called SPINN (Stack-augmented Parser-Interpreter Neural Network) [3]. SPINN is a standard neural shift-reducer parser augmented with a tracking LSTM, which reads in local context.

2.2. SNLI: State of the Art Baseline

We have implemented Stanford's SPINN shift-reduce dependency parser [3] in PyTorch and trained it on the SNLI dataset [2]. We have achieved state of the art performance, surpassing the previous reported results by a small margin. We obtained dependency parses for each sentence pair from the SNLI dataset from which we derive gold standard shift-reduce sequences. The composition function used in SPINN is a Tree LSTM [28]. Our experiments hold constant the SNLI classifier, while varying the degree of supervision over tree structure. The baseline model imposes a negative log likelihood loss on the classification decision, while following the shift-reduce sequence provided by the gold standard. Full teacher forcing is implemented during both train and test time and the model does not learn any shift reduce decisions. It simply learns a composition function over dependency relations as well as a final classifier. The second model implements a Tracking LSTM, which is used to predict shift-reduce action as well as to provide linear context to the composition Tree LSTM. As in the paper, the Tracking LSTM is updated with the concatenated representation of the top two subtrees on the stack and the top of the buffer and predicts a 2-way softmax (shift or reduce). The tree follows the gold standard shift-reduce sequences yet the loss function includes the negative log likelihood of the gold standard actions. The relative weight between sentence classification error and average shift-reduce error is parametrized by λ, which we anneal from 4.0 to 0.5 over the course of training. The loss function for a single training example (averaged over mini-batches) is given by:

$$Loss(s_i, s_j, \vec{A_i}^*, \vec{A_j}^*, y^*(s_i, s_j)) =$$
$$-\log P(y^*|s_i, s_j) - \frac{\lambda}{|\vec{A_i}^*| + |\vec{A_j}^*|}\left(\sum_{a \epsilon \vec{A_i}^*, \vec{A_j}^*} \log P(a)\right) \tag{1}$$

Where s_i is the premise sentence and s_j is the hypothesis sentence. $\vec{A_i}^*$ and $\vec{A_j}^*$ are the gold standard shift reduce sequences for the premise and hypothesis sentence, respectively. The fully unsupervised model sets λ to 0. The final model is fully unsupervised: the tracking LSTM predicts shift-reduce actions and the ensuing loss is computed over just the final sentence-to-sentence classification error. Actions which produce invalid states: empty buffer or empty stack, are replaced with the (only) valid action.

2.3. SNLI: State of the Art Results

Table 1. SPINN results. Track refers to use of a Tracking LSTM and Unsupervised refers to removal of teacher forcing and setting λ from Equation 1 to 0.

Model - SPINN	**Accuracy**	**Epochs**
no Tracking (Tree-LSTM)	83.6%	16
Supervised Tracking	82.9%	8
Unsupervised Tracking	83.6%	12

Hyperparameters for the experiments are listed in Appendix Table 7. We use an Adam optimizer with an initial learning rate of 0.001, betas of $(0.9, 0.999)$, and an epsilon of 1×10^{-08}. We trained models until convergence which for most experiments occurred after 5 epochs, evaluating every 50k batches. We use Batch Normalization [12] in addition to Dropout [26] after the input modeling layer and after each SNLI classification hidden layer.

Each model performs very well with small variation in results between them. Given the different training procedure and objective functions, this is a surprising yet encouraging result. Being unfettered from transition decisions seems to fully make up for the lack of direction during direction. The model converges in the same number of epochs which demonstrably highlights its ability to very quickly learn structure on its own. Or rather, that the downstream objective is robust/resilient to topology.

2.4. SNLI: Error Analysis

The SNLI dataset consists of pairs of premise and hypothesis sentences along with a label representing the semantic relation between premise and hypothesis: "entailment", "contradiction", or "neutral". The premise sentences are obtained from Flickr captions whereas Mechanical Turkers provided hypothesis sentences based on both the desired semantic label and the photo on which the premise caption is based. The hypothesis sentences are shorter in length on average than the premises. "Entailment" encapsulates implication in which the premise implies the *truthiness* of the hypothesis. "Contradiction" represents a *falsehood* in the hypothesis which directly contradicts the premise. Neutral indicates ambiguity - something "which *might* be true about the photo" [2]. Below, we show a confusion matrix for the baseline SPINN predictions on the full development set for the 3 labels.

Table 2. Rows represent true labels and columns represent the model prediction. Cells represent the row-normalized percentage for a given true label.

	Entail.	Contrad.	Neutral	Count
Entail.	89.76%	2.34%	7.90%	$3,329$
Contrad.	6.71%	82.76%	10.52%	$3,278$
Neutral	13.14%	8.66%	78.21%	$3,235$

Entailment Error Analysis The model performs best on entailment sentence pairs. The hypothesis sentences for entailment are predominantly condensed versions of the entailment, often with adjectives and prepositional phrases removed. The model correctly identifies that the premise: "a man in a red shirt is sitting on top of a rocky mountain" implies "a man sitting on top of a mountain". Such positive examples demonstrate that the models is not be distracted by extraneous information. Yet, the model is thrown off when there is lexical divergence between premise and hypothesis. It classifies the following pair as a contradiction: "a man is holding a child on his shoulders", "a child is riding on a man's shoulders". The dependency parses promote the subject of the sentence for which the difference: "man" versus "child" appears as a contradiction. Unsupervised models are not forced to promote lexical heads (subjects) in this same fashion. This error also highlights the potential for sentence-to-sentence attention to guide implicit alignment of sentences with semantic similarity but high lexical divergence. Similarly, the model struggles to properly identify entailment when the subject is removed from the hypothesis. It incorrectly classifies the hypothesis "the machinery is very large" given the premise "two men stand near a piece of large machinery" as neutral, probably because there is no mention of the men in the hypothesis. The dependency tree highlights the "two men" which are unidentified in the hypothesis and hence the model fails to validate the truth of what it does mention "the machinery".

Contradiction Error Analysis The model performs relatively well in predicting contradictions. The model is able to identify the the instances where the actor of the pair of sentences is the same and the actions can not be performed together. For example, it rightly identifies "a chef is cutting a piece of fish with a chefs knife ." and "the chef is at home, fast asleep." as contradiction. But the model lacks general-world knowledge and reasoning. It falsely identifies the following pair as neutral (instead of contradiction): "a boy splashing through the ocean ." and "the boy is in Kansas.", indicating that the model fails to recognize that Kansas is landlocked. We check the training data to see if it included the knowledge. The training data has just one example, where "Two kids running down a sand dune to the beach.", "The kids are on vacation in Kansas." is labeled as contradiction. The model fails to fine-tune to this due to the sparsity of examples. Another interesting pair worth noting is: "a car is loaded with items on the top." and "the car is a convertible ." which has been classified as "neutral". This shows the model's failure to recognize that convertibles can't be loaded on top. We search for this fact in the training data. Among all the examples involving convertibles, we find some pairs of sentences which indicate that a convertible is an open-topped car. But the pair "A baby sitting on a blue convertible." and "Small child on top of car." has been gold-labeled as "entailment", which seems erroneous and may have lead to the incorrect prediction in the previous example.

Neutral Error Analysis Neutral is the hardest category according to classification accuracy results. It is also the class with the lowest human inter-annotator agreement [2]. Consider the following false positive where the true label is "entailment" but the predicted label is "neutral": the premise is "two men stand near a piece of large machinery ." and the hypothesis is "the machinery is very large". There is no mention of "men", the main entity in the premise, in the hypothesis yet there is no contradiction which makes neutral a plausible prediction. This also relates to the distinction between "subject" and "topic" which is a non-trivial notion in language. However, in this example "large" is not identified as a modifier of "machinery" in the parse. Such parsing errors contribute to the semantic NLI classification error and add to its inherent difficulty. False negatives include harder problems such as identifying (direction of) causality from lexically very similar input and limitations in identifying semantic equivalence when the grammatical construction is very different. An example of the first is the premise-hypothesis pair: "a gas station and a car on fire" and "a car sets a gas station on fire". The model is influenced by the surface overlap and needs to learn subtle causality markers. Another false-negative example is the following sentence pair where the true label is "neutral" but the predicted label is "entailment": "a young man does tricks on his skateboard" and "a guy rolls forward". The sentences differ lexically but exhibit constructional similarity and strong correspondence between the subject in the two sentences: "a young man" and "a guy".

We also analyze the accuracy by hypothesis sentence length which, as expected, declines as sentence length increases as shown in Table 8 in the Appendix. The result is consistent with general understanding about sentence modeling. Longer sentences are more difficult to comprehend and condense into a fixed length vector. We discuss ways to deal with longer sentences in the improvement section.

In order to understand label correlation with surface patterns in the sentences, we fit a multinomial logistic regression model on the data using manually engineered features. We designed and extracted the following features: word overlap between the premise and hypothesis (X_1), length of each (X_2, X_3), difference in length (X_4), and whether the premise or hypothesis contains a negation (X_5, X_6). We mapped the classes into ordinal (contradiction, neutral, entailment) scale to study the association between each predictor/feature and the label as a response variable. We consider the entailment category as the reference category.

$$ln(\frac{\pi_{\text{neutral}}}{\pi_{\text{entailment}}}) = -2.07 - 0.27X_1 - 0.04X_2 + 0.45X_3 + 0.05X_4 - 0.01X_5 - 0.25X_6 \quad (2)$$

$$ln(\frac{\pi_{\text{contradiction}}}{\pi_{\text{entailment}}}) = -1.47 - 2.78X_1 + 0.09X_2 + 0.24X_3 - 0.07X_4 - 0.13X_5 - 0.31X_6 \quad (3)$$

We note the following interesting findings. The length of the hypothesis strongly influences the prediction and is associated with higher chance of neutral followed by contradiction predictions compared to entailment. Overlap between the premise and the hypothesis reduces the chance of contradiction. Negation in the hypothesis sentence has far more influence on prediction compared to negation in the premise sentence. Also a longer hypothesis is more likely to result in contradiction prediction while a longer premise is more likely to result in neutral prediction though the individual length of the premise/hypothesis is a weaker predictor. Since the goal of the project is to learn general sentence level representation, we have not incorporated these features in our models. However, we propose to integrate these features for specifically improving SNLI in future work.

2.5. Conceptual Dependency

We also experiment with different approaches for a new task, along the line of semantics and meaning representation, where the goal is to learn the *conceptual dependency* between two documents. Document i is conceptually dependent on document j if understanding of the former requires information from the latter. In the educational domain, the closest analogy to conceptual dependency is prerequisite relation. The task of predicting the prerequisite relation between two courses was first introduced in [30] where bag-of-word representation of courses with partially observed prerequisite relations are mapped into a common vocabulary space, the concept graph, and inference is done in this space by adjacency matrix completion and then projected back onto the document (course) space to infer missing links in the course space. We refer to this model as CGL. More recently [33] present results on a fine-grained variant of the problem where dependencies are learned between atomic *concepts* not entire documents. In [33], various classifiers including Naive Bayes, Logistic Regression, SVM, and Random Forest are used with a heavily engineered set of lexical, contextual and structural features.

We present first neural results on prerequisite prediction and compare results with the only available baseline on the task [30]. Since the data does not have enough linguistic structure (most courses are described using a sequence of short sentences or phrases representing concepts and topics), and for comparability with the baseline, which did not employ syntactic features, we ran the unsupervised SPINN model, a convolution neural network model (CNN) with variable width convolutions, and a hierarchical attention network. The CNN model we implemented is a deep model based on the state-of-the-art CNN for sentence classification [14] with several modifications in filter design, regularization and hyperparameters. ReLU activations were used and a softmax for the final layer. The model was trained for 20 epochs using RMSprop on categorical crossentropy loss with mini-batch size of 64. For comparability with the baseline, AUC metric was implemented and the model was further evaluated on this metric. The model was implemented in Keras which does not support AUC so it was implemented via callbacks. We used the same dataset of course textual descriptions in [30] but curated it for the neural classification task. The final data set contains 6700 instances. We note the extreme skewness of the data since only a very small subset of course pairs are positive instances of the prerequisite relation. After creating pairs of courses, we randomly undersample the negative class such that the resulting distribution is still comparable to the baseline. This results in a dataset where the positive class constitutes 26% of the data. We finally shuffle the instances to avoid bias in learning. We outperform the baseline by a statistically significant margin. Although the baseline is not neural, we note that it is based on a large-scale optimization and has nearly a billion parameters in the original space whereas our neural model has 2.8 million parameters. We have also run the hierarchical attention model of Yang et al. [31] which is the state-of-the-art neural model for several document classifications tasks [31]. This model achieves best performance on the prerequisite prediction task (+1.7 points in prerequisite prediction accuracy compared to the best performing CNN variant) as shown in Table 3 but it takes considerably longer time to converge. We therefore compare AUC results to the CGL baseline using the more efficient CNN model.

Model	Accuracy
SPINN	83.5%
CNN-variable convolutions	84.1%
Hierarchical Attention	85.7%

Table 3. Performance of different neural models on prerequisite prediction: Accuracy

To compare with baseline results, we run experiments on course datasets from different sources and report results in Table 4 next. Overall, the neural model outperforms the adjacency matrix completion approach of [30]. We note a drop in performance on the MIT dataset where the baseline already achieves large AUC value.

dataset	CGL	CNN	Δ
Combined	89%	96%	+7
MIT	96%	94%	-2
CMU	79%	97%	+18
Princeton	92%	97%	+5

Table 4. CGL vs. neural model: AUC

The baseline approach explicitly enables transfer learning by mapping the course documents into a canonical concept space. The neural model does not explicitly model transfer but enables it indirectly through the distributed representation of words (embeddings). Table 5 presents transfer learning results where the model is trained on data from a given source and tested on different target data. The neural model does enable transfer learning to some extent and more interestingly enables transfer from a smaller to a larger dataset (last row) which was not possible with the CGL baseline.

source	target	CGL	CNN
MIT	Princeton	72%	66%
MIT	CMU	70%	82%
CMU	Princeton	NA	69%
Princeton	CMU	NA	63%

Table 5. Transfer Learning Results: AUC

3. Latest Research

3.1. Latent Structure Learning

Our goal for the semester is latent learning of dependency structure through downstream semantic tasks. Unsupervised tree structure learning is non-trivial due to the fact that tree construction decisions are hard and thus non-differentiable. For the unsupervised version of SPINN shown above, the shift reduce predictor receives no error signal from the downstream sentence classification. The loss function propagates its error signal solely to the composition function. Creating a direct feedback link between downstream prediction requires either introducing a fully differentiable stack [11], or Reinforcement Learning [32]. We describe work and experiments on these two improvements to the SPINN model next.

3.1.1 Continuous Stack

A continuous stack, as introduced by [11], enables soft push and pop operations by introducing a valence vector indicating the confidence of the previously "hard" action. We implement a continuous stack version of SPINN in which each timestep involves a combination of a shift and a reduce with varying strengths. Read operations are a weighted average of the top element(s) of the stack according to a probability waterfall defined by the valence vector. Both push and pop scores are pushed through a sigmoid to produce values between 0 and 1 with expectation 0.5. We explore mechanisms which encourage coherent tree structures and decisive actions while remaining with minimal supervision. Full non-supervision of tree structure learning is intractable given the propensity for the model to diverge to dominant actions (shift or reduce). Guiding the model toward producing a singular representation whilst maintaining minimal supvervision is a very difficult task. To accomplish this, we try several training modifications. First, we train the model using teacher forcing, where we treat a shift action as having "1" valence, and a reduce as having "0" valence. We anneal the cost associated with the teacher loss function: the λ in Equation 1. After a few epochs (2-3), we turn off teacher forcing and allow the model to make its own actions. This strategy, however, proved unstable and produced a divergent model. We think future work should explore ways to gradually move from binary teacher forced valences to continuous actions during training exploration, in addition to gradually removing teacher forcing of actions from the loss function. This would avoid having the rug pulled out from under the model. We also explore a model which imposes a cost on the MSE between the sum of the shift and reduce valences. Standard tree S-R construction requires N shift operations and N-1 reduce operations. Hence we measure mean squared divergence between the sums of both shift and reduce valences for the model. While stabilizing early training, this does not solve the divergence issue. We think that annealling the action space from coherent to model-driven represents the best hope for the continuous stack. We present results for fully unsupervised continuous stack since we do not receive a boost from the enhanced training techniques.

3.1.2 Reinforcement Learning

Yogatama et al. [32] propose Reinforcement Learning for a variety of NLP tasks, including using policy gradient methods to learn latent tree structures for downstream semantic objectives. Their method, while effective, converges very slowly. Hence, we introduce a more sophisticated reward structure for training REINFORCE which imposes a cost for deviations from balanced trees in order to more quickly align the topology selection with more robust hierarchical structure. We experiment with several distance formulas and determine that the optimal structural variable on which to impose a cost is tree height. Tree height for purely left-branching or purely right-branching trees is the length of the sentence. The minimal height for binary trees is $log(|sentence|)$. To encourage compact hierarchical representations which group related, like terms, we create a reward for tree height as such: $(height_{expected} - height_{actual})/height_{expected}$. We define expected height as the average of minimal height and max height separately for each tree. We interpolate the reward between structure and downstream objective according to a hyperparameter which we set to 0.25 (less weight on the structural reward). In order to incorporate structural rewards, we have to provide the action history to the tracking LSTM or else it won't be able to identify the source of the structural reward. To do this, we learn embeddings for shift and reduce actions which are fed into the tracking LSTM at every timestep (for the previously chosen action). We choose 20 as the embedding dimension. Furthermore, to model sequential action selection, we diverge from Yogatama et al. [32], who use a static MLP to predict actions (receive no prior state information), to using the SPINN tracking LSTM (modified to serve as a Policy Network from which actions are sampled according to a multinomial distribution). During training, actions are sampled and during test time, actions are taken as the argmax according to the Policy Network at each timestep.

3.2. Bi-SPINN

Given the tremendous success of bi-directional LSTMs on many NLP tasks, we apply the general forward-backward paradigm to the SPINN model. Bi-LSTMS can help mitigate the issue of early summarization which is heightened for longer contexts. Each pass need not summarize the entire preceding context accurately, given that there are two passes. Our model performance, as expected, degrades as context length increases so we hypothesized that bi-directional SPINN might help bridge the gap. To accomplish Bi-Directional SPINN (Bi-SPINN), we run SPINN both in the forward and backward directions, treating the final sentence summarization as the concatenation of the top of the stack after the forward pass with that of the backward pass. Given that language is generated left to right, topological modeling from right to left is an NP-hard task. We find that adding the right to left modeling does not add value above and beyond left to right modeling, and we hypothesize that further help is needed to guide the process for the backward pass. We introduce a few possible extensions in the Future Work Section 5.

3.2.1 Tracking LSTM Modeling Layer

The Tracking LSTM reads in local context: top of the stack and buffer, at each timestep. In this sense, it represents a hybrid sequential-hierarchical summary of the context. In order to make use of this for sentence representation, we attempt a model which stores the tracking state at each timestep as a sequential representation of the reading process. We then fuse this information into a single vector by passing it through a Bi-LSTM. We do this on Bi-SPINN by concatenating the tracking LSTM's hidden states at each timestep for the forward and backward passes, respectively. We take the concatenation of the final output of both the forward and backward passes of the modeling LSTM as an additional sentence summarization. It encapsulates the cumulative forward and backward reading of the sentence. We then concatenate the forward-backward SPINN pass with the output of the modeling layer and perform classification between the two sentences. Due to additional model complexity, however, this model diverges. We think that like with Bi-SPINN, enhanced regularization could prevent such over-fitting but initial experiments with higher L2 norm does not alleviate the over-fitting problem.

3.3. Fine-tuning with Training Oracle

Oftentimes proper sentence modeling relies more on simple external knowledge regarding rare terms rather than deep word-level analysis. While analyzing the model performance, we observe that the model misses general external knowledge that was sparse in the data. For example, it failed to infer that Kansas is landlocked, because there were just 10 examples involving Kansas in the training data and only 1 of them indicated that it is landlocked. The model struggles to perform well on such examples given its lack of exposure to the rare term, and no external knowledge base fallback. To mitigate this issue, we gain inspiration from the field of Machine Translation.

Given the success of Information Retrieval on training data in Machine Translation [10, 18], we apply a similar method to improve our model. The general idea is to fine-tune the model on the training data that is similar to the test data and then attempt the translation.

On directly adopting the method on the discriminative task of sentence classification, we found that the model is very prone to overfitting. Therefore we explore different similarity metrics which might give us an improved subset of training examples to finetune the model.

We implement a re-scorer which extracts all test examples for which a high IDF-term appears in both the premise and the hypothesis. We then leverage the other examples to find training examples which possess a high similarity score to the given test example. We implement three different methods for computing similarity between a test example query and related training example(s). They progress from coarse to more sophisticated measures. We first compute a term frequency dictionary and resulting IDF scores for every vocabulary term in the training dataset. Based on these IDF scores and a threshold, we determine and store a set of rare *concepts* across the vocabulary. For a given test sentence pair, we look for the presence of any one of those concepts first in the hypothesis. If such a concept exists, we extract all training hypothesis sentences which contain this concept $>= 1$ times. We, then, repeat this process separately for the premise sentence. This produces a separate list of hypothesis and premise sentences which share conceptual relations to the current test example. We then take the intersection of such training sentence pairs as highly relevant examples for fine-tuning. We fine-tune the model on this subset and re-score the test example. We take the score post fine-tuning to produce a final prediction.

For ensuring that the chosen training examples are contextually closer to the test sentence pair, we further filter the selected set using:

$$min(Jaccard\_score(test\_hyp, train\_hyp),\\ Jaccard\_score(test\_prem, train\_prem)).$$

After manually inspecting the examples, we set the threshold to 0.15. Jaccard Score provides the similarity between two sentences, disregarding sequential information.

$$Jaccard\_Score(S_1, S_2) = \frac{Intersection(S\_1, S\_2)}{Union(S\_1, S\_2)}$$

The final similarity metric with which we experiment is to use tree edit distance as defined by Zhang and Shasha [34]. We use the python package *zss* to compute it (in particular, the simple_distance function).

Computational Complexity of Re-training: If implemented naively, the suggested process can be exceedingly slow, given that both retrieval and training are slow processes. In order to speed this process, we make retrieval faster by indexing the training dataset as a pre-processing step. Also, we set the threshold for the selection of training examples high to ensure that re-scoring is done only if clean training example set is obtained. With this threshold, re-scoring is triggered for 6.5% of the test cases, which is ideal as re-training is a slow process.

3.4. Sentence2Sentence Attention

Our error analysis reveals that oftentimes the hypothesis focuses on a subset of the premise sentence. Hence, inter-sentence attention should enable each sentence to filter out irrelevant, additional, information. This information can distract and confuse the model. To incorporate attention, we try two different methods.

For the first, we first encode each sentence with a Bi-LSTM and initialize the *other* sentence's tracking LSTM with its final state. This should promote more task-aware dependency parsing. We have run experiments with this enhancement but unfortunately find that it does not improve results. We think word-level co-attention might be more influential because it is more fine-grained than conditioning the tracking LSTM on a single vector summary of the entire other sentence. Our results for REINFORCE and continuous stack use this weak attention modeling as the default. No other experiments to our knowledge do, but either way, it does not move the needle.

The other method of incorporating involves encoding Premise using SPINN first and then providing the Sentence embedding of Premise to the LSTM performing Reduce actions for Hypothesis .

$$attend_t = (A^l_{t-1} + A^r_{t-1}) * A_{premise}$$

$$A_t = attend_t * W_a + C_t$$

And finally adding this attention to the hidden state:

$$h_{new} = h_{orig} + tanh(attend_t)$$

The motivation is to apply the attention of Premise over the child cell states and decide the hidden state of the parent accordingly. However, we found this sort of attention to be highly aggressive as the final classifier continues to get premise embedding in addition to hypothesis with heavy attention over premise, causing it to fail while differentiating the nuances in the hypothesis.

4. Latest Results and Analysis

4.1. Results

4.2. SNLI: SPINN Enhancements

Though the number of epochs are different, in each case we trained the model until the first of either a plateau in accuracy, a divergence (as was the case with continuous stack), or have not converged yet (slow training marked by *).

4.3. Analysis

With the above set of experiments we find that the SPINN baseline in its original form is robust.

Table 6. SPINN enhancement results.

Model - SPINN	**Accuracy**	**Epochs**
SPINN Baseline	83.53%	16
Continuous Stack	67.3%	1
Bi-SPINN	81.8%	5*
Bi-SPINN + Modeling Layer	65.9%	1*
REINFORCE	79.3%	5*
REINFORCE w/ Height Reward	78.7%	3*
IR Re-Scoring (Rare Words)	82.81%	NA
IR Re-Scoring (Jaccard)	83.57%	NA
IR Re-Scoring (Jaccard + TreeEdit)	**83.59**%	NA
Attention (3.4)	77.61%	7

The continuous stack - both with initial teacher forcing and fully autonomous - diverges when the tracking LSTM is allowed to produce shift-reduce valences on its own. The margin for error in producing soft S-R reduce actions which result in the final read vector being a sentence summary is very small. For instance, the model can push and pop $2N - 1$ times and, as long as the final S-R valence vector is weighted more heavily to a shift, the final read vector may be just the final word's hidden state, rather than a full sentence subtree. This causes the model's sentence summaries to be wildly variant and eventually causes the model to diverge. More work is needed to transition the soft actions from binary teacher forcing to autonomous (sigmoid) but to a certain extent, this negates the purpose of the continuous stack (no transitions). The continuous stack in this sense might be best served as a way to fine tune an MLE model with predefined transitions to a specific objective.

With REINFORCE, training is very slow to train which and does not offer much upside above and beyond SPINN. Thus, given SPINN's robustness/resilience to both good and bad topology, REINFORCE does not seem justified to the task of sentence modeling. Further imposing a height constraint does not add much value or speed up training, given that REINFORCE usually works best with minimal action manipulation. The beauty of Reinforcement Learning is its simplicity and ability to navigate complex action spaces with a simple downstream objective. Tampering with this simplicity provides little benefit. Both REINFORCE models (with and without augmented rewards), take very long to train and the latest versions have not reached convergence yet after a few days of training. There seems to be more capacity in the model but we do not think too much based on prior experiments where results plateaued right around the baseline numbers (82-83%).

With Bi-SPINN we hope to obtain more contextual information, however the process is restrictively slow, clocking just 5 epochs in 100 hours. Each Bi-SPINN involves 2 times SPINN run for a data-point. Another interesting observation is that we expected a steeper decay in training loss due to the explosion in the number of parameters on moving from SPINN to Bi-SPINN, expecting overfitting in the later epochs. However, the observation was the contrary, the rate of reduction in training loss for Bi-SPINN is slower. We are continuing to investigate if this trend is followed in later epochs as well. Furthermore, adding a modeling a layer has not improved results yet. We are running more experiments with increased regularization in light of adding many more parameters but as of now, we see no improvement.

For the IR re-scoring, we see gradual improvement in results as the similarity metric becomes more narrow/precise. The best scoring model involves gathering examples according to a threshold based on maximum tree edit distance. The re-scorer is very conservative with regards to changing its label prediction. The re-scoring was triggered in only 6% of the cases. But when it does, it more often than not improves the label prediction. Of the 688 instances, the re-scoring made the prediction better 11 times and made it worse 6 times. Though this is not the ideal scenario, but re-scoring helps in 65% cases, making it improve the overall performance.

5. Future Work

5.1. Bi-SPINN with Regularization

Given the success of Bi-LSTMS we think that there is promise in pursuing bi-directional recurrent networks through structure. The biggest impediment relates to guiding the backward pass. Given that language is constructed from left to right, it is much harder to determine optimal Shift-Reduce sequences from right to left. In a working Bi-SPINN model, the left and right passes should impose a conformity constraint on one another. In other words, each pass should learn from the reverse pass through a soft regularization constraint on tree construction. Penalizing this properly might require REINFORCE or soft tying of SPINN parameters.

5.2. Self-Attention

Self-Attention has proved immensely successful to natural language tasks. Incorporating proper context modeling prior to tree construction might guide the model to more coherent balanced structures. Even passing the leaf nodes through a Bi-LSTM would be a good first step. Alternatively, one could reasonably propose a model which makes two passes through the SPINN parser. The first pass builds up a summary representation of the sentence - or an array of partial subtrees - and uses it as an attention query vector for the next pass. This would be commensurate with reading a sentence first to determine the salient message, and then re-reading with this in mind. Bi-Directional Attention Flow (BiDAF) would be a simple way to reincorporate this summary context into the next pass [23].

6. Conclusion

We successfully reproduce the results of the SPINN paper but fail to produce significant statistically significant contributions beyond the baseline. We attempt a multitude of approaches: continuous differentiable stack 3.1.1, REINFORCE with augmented structural rewards and prior action tracking 3.1.2, IR-based fine-tuning and re-ranking 3.3, bi-directional SPINN 3.2, and a tracking state modelling layer 3.2.1. We hypothesize that the failure of these meaningful, well-motivated extensions to produce concomitant increases in the test accuracy on SNLI amounts to the robustness of SPINN to modifications in topological structure. The unsupervised SPINN model trains almost identically to the fully supervised version of SPINN. Minor modifications in the tree structure do not seem to impact its representative power for classification. After doing extensive experimentation, we still believe in the pragmatism and feasibility of our extensions but feel that richer leaf level representations are required in order to guide the tree to better learn word-level interactions. In other words, more dramatic steps to fuse local context before tree constructions is likely needed to push the model over the plateaued state in which it currently finds itself. We learned a lot from each of our efforts and hope our well-motivated research hypotheses can spur further research in the exciting area of compositional representation learning.

Appendix

6.1. SPINN hyperparameters

Table 7. SPINN hyperparameters across experiments.

Name	**Value**
Batch Size	32
Learning Rate	0.001
Embedding dims	300
Gradient threshold	5
LSTM Hidden dims	150
SNLI MLP hidden layers	2
SNLI MLP hidden dims	1024
Dropout rate	0.1
Teacher forcing lambda init	4.0
Teacher forcing lambda end	0.5
L2-norm weight decay	1e-5

6.2. Error distribution by Hypothesis Length

Table 8. Accuracy by hypothesis sentence length.

Length	Accuracy	% of data
0-5	89.0%	6%
5-10	83.2%	66%
10-15	81.5%	23%
15-20	77.0%	4%
20+	79.5%	1%
W Avg	83.6%	100%

Distribution of work

Janani worked on IR based fine-tuning, Bi-SPINN (without REINFORCE and modeling layer), REINFORCE (stochastic version), and Attention (2).

Fatima Al-Raisi implemented the variable width convolution CNN model and hierarchical attention model, and curated the course prerequisite dataset, ran prerequisite prediction and transfer learning experiments, designed the sentence features for SNLI and explored predictor variables for SNLI by fitting a multinomial regression model, contributed to the error analysis of SPINN and proposed the Bi-SPINN model.

Griffin Adams worked on the continuous stack, REINFORCE with/without structural rewards, tree edit distance for the IR model, the hierarchical modeling layer, and Bi-SPINN.

References

[1] D. Alvarez-Melis and T. S. Jaakkola. Tree-structured decoding with doubly-recurrent neural networks. 2016. 1

[2] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015. 2, 3

[3] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*, 2016. 1, 2

[4] D. Chen and C. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014. 1, 2

[5] J. Cheng and D. Kartsaklis. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. *arXiv preprint arXiv:1508.02354*, 2015. 1

[6] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017. 1

[7] M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Genoa Italy, 2006. 2

[8] D. Dowty. Compositionality as an empirical problem. *Direct compositionality*, (14):23–101, 2007. 1

[9] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*, 2015. 1, 2

[10] M. A. Farajian, M. Turchi, M. Negri, and M. Federico. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, pages 127–137, 2017. 6

[11] E. Grefenstette, K. M. Hermann, M. Suleyman, and P. Blunsom. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1828–1836, 2015. 5

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 2

[13] M. Iyyer, J. L. Boyd-Graber, L. M. B. Claudino, R. Socher, and H. Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014. 1

[14] Y. Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. 4

[15] E. Kiperwasser and Y. Goldberg. Easy-first dependency parsing with hierarchical tree lstms. *arXiv preprint arXiv:1603.00375*, 2016. 1

[16] Z. Kozareva and A. Montoyo. Paraphrase identification on the basis of supervised machine learning techniques. In *FinTAL*, pages 524–533. Springer, 2006. 1

[17] J. Li, M.-T. Luong, D. Jurafsky, and E. Hovy. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*, 2015. 1

[18] X. Li, J. Zhang, and C. Zong. One sentence one model for neural machine translation. *arXiv preprint arXiv:1609.06490*, 2016. 6

[19] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017. 1

[20] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *SemEval@ COLING*, pages 1–8, 2014. 1

[21] R. Mihalcea, C. Corley, C. Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006. 1

[22] J. Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*. Citeseer, 2003. 2

[23] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016. 8

[24] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011. 1

[25] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. 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, 2013. 1

[26] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014. 2

[27] Z. G. Szab¡. Compositionality. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2017 edition, 2017. 1

[28] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015. 1, 2

[29] Y. Wilks. A preferential, pattern-seeking, semantics for natural language inference. *Artificial intelligence*, 6(1):53–74, 1975. 1

[30] Y. Yang, H. Liu, J. Carbonell, and W. Ma. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 159–168, New York, NY, USA, 2015. ACM. 4, 5

[31] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489. The Association for Computational Linguistics, 2016. 4

[32] D. Yogatama, P. Blunsom, C. Dyer, E. Grefenstette, and W. Ling. Learning to compose words into sentences with reinforcement learning. *arXiv preprint arXiv:1611.09100*, 2016. 5, 6

[33] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014. 1, 4

[34] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989. 7

[35] X. Zhang, L. Lu, and M. Lapata. Top-down tree long short-term memory networks. *arXiv preprint arXiv:1511.00060*, 2015. 1

[36] Y. Zhou, C. Liu, and Y. Pan. Modelling sentence pairs with tree-structured attentive encoder. *arXiv preprint arXiv:1610.02806*, 2016. 1