"Unmatched" Attention for Natural Language Inference

Vinson Luo
Department of Computer Science
Stanford University
Stanford, CA 94305
vluo@stanford.edu

Homero Roman
Department of Computer Science
Stanford University
Stanford, CA 94305
homero@stanford.edu

Alex Tamkin
Department of Computer Science
Stanford University
Stanford, CA 94305
atamkin@stanford.edu

Abstract

Recognizing natural language entailment and contradiction is a key ability for a human-like artificial intelligence to demonstrate semantic understanding. The recent creation of a large dataset for natural language inference has allowed for a surge of deep learning models to emerge, with recent state-of-the-art models finding success by incorporating robust attention mechanisms. In this paper, we implement a near-state-of-the-art model and explore the effects of several changes to its attention mechanism. While these changes, which we call matrix-based attention weighting and "unmatched" attention, do not significantly improve upon the accuracy of this base model, they underscore the range of possibilities for improved attention models in the future.

1 Introduction

An artificial intelligence system seeking to understand and reason about natural language must to some degree be able to infer the relationships between different natural language statements. The task of Natural Language Inference (NLI) attempts to capture this need; for a given pair of sentences, an NLI system must determine whether one sentence entails or contradicts the other.

More formally, given a premise sentence p and a hypothesis h, an NLI system attempts to predict whether the relationship of the hypothesis to the premise is that of "entailment", "contradiction", or "neutral" (denoting the lack of relation). We demonstrate this task using an example:

- p: A girl is playing violin along with a group of people.
- h_1: A girl is washing a load of laundry.
- h_2: A girl is playing an instrument.
- h_3: A group of people are playing in a symphony.

The pair (p, h_1) constitutes a contradiction because h_1 cannot be true given p (it's impossible to play violin while washing a load of laundry); the pair (p, h_2) constitutes an entailment because h_2 must be true given p (a violin is an instrument); while the pair (p, h_3) should be labeled as neutral because h_3 may or may not be true given p (the group of people could be a symphony, but is not necessarily one).

These labels help draw relationships between sentences. This is important because being able to make inferences is a key test of understanding. Having an accurate way to test for understanding in turn can help improve systems that require semantic analysis of texts. For instance, by helping capture sentence meaning, natural language inference can be used to improve semantic search where the idea is to improve search accuracy by understanding the searcher's intent and the contextual meaning of the input words. Another important application is in document summarization. In order to have a "good" summary the system must be able to ensure that the semantic meaning of the original document is preserved while still capturing a large portion of the document. Accurate and robust natural language inference can in theory help test for a "good" summary by matching portions of the original document to sentences in the summary to show that the document entails the summary and the summary entails most of the document.

2 Related Work

In prior years, the most accurate NLI systems were often composed of elaborate pipelines that relied on many hand-crafted features (for example [1]). Different approaches, such as the use of data-intensive neural architectures, were unsuccessful in large part due to the lack of a sizable dataset for training. Since 2015, however, the availability of the Stanford Natural Language Inference (SNLI) corpus has allowed systems based on deeper neural network architectures to surpass the performance of traditional NLP pipelines.

The SNLI corpus contains around 570k manually written (premise, hypothesis) pairs alongside associated labels, determined through the collective agreement of human labelers [2]. Furthermore, the corpus is partitioned into fixed train, validation, and test sets (with sizes of 550k, 10k, and 10k, respectively), providing a common framework on which to evaluate different NLI systems.

Bowman et al. provided a rudimentary neural network for the SNLI task as well. Their model uses LSTMs [3] to independently process the premise and hypothesis before using the final LSTM hidden states as inputs into a feedforward neural classifier. This simple model achieved an accuracy of 77.6% on the test dataset, far better than previous classifiers of its type.

Rocktschel et al. found that introducing attention mechanisms greatly improved the performance of this basic classifier, achieving 83.5% accuracy on the SNLI test dataset using a model with word-by-word attention [4]. Current state of the art models, such as that of Chen et al.[5], incorporate attention in a multitude of ways to achieve test accuracies of over 88%.

Chen et al. introduce the idea of incorporating syntactic parse information. Ensembling a recursive neural netowrk model with a sequential model, they achieve additional improvement over the sequential model alone. In addition, performance improves even when the parse information is added to an already very strong system.[5] Their sequential model, which we build on in this paper, is as follows: First run the premise and hypothesis through bidirectional Long Short Term Memory networks (BiLSTMs), and collect hidden states $\bar{a}$, $\bar{b}$.

$$\bar{a}_i = BiLSTM_1(a) \forall i \in [1, \ldots, l_a], \tag{1}$$

$$\bar{b}_j = BiLSTM_1(b) \forall j \in [1, \ldots, l_b] \tag{2}$$

Next, they generate attention weights e_{ij} = $a_i^T b_j$ as similarity scores, then run these through a softmax to get e'_{ij}. Next, they create "mimic" vectors for each word in both sentences, which are the weighted sum of the respective attention weights times each word in the other sentence. $\tilde{a}, \tilde{b}$ contain the mimic vectors for sentences a, b respectively:

$$\tilde{a}_i = e_i \bar{b} \tag{3}$$

$$\tilde{b}_i = (e^T)_i \bar{a} \tag{4}$$

Then they create **m** vectors for a and b:

$$m_a = [\bar{a}, \tilde{a}, \bar{a} - \tilde{a}, \bar{a} \circ \tilde{a}] \tag{5}$$

$$m_b = [\bar{b}, \tilde{b}, \bar{b} - \tilde{b}, \bar{b} \circ \tilde{b}] \tag{6}$$

where $\circ$ denotes the Hadamard product. Next, they run the m_a and m_b through a different BiLSTM to get vectors v_1 and v_2:

$$v_{1_i} = BiLSTM_2(m_a) \forall i \in [1, \ldots, l_a] \tag{7}$$

$$v_{2_i} = BiLSTM_2(m_b) \forall i \in [1, \ldots, l_b] \tag{8}$$

Then, they generate the final knowledge representation, v, by max-pooling and average-pooling the v vectors:

$$v = [\text{max\_pool}(v_1), \text{avg\_pool}(v_1), \text{max\_pool}(v_2), \text{avg\_pool}(v_2)] \tag{9}$$

Finally they predict the class by putting v through a simple feedforward classifier using tanh activations. In the paper they also take the additional step of ensembling this model with a tree-RNN model, feeding the root states into the decoder instead of doing the pooling. This results in slightly better results.

3 Approach

We expand upon the model of Chen et al. by testing several extensions to their attention mechanism. First, we generalize the method of producing attention weights by introducing a trainable weight matrix that combines context representations from the premise and hypothesis. Secondly, we experiment with the usage of "unmatchedness" heuristics which attempt to capture whether words in one sentence have strong counterparts in the other.

We add both of these extensions on top of our implementation of the 600D EBIM LSTM classifier described in Chen et al. (detailed in the previous section) [5]. All components other than the attention mechanisms are the same as those in Chen et al., with the possible exception of the final feedforward classifier, which consists of three 300-D tanh activated dense layers in addition to a final softmax layer. Chen et al. does not provide specifics on their feedforward classifier other than the fact that they use tanh activated layers.

3.1 Matrix based attention weighting

The base model of Chen et al. utilizes the following method to produce attention weights e_{ij}, the relative similarity between the ith premise word and the jth hypothesis word:

$$e_{ij} = \bar{\boldsymbol{a}}_i^T \bar{\boldsymbol{b}}_j, \forall i \in [1, \ldots, \ell_a], \forall j \in [1, \ldots, \ell_b] \tag{10}$$

where $\bar{\boldsymbol{a}}_i$ and $\bar{\boldsymbol{b}}_j$ are the context representations of the ith premise word and jth hypothesis word, respectively.

We further generalize this method of producing attention weights by introducing a new trainable parameter $\boldsymbol{W}_a$, incorporating it into our computation of e_{ij} as follows:

$$e_{ij} = \bar{\boldsymbol{a}}_i^T \boldsymbol{W}_a \bar{\boldsymbol{b}}_j \tag{11}$$

This type of attention generalization for this class of matching LSTMs has found success in the similar problem of Neural Machine Translation (see [6]).

3.2 "Unmatchedness" heuristics

Many of the sentence pairs labeled as neutral or contradictory in the SNLI dataset contain words in either the premise or the hypothesis which do not correspond at all to any words in the other sentence. We call these words "unmatched" words, and the presence of unmatched words in either the premise or hypothesis can be used as an additional feature to help determine the relationship between the premise and hypothesis.

For instance, one might expect that a sentence pair of class "entailment" may have unmatched words in the premise if it has any at all, while sentence pairs that are contradictory may contain large

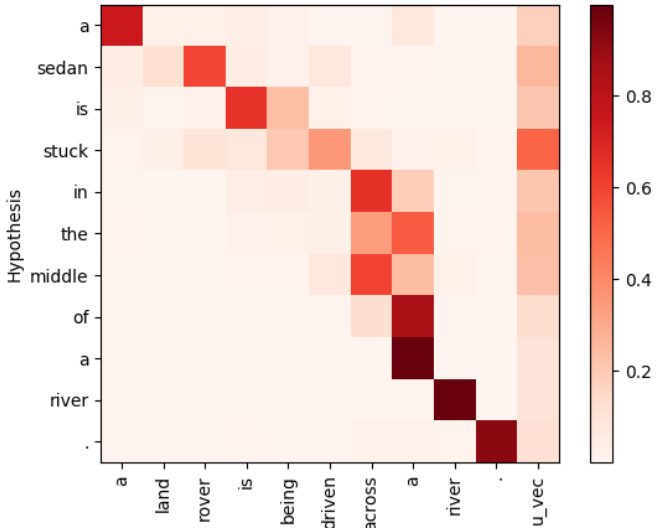

Figure 1: Attention weights and computed $\boldsymbol{u}_b$ vector for a sample hypothesis attending on a premise. The values of the $\boldsymbol{u}_b$ vector are shown in the final column.

amounts of unmatched words in the hypothesis. Neutral sentences, on the other hand, might contain several unmatched words in both the premise and hypothesis.

We derive a metric for finding the "unmatchedness" of words in a sentence from the attention weights found for words in the premise and hypothesis. For word i of the premise we have attention weights on the hypothesis $\boldsymbol{w}_{aij}$ given by:

$$\boldsymbol{w}_{aij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_b} \exp(e_{ik})} \tag{12}$$

Premise words which do not have a strong counterpart in the hypothesis will tend to have uniform $\boldsymbol{w}_{aij}$, while words with a strong counterpart will be much sparser, with a single value close to 1 in $\boldsymbol{w}_{aij}$. Taking this into account we generate an unmatchedness heuristic u_{ai} for each word in the premise based on the relative sparsity of the word's attention weighting on the hypothesis, computed using the ℓ_2 norm:

$$\ell_{2i} = \|w_{aij}\|_2^2 \tag{13}$$

$$u_{ai} = 1/(\ell_b * \ell_{2i}) \tag{14}$$

where the additional factor of ℓ_b in the denominator is designed to normalize the quantity u_{ai} with respect to the length of the hypothesis. Because $\sum_j w_{aij} = 1$, we find that u_{ai} always takes on values in the range $[1/\ell_b, 1]$, with higher values corresponding to more uniform attention weights.

We also compute the corresponding vector $\boldsymbol{u}_b$ for hypothesis attending on the premise.

A visualization of generated values of $\boldsymbol{u}_b$ alongside attention weights generated by one of our models is shown in Figure 1. The word "stuck", which appears in the hypothesis but does not have a strong counterpart in the premise, has a high corresponding value in u_b.

In our first approach for using u_{ai}, we simply append u_{ai} to the m vector used as input to the second set of LSTMs, in essence adding a single dimensional feature to every word indicating its unmatchedness. We call this approach the unmatched "append" approach.

We also test a second approach for using u_{ai}, which we call the unmatched "heuristics" approach. The "heuristics" approach takes into account the identity of unmatched words and formulates a

vector resembling the $\boldsymbol{m}$ vectors in the baseline Chen et al. model. Specifically, we compute:

$$\boldsymbol{m}'_a = [\boldsymbol{u}_a \odot \bar{\boldsymbol{a}}, \boldsymbol{u}_a \odot \bar{\boldsymbol{a}} - \tilde{\boldsymbol{a}}, \boldsymbol{u}_a \odot \bar{\boldsymbol{a}} \odot \tilde{\boldsymbol{a}}] \tag{15}$$

where $\tilde{\boldsymbol{a}}$ is the mimic vector for the hypothesis. We concatenate m_a and m'_a to form a new input on the premise side for our second bidirectional LSTM layer. We apply the same approach to m_b and m'_b.

4 Experiments

4.1 Evaluation metrics

We train our models to minimize the cross-entropy loss between predicted label probabilities and gold labels. However, following the lead of previous work in natural language inference, for evaluation purposes we use the accuracy of the classifier (number of correct predictions / number of test examples) as our final metric. Predictions are made off of the label with the highest predicted probability.

4.2 Training details

We trained all of our models on the train partition of the SNLI dataset, which contains 550k labeled sentence pairs. Out of sample error for an iteration of a model was computed using the 10k validation dataset, and the version of each model with the highest accuracy on the validation dataset was used for final testing.

We initialize our embedding vectors using *300-D GloVe 6B* vectors [7], with out of vocabulary (OOV) words initialized with independent random samples from a Gaussian distribution with standard deviation 1.

For optimization of neural models, we use the Adam optimizer [8] using the parameter values specified in the paper: initial learning rate of 0.0004, beta1 = 0.9, and beta2 = 0.999. We train our models (including embeddings) using a batch size of 512.

Sentences that contained more than 30 words were truncated to 30 words in length. We found that only 1.8% of premises in the training dataset and 2.5% of premises in the test dataset required truncation. Hypotheses rarely if ever required truncation, with 0.02% of the training hypotheses and 0% of the test hypotheses requiring truncation.

4.3 Results

A summary of our results can be seen in Table 1. Ultimately, due to resource constraints for hyperparameter search and some lack of specificity in the Chen et al. paper, our implementation of their model attained a test accuracy of 87.2 relative to their 87.7. We found that neither the addition of matrix based attention weightings nor the "unmatchedness" heuristics resulted in a significant increase in test accuracy over our implementation of Chen's model.

Figure 1 illustrates the use of attention for a pair of sentences with class "contradiction." For each word in the hypothesis (the sentence on the y-axis) the figure shows the corresponding attention weights for each word in the premise (x-axis). The rightmost column of squares labeled **u_vec** shows the u-vector for the attention weights. The high value for the word "stuck" in the hypothesis indicates that it does not have a strong counterpart in the premise.

Figure 2 shows a confusion matrix of predicted labels on the x-axis vs gold labels on the y-axis. From this, one can see that the model makes comparatively few errors where it confuses entailment with contradiction. Indeed, the model makes more than twice as many errors when either the predicted or the gold label is neutral. In particular, the model makes the most mistakes when misclassifying neutral sentence pairs as entailments.

Figure 2: Confusion matrix for the unmatched heuristics model.

Model	Train Accuracy (%)	Test Accuracy (%)
Chen et al. 600D EBIM	92.9	87.7
Our 600D EBIM	90.8	87.2
Matrix	90.2	87.3
Unmatched append	90.4	87.2
Unmatched heuristics	90.2	87.2

Table 1: Results

5 Conclusion

In this paper we explore several different mechanisms for expanding upon an advanced model for natural langauge inference on the SNLI corpus. In particular, we build upon Chen et al.'s EBIM attention model, and explore two potential additions to the model: a matrix-based attention weighting and an "unmatchedness heuristic." While neither addition achieves higher test set accuracy than the base Chen model, they highlight the possibilities for building upon this model and creating more advanced attention-based models for NLI.

For future work, we intend to explore different types of attention-based models, including possibly a word-by-word-like model similar to Rocktaschel et al.'s work as well as different types of unmatched heuristics. In addition, since Chen et al. reported increased performance when using tree-LSTMs, we would like to explore recuisive architectures, as well as exploring the potential gain from ensembling models.

6 References

[1] Alice Lai and Julia Hockenmaier. A denotational and distribution approach to semantics. *SemEval 2014*, 2014.

[2] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[3] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[4] Tim Rocktschel, Edward Grefenstette, Karl Moritz Hermann, Toms Kocisk, , and Phil Blunsom. Reasoning about entailment with neural attention. *CoRR abs/1509.06664*, 2015.

[5] Qian Chen, Xiaodan Zhu, Zhenhua Ling, and Si Wei. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.

[6] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[8] Diederik P. Kingma and Lei Jimmy Ba. Adam: A method for stochastic optimization. *CoRR, abs/1412.6980*, 2014.