Neural Networks for Natural Language Inference

Sebastian Schuster
Department of Linguistics
Stanford University
sebschu@stanford.edu

Abstract

Predicting whether a sentence entails another sentence, contradicts another sentence, or is in a neutral entailment relation with another sentence is both an important NLP task as well as a sophisticated way of testing semantic sentence encoding models. In this project, I evaluate three sentence encoding models on the Stanford Natural Language Inference (SNLI) corpus. In particular, I investigate whether the incorporation of syntactic information in the form of dependency tree labels into a recurrent model leads to better sentence representations. I confirm previous results that show that LSTM-RNNs outperform a simple sum-of-words baseline but my results also suggest that this simple method of incorporating syntactic information has no stable positive effects on the performance of the model.

1 Introduction

The task of natural language inference is to determine whether a given hypothesis sentence is entailed by a given premise sentence. For example, the premise "On April 21st, Barack Obama met with officials in Cuba." entails the hypothesis "On April 21st, Barack Obama was in Cuba." while it does not entail the hypothesis "On April 21st, Michelle Obama was in Cuba.". This is an important task because many other natural language understanding tasks can be framed as NLI problems. For instance, if we want to extract the relations between entities from large web corpora, we can formulate hypotheses of the form "entity-1 relation entity-2" and check whether any of these candidate hypotheses are entailed by a sentence in the corpus. Other applications for which the notion of entailment is highly relevant include commonsense reasoning (e.g., [2]) and semantic parsing.

The automatic prediction of entailment relations between two sentences can be very challenging because a system has to be able to reason about many complex semantic phenomena. For example, it has to know about the monotonicity properties of quantificational determines such as *every* and *some* and that *every NP VP* always entails *some NP VP* but not the other way round. Further, for almost all sentence pair, the system is required to have some world knowledge to make the right prediction.

In recent years, there has been a lot of interest in applying different neural network models to this task (e.g., [3]) in order to investigate whether such models can learn to predict entailment relations from examples. Several models have shown promising results [21, 13, 16, 7, 5] but based on these results it is still not entirely clear whether incorporating syntactic information into the model leads to performance gains that cannot be attained through other means. In this paper, I am taking a first step towards answering this question and I am investigating whether the incorporation of dependency labels into a sequence-based model leads to performance improvements on the entailment relation prediction task.

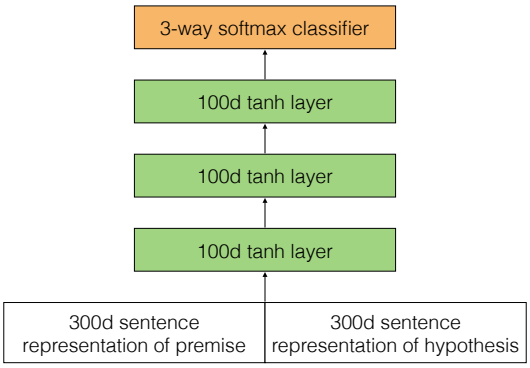

Figure 1: Basic network architecture. (Adapted from [4].)

2 Task and Data

The task of predicting entailment relations is formally defined as following. Given a premise sentence p and a hypothesis sentence h, we want to predict the entailment relation $r \in \{\text{entailment}, \text{neutral}, \text{contradiction}\}$. p entails h iff whenever p is true h is also true, p contradicts h iff whenever p is true h is false, and p and h have a neutral entailment relation if they neither entail nor contradict each other.

For my experiments, I use the Stanford Natural Language Inference (SNLI) corpus which was collected by [4]. This corpus contains 570,152 sentence pairs, split into a training, development, and test set of size 550,152, 10,000, and 10,000, respectively. Each sentence pair contains a premise and a hypothesis and a label indicating the relation between the premise and hypothesis: *entailment*, *neutral*, or *contradiction*. 2% of the sentences do not have a gold label as no label was assigned by the majority of the MTurk workers. I exclude these examples from my experiments.

3 Models

I follow [4] and use the same basic architecture with different sentence encoding models. The architecture of my model is illustrated in Figure 1. I encode both the premise and the hypothesis as a 300-dimensional sentence vector, concatenate these vectors and feed them through three 100-dimensional hidden units with a tanh non-linearity. The final predictions are output by a 3-way softmax projection layer. For each layer, I use a different weight matrix W_i and a different bias term b_i. The complete foward-propagation given two sentence vectors s_p and s_h is then:

$$
\begin{aligned}
h_1 &= tanh([s_p; s_h]W_1 + b_1) \\
h_2 &= tanh(h_1 W_2 + b_2) \\
h_3 &= tanh(h_2 W_3 + b_3) \\
\hat{y} &= softmax(h_3 U + b_s)
\end{aligned}
$$

As a loss function, I use cross-entropy loss and I apply L_2-regularization to all model parameters θ (all weight matrices W_i and U). I therefore end up with the following total loss function.

$$loss(\hat{y}, y) = CE(\hat{y}, y) + \frac{\lambda}{2}||\theta||_2^2$$

3.1 Sentence models

I evaluate three different sentence models, namely a sum-of-words baseline, a recurrent neural network (RNN) with Long short-term memory (LSTM) units [9], and an LSTM-RNN with words and

dependency labels as inputs. I initialize the word embedding matrix of all models with the pre-trained 300-dimensional GloVe word vectors [15].

Sum of words The sentence representation of this model is simply the sum of the word vectors of the individual words. Formally, given a sentence $x_1, ..., x_n$ where x_t is the corresponding word vector of the t-th word in the sentence, the sentence representation s is $s = \sum_{i=1}^{n} x_t$.

LSTM-RNN This model is based on an LSTM-recurrent neural network. At each word t of the sentence, given the word vector x_t, we compute the hidden layer h_t as following.

$$
\begin{aligned}
i_t &= \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right) \\
f_t &= \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right) \\
o_t &= \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right) \\
u_t &= tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right) \\
c_t &= i_t \circ u_t + f_t \circ c_{t-1} \\
h_t &= o_t \circ tanh(c_t)
\end{aligned}
$$

i_t is an input gate, f_t is a forget gate, o_t is an output gate, and c_t is a memory cell. The sentence representation s in this model is the hidden layer h_n of the last word n of the sentence.

LSTM-RNN with dependency labels This model has the same architecture as the previous model but instead of only using the word vectors as input to the RNN, I concatenate the 300-dimensional word vector and a 10-dimensional dependency label vector. For each word, I use the dependency label of the relation between the word and its head, or the special relation *root* for the head of the sentence. I obtain these labels by converting the constituency trees of the SNLI dataset to English Universal Dependencies (UD) [8, 14] using the Stanford UD converter [17]. The embedding matrix for the dependency labels is randomly initialized and learned during training. The sentence representation is again the hidden layer h_n of the last word of the sentence.

3.2 Implementation and Tuning

I implemented all models with TensorFlow [1]. For the LSTM-RNN models, I use the default implementation in TensorFlow. I train all models with the Adam optimizer [10] with a learning rate of 0.001. I apply dropout to the output of the sentence encoding models with a dropout probability of 0.1. I use mini-batches of size 64 and I tune each model for 10 iterations and keep the model with the highest prediction accuracy on the development set. Considering the relative high runtime of the training procedure, I did not perform extensive hyperparameter tuning. Nevertheless, I experimented with several different regularization strengths, and ended up with $\lambda = 0.001$ for the sum-of-words model and $\lambda = 0.00001$ for the LSTM-RNN models. I also experimented with different dimensions of the hidden layer of the LSTM (100 and 300) and found that a hidden layer with 300 units works best. Further, I also tried using another hidden tanh-layer that projects the 300-dimensional word vectors into 100-dimensional word vectors before they are used as input to the RNN as proposed by [4] but this also hurt performance and therefore I do not use this layer in my final implementation.

4 Results and Discussion

The upper part of Table 1 shows the prediction accuracies of my three models on the training, development, and test set. These results show that the LSTM-RNN significantly outperforms the sum-of-words baseline. The dependency labels lead to an improvement in accuracy of almost 1 percentage point on the development set but don't seem to improve the predictions on the test set.

Learning Figure 2 shows the accuracy of my models after each epoch. This plot shows that all models converge relatively fast and that the two LSTM models are able to fit the training data in a better way which is not surprising considering that these models have a lot more parameters than the sum-of-words baseline.

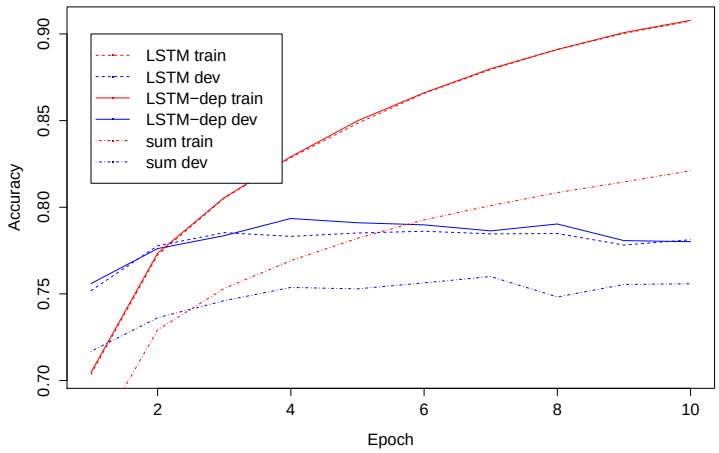

Figure 2: Accuracy of the different models (sum-of-words, LSTM-RNN, LSTM-RNN with dependency labels) after each training epoch on the training and development set.

Qualitative comparison In order to assess the effect of including the dependency labels on the performance of the RNN, I manually compared some of the predictions of the two RNN models (henceforth *basic* and *deplabel*) on the development set. First of all, adding the dependency label information did not only improve the predictions, it also led to wrong predictions on some sentences that the *basic* model got right. Nevertheless, as shown in Table 1, it had an overall positive effect.

If we analyze the individual sentences that the *deplabel* model got right and the *basic* model got wrong, then there don't seem to be any specific constructions for which the dependency labels helped and many of the differences in output between the two models seem random. However, one notable thing was that the dependency labels seem to help the model determine which parts of the sentence are part of the main clause. This seems to improve performance when the premise contains a relative or adverbial clause and the hypothesis mainly focuses on information contained in the main clause. For example, the *deplabel* model got the following sentences right, while the *basic* model predicted the wrong label.

Premise: A young man is pushing a lawn mower to mow the grass.
Hypothesis: A guy rides a lawn mower in the grass.
Label: Contradiction

In this example, the *deplabel* model seems to be better at focusing on the main predicates *pushing* and *riding* and is therefore able to predict that these two actions contradict each other. This also seems to be the case for sentences that encode multiple events with an adverbial clause such as the following.

	Training	Development	Test
Sum-of-words	82.6	76.1	75.2
LSTM-RNN	86.6	78.6	**78.5**
LSTM-RNN w. dep. labels	82.9	**79.4**	**78.5**
100D LSTM-RNN [4]	84.8	-	77.6
1024D GRU [21]	98.8	-	81.4
300D SPINN [5]	89.2	-	83.2

Table 1: Prediction accuracies of my models and previously reported results on the training, development and test data. The numbers in bold are the best results among my models.

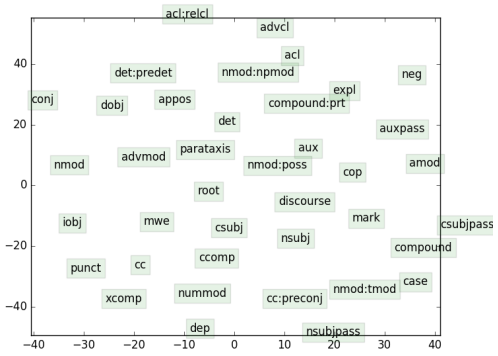

Figure 3: Visualization of the dependency label embeddings.

Premise: A shirtless man is singing into a microphone while a woman next to him plays an accordion.
Hypothesis: A man is singing into a microphone.
Label: Entailment

For this example, the *deplabel* model correctly identified that the main clause entails the hypothesis while the *basic* model incorrectly predicted that these two sentences contradict each other, presumably because the adverbial clause contradicts the hypothesis.

Dependency label embeddings In order to assess, whether the *deplabel* model learned meaningful embeddings for dependency labels, I projected the 10-dimensional dependency label vectors into the 2-dimensional space using t-SNE [20]. The resulting 2-dimensional vectors are visualized in Figure 3. This figure shows that there aren't any clear clusters but at the same time the labels that are close to each other often have similar syntactic functions in the sentence. For example, nominal subjects (`nsubj`) and clausal subjects (`csubj`) are very close together. Likewise, relations denoting clausal modifiers such as adnominal modifiers (`acl`) and adverbial modifiers (`advcl`) are also very close together. However, compared to a similar plot of dependency label embeddings in the context of a neural network-based dependency parser [6], my model does not seem to be able to learn the similarities very well which suggests that the model cannot make good use of the dependency labels.

Comparison to previous results The lower part of Table 1 shows the results of sentence-encoders that were used by other researchers. Compared to the LSTM-RNN by [4], my LSTM-RNN models perform almost 1 percentage points better which suggests that the larger hidden units of the RNN and the resulting bigger number of parameters leads to a better model. However, my relatively simple model cannot compete with the more complex models by [21] and [5]. The 1024-dimensional GRU sentence encoder by [21] contains even more parameters and makes use of pre-trained 'skip-thought' vectors [11] which both seems to further improve performance. This suggests that potentially increasing the dimension of the hidden layer could further improve the performance of my LSTM-RNN. Lastly, the SPINN sentence encoder by [5] performs even better. As this model recursively combines the meaning of phrases along a constituency tress, this suggests that incorporating syntactic information in a more complex way does lead to a better model. Whether and to what extent this is also true for dependency trees still has to be investigated.

5 Related Work

Early work on NLI typically aligned individual words or phrases and then used symbolic reasoning to compute whether the premise entailed the hypothesis (e.g., [12]). More recently, many works investigated how well distributed word and sentence representations can be used for this task and since the recent release of the SNLI corpus, NLI has become a popular testing ground for sentence

representation models. [21] propose a model based on *order-embeddings* and view the task as a partial order completion. They consider a hypothesis to entail a premise if the hypothesis dominates the premise in the predicted partial order. [13] proposed a tree-based convolutional neural network for the task. They use a feature detector to extract features from a dependency tree representation of a sentence and then have a pooling layer to turn these features into a fixed-size sentence representation which they combine by concatenating, taking the difference, and computing the element-wise product before feeding this combined vector into a softmax classifier. [16] use a LSTM with a word-by-word attention model, and [22] propose a modified version of the LSTM model which they call match-LSTM (mLSTM). [7] also propose a variant of an LSTM to predict the relationship between the premise and the hypothesis. Finaly, [5] propose the Stack-augmented Parser-Interpreter Neural Network (SPINN) architecture which parses the sentence into a constituency tree and at the same time also recursively builds up a sentence vector.

6 Conclusion and Future Work

For this project, I implemented three different sentence encoding models and evaluated them in the SNLI task. I particularly investigated whether a simple incorporation of dependency labels has an effect on the model's performance. While I did observe a positive effect on the performance on the delvelopment set, this did not carry over to the test set which suggests that this is not a stable effect and mainly caused by random factors. This assumption is further supported by a qualitative analysis of the output of the two RNN models which reveals almost no systematic improvements and also a qualitative analysis of the dependency label embeddings do not suggest that the model learned very meaningful representations.

However, despite the limited success, I don't think that these results suggest that the incorporation of syntactic information in the form of dependency trees is not helpful as I chose a very simple method to incorporate this information. An obvious next step would be to adapt the SPINN model to dependency trees and to use a composition function similar to the ones by [19] and [18]. Such a model could potentially make better use of the syntactic information and therefore lead to more stable improvements.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Gabor Angeli and Christopher D. Manning. Naturalli: Natural logic inference for common sense reasoning. In *EMNLP*, 2014.

[3] Samuel R. Bowman. Can recursive neural tensor networks learn logical reasoning? *arXiv preprint*, 2013.

[4] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,Lisbon, Portugal, 17-21 September 2015*, pages 632–642, 2015.

[5] Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. A Fast Unified Model for Parsing and Sentence Understanding. *arXiv preprint*, 2016.

[6] Danqi Chen and Christopher D 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.

[7] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long Short-Term Memory-Networks for Machine Reading. *arXiv preprint*, 2016.

[8] Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal Stanford Dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, 2014.

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

[10] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[11] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284, 2015.

[12] Bill MacCartney and Christopher D Manning. Natural logic and natural language inference. *In Harry Bunt, Johan Bos, and Stephen Pulman (eds.), Computing Meaning, volume 4*, pages 129–147, 2014.

[13] Lili Mou, Men Rui, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Recognizing Entailment and Contradiction by Tree-based Convolution. *arXiv preprint*, 2015.

[14] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, 2016.

[15] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[16] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. Reasoning about Entailment with Neural Attention. *arXiv preprint*, 2015.

[17] Sebastian Schuster and Christopher D. Manning. Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2016.

[18] Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the Association for Computational Linguistics*, 2(April):207–218, 2014.

[19] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 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*, pages 1556–1566, 2015.

[20] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[21] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-Embeddings of Images and Language. *arXiv preprint*, 2015.

[22] Shuohang Wang and Jing Jiang. Learning Natural Language Inference with LSTM. *NAACL*, 2016.