Natural Language Inference with Attentive Neural Networks

Varun Vijay
Department of Computer Science
Stanford University
varun3@stanford.edu

Julien Kawawa-Beaudan
Department of Computer Science
Stanford University
julienkb@stanford.edu

Kenny Leung
Department of Computer Science
Stanford University
kenleung@stanford.edu

Abstract

The task of recognizing textual entailment (RTE) is to infer whether a given premise sentence entails, contradicts, or stands neutral to a hypothesis sentence. In this paper, we implement and evaluate a number of state-of-the-art attention based sequence models. We observe that models using a matching-aggregation framework perform extremely well, but repeatedly make mistakes evaluating sentence pairs with ambiguity and irrelevant detail. We also experiment with stacked versions of these models. Finally, we perform an analysis of the Stanford Natural Language Inference (SNLI) dataset and note that we may be approaching an upper bound on model performance.

1 Introduction

Our ability to evaluate the relationship between sentences is essential for tackling a variety of natural language challenges, such as text summarization, information extraction, and machine translation. This challenge is formalized as the natural language inference task of Recognizing Textual Entailment (RTE), which involves classifying the relationship between two sentences as one of entailment, contradiction, or neutrality. For instance, the premise “Garfield is a cat”, naturally entails the statement “Garfield has paws”, contradicts the statement “Garfield is a German Shepherd”, and is neutral to the statement “Garfield enjoys sleeping”.

Systems for RTE traditionally relied on piplines involving heavy feature engineering, natural logic analysis, and specialized modules such as negation detection (e.g. Lai and Hockenmaier, 2014; Jimenez et al., 2014; Zhao et al., 2014; Beltagy et al., 2015). The publication of the Stanford Natural Language Inference (SNLI) dataset by Bowman et al. (2015) made it possible to train neural network models that exceeded the performance of earlier techniques on inference, reproducing the tremendous success that deep learning has achieved in other natural langauge processing (NLP) areas. These models have the benefit of not requiring expensive customization, while being able to capture nuances that are difficult to encode into manually specified rules.

Previous papers have found success using Recurrent Neural Networks and Attention modeling to capture the relationships between sentences. These approaches have improved the baseline performance of 77.6% (Bowman et al. 2015) to the current state-of-the-art for a single model of 87.5% (Wang et al. 2017, Sha et al. 2016). In this paper, we perform an evaluation and comparison of the best performing models, with the goal of understanding the essential architectural elements for this task. We also explore the possibility of developing a progressively finer attention vector by stacking attention layers. Finally, we perform a detailed analysis of our models' performance on particular input sentences to propose further research directions and highlight some limitations of the dataset.

1.1 Data

The Stanford Natural Language Inference (SNLI) corpus contains 570,000 hand-labeled sentence pairs, divided into training, development and test splits. The dataset was collected using a crowd-sourcing system in which participants on Amazon Mechanical Turk were asked to provide descriptions of an image that were true, might be true, and were definitely not true. The framework of describing an image was meant to reduce ambiguities by ensuring that the logical relationship between the sentences could be deduced from the image (Bowman et al. 2015).

2 Models

2.1 Baselines

Our baseline models for the RTE task are the sum of words model and a basic recurrent neural network model. In both baseline models, we represent the premise and hypothesis sentences separately, and then concatenate the two representations. This representation is then passed through a 3-layer neural network with tanh activation. The final prediction is made by applying the softmax function to the output of this neural network. Both baseline implementations are based on the versions introduced by Bowman et al. in the original paper which described the SNLI dataset.

2.1.1 Sum-of-Words

The simplest approach for handling RTE is to represent each premise and hypothesis sentence as the average of its constituent's word embeddings. This is commonly known as the rudimentary bag-of-words model.

2.1.2 Recurrent Neural Network

The key idea of a recurrent neural network is that it iteratively updates the representation of a sentence by tending to the words of the sentence in order. At each time step t, the hidden state h_t is updated at each step by the value of the given word embedding and the hidden layer of the previous time step. The final hidden state of the recurrent neural network represents the embedding of the entire sentence. Both the premise and hypothesis sentences are input into separate recurrent neural networks, and the resulting sentence embeddings are concatenated and classified as described above. We considered adopting a Siamese model, training the same set of weights between both recurrent neural networks, but opted out of this approach in favor of improving more complicated approaches involving LSTMs, as described below.

2.2 Models with attention

One of the main weaknesses of basic RNNs is that these models tend to quickly forget information about previous words. They also suffer from the vanishing gradient problem, in which back-propagation provides little to no feedback on earlier inputs in the sequence. This weakness motivates RNNs with long-short-term memory (LSTM) units, which have been successfully used for many NLP tasks including machine translation and RTE. Each LSTM unit maintains a memory cell, which represents the current state of the model. In addition, each LSTM contains three gates - a forget gate, an output gate, and an update gate - which control the flow of information in and out of the memory cell. If c_t, x_t, and h_t represent cell memory, input, and hidden states, respectively, of the LSTM at time t, the equations governing the LSTM are:

$$\begin{aligned} H &= \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} & o_t &= \sigma(W^o H + b^o) \\ i_t &= \sigma(W^i H + b^i) & c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W^c H + b^c) \\ f_t &= \sigma(W^f H + b^f) & h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{1}$$

2.2.1 LSTM with Attention

One of the first models to use a substantially more complicated model than a basic RNN was the attention model (Rocktaschel et al. 2015). Rocktaschel's model addresses the difficulty that RNNs have in retaining the meaning of words across long sentences. In other words, the final hidden state of an RNN or LSTM on the concatenated premise and hypothesis mostly captures the meaning of the hypothesis. However, by using attention, it is possible to incorporate a weighted version of the word embeddings in the premise when computing later hidden states.

The model introduced by Rocktaschel is defined as follows: $Y = [h_1 \dots h_L] \in \mathbb{R}^{k \times L}$, where $h_1 \dots h_L$ are the hidden states of the RNN encoding the premise, L is the length of the premise, and k is the size of the hidden state. In addition, e_L is defined as a vector of 1s with length L. The weighted combination of the premise hidden states, r, is computed as:

$$M = \tanh(W^Y Y + W^h h_n \otimes e_L)$$

$$\alpha = \text{softmax}(w^T M)$$

$$r = Y \alpha^T$$

Finally, the prediction is made on a transformed version of r and h_N, the final hidden state of the RNN on the hypothesis: $h^* = \tanh(W^P r + W^x h_N)$.

2.2.2 Word-by-Word Attention

One flaw with the basic attention model is that it only captures the overall importance of words in the premise, but not the interaction between individual words in the hypothesis and premise. Rocktaschel et al. improved on this with a word-by-word attention model, which resembles the original attention model except that it calculates the attention weights of the premise for each word in the hypothesis. The attention is calculated by an RNN which takes as input at each cell the previous state, one hidden state of the hypothesis RNN, and all the hidden states of the premise RNN. Using the same notation as above, word-by-word attention is defined as:

$$M_t = \tanh(W^Y Y + (W^h h_t + W^x h_n) \otimes e_L)$$

$$\alpha_t = \text{softmax}(w^T M_t)$$

$$r_t = Y \alpha^T + \tanh(W^t r_{t-1})$$

The final classification is performed by running a one-layer neural network with softmax activation on the last output of the attention RNN and the last output of the hypothesis RNN, $h^* = \tanh(W^P r_L + W^x h_N)$.

2.2.3 mLSTM

Another model that uses attention is the mLSTM model (Wang, Jiang, 2015). This model still uses an RNN to encode the premise and hypothesis, and then uses the hidden states of this RNN as inputs to the mLSTM. Specifically, let x_t be the output of the hidden state of the premise RNN, r_t be the attention-weighted premise vectors calculated for word-by-word attention, and h_{t-1} be the previous hidden state of the mLSTM, at time t. Then, the input to each mLSTM unit is

$$H = \begin{bmatrix} x_t \\ r_t \\ h_{t-1} \end{bmatrix}$$

Besides the format of the input, the update equations for the mLSTM are identical to the update equations for standard LSTMs. Unlike the standard attention model, the prediction does not use the sum of the final RNN state with the attention-weighted premise states; the prediction is made on the final output of the mLSTM, which captures both the premise and hypothesis sentences.

2.3 Matching-Aggregation Models

The state-of-the-art models proposed by Chen et al., 2016, and Wang et al., 2017, extend the basic attention-enhanced sequence model framework by modifying either the attention mechanism or the method of combining the attention results. We refer to these models as matching-aggregation models. Broadly speaking, they consist of the following modules:

- Encoding: a sequence model such as an LSTM or BiLSTM is used to transform the word vectors by adding contextual information.
- Matching: similarities and differences between individual premise and hypothesis words are computed to produce richer context representations.
- Subcomponent generation: optionally, the results of the matching are used to generate a number of simple features such as differences and element-wise products.
- Aggregation: a second sequence model, possibly combined with max or average pooling, is used to aggregate the results of the matching process.
- Prediction: as before, a softmax classifier is used to judge the logical relationship between the sentences.

2.4 Enhanced BiLSTM Model

The matching-aggregation model (Chen et al. 2016) uses a BiLSTM to encode the word vectors of premise and hypothesis into hidden states. The BiLSTM computes these outputs by running two independent LSTMs, one in the forward direction and one in the backward direction. The resulting vectors are then concatenated.
The model performs matching by computing dot products between each word in the premise and each word in the hypothesis. Given premise hidden vectors $a_1, \ldots, a_n$ and hypothesis vectors $b_1, \ldots, b_m$ we compute:

$$e_{ij} = a_i^T b_j$$

The attention weights are used to compute context vectors capturing a view of the premise for each word in the hypothesis and vice versa:

$$\alpha_i = \sum_{j=1}^{m} \frac{exp(e_i j)}{\sum_k^m exp(e_i k)} b_j$$

$$\beta_j = \sum_{i=1}^{m} \frac{exp(e_i j)}{\sum_k^n exp(e_k j)} a_i$$

. The model then uses a subcomponent generation module with the following features:

$$m_a = (a, \alpha, a - \alpha, a \circ \alpha)$$
$$m_b = (b, \beta, b - \beta, b \circ \beta)$$

Finally, in order to aggregate the results, a second BiLSTM is run over the subcomponent vectors. The resulting hidden states are then combined in two different ways, by computing the max and average over the time dimension.

2.5 Multi-Perspective Model

The Bilateral Multi-Perspective Model (Wang et al. 2017) also uses a BiLSTM to encode the premise and hypothesis. The model then applies a multi-perspective cosine distance operation to compute the matchings. Given hidden vectors, a and b, this is computed by:

$$f_m(a, b, W) = (cosine(W_1 \circ a, W_1 \circ b), \ldots, (cosine(W_n \circ a, W_n \circ b)$$

The weight matrix projects the vectors into a number of different hidden spaces, and a cosine distance is computed in each one.
The model also modifies the aggregation step by introducing several different methods of combining results, including full matching, which averages over the vectors involving the final hidden state, and

maxpool matching, which performs a similar operation using a maximum. We simplify the model by omitting the attentive maxpool matching operation, which was observed to yield inconsistent results.

2.6 Implementation Details

In order to fairly evaluate the models, we made some implementation decisions differently than the original papers. For example, while the original papers by Bowman and Rocktaschel used the word2vec models as the word embeddings, we used the GloVe vectors for all the models.

Another decision that some papers disagreed on was how to treat out-of-vocabulary words. Even with the 640 billion word version of the GloVe vectors, there are roughly 4,000 words (out of 37,000 total) in the SNLI dataset which are missing from GloVe. Rocktaschel and several other authors decided to randomly initialize word vectors for these missing words, and train them along with the model. Other authors decided to represent missing words by the average of the word vectors of their neighboring words, and keep those word vectors constant. To have consistent results, our models always randomly initialize the missing word vectors, and train only the missing word representations.

Finally, one observation made by Wang and Jiang is that unrelated sentences should not attend to each other at all. To remedy this, they suggest using adding a NULL word to the beginning of each hypothesis, allowing unrelated words in the premise to abstain from attending to the original words in the hypothesis. We implemented this for all of our attention models.

3 Experiments

We had two main goals for this project. The first was to gain an understanding of the state-of-the-art models for RTE and implement some of these models. The second, naturally, was to improve the existing models and hopefully beat the performance of the current best-performing models.

In order to achieve our first goal, we implemented all of the models described above in Tensorflow. For many of the simpler models, we consistently trained each model for 50 training epochs over the training data. However, due to time constraints, we stopped training on many of the larger models when we noticed more than 3 consecutive epochs without improvement on the development set accuracy. We also experimented with using different dropout rates ([0.05, 0.10, 0.15, 0.20]) and L2 regularization coefficient ($[1E^{-4}, 4E^{-4}, 1E^{-3}]$). Again, due to time constraints, we consistently reused the same parameters which worked best on our small models, a dropout rate of 0.15 and an L2 regularization coefficient of $1E^{-4}$, while training our larger models.

3.1 Novel experiments

In order to achieve our second goal, we also implemented several variations on existing models. These included a prototype feedback model, which ran an entire model twice, augmenting the second execution with the final representation produced by the first, and two stacked models. The latter consisted of two copies of a regular attention model, such as the Enhanced BiLSTM Model or the Bilateral Multi-Perspective Model. The idea behind these models is that the context representations generated by the first attention layer might allow the second attention layer to take into account global information. While we were able to successfully able to train these models to approach the performance of their single-layer counterparts, we could not exceed it; our best accuracy was 85.2% with a stacked BiLSTM attention model. The leads us to believe that the second attention layers simply repeated the feature extraction performed by the first. Further research could explore whether this is due to inherent limitations in the stacked LSTM structure or rather our specific models and hyperparameter choices.

3.2 Quantitative results

Model	Hidden size	Number of parameters	Dev.	Test
Sum of Words	200	181k	0.756	0.755
RNN Encoder	200	342k	0.779	0.772
Attention	200	924k	0.8011	0.7988
Word by word attention	200	1M	0.8017	0.7955
mLSTM	200	924k	0.8042	0.7934
Multi-Perspective Matching	100	2.7M	0.844	0.837
Chen Attention Model	300	7.7M	0.868	**0.859**
Stacked Chen Attention Model	300	13M	0.86	0.852
Chen + MPM Ensemble	300	10.4M	0.8741	**0.8687**

As the table above shows, we were able to achieve performance comparable to those of the best models. In particular, the current best model reported on the SNLI leader board, by Wang et al. 2017, achieves 88.8% accuracy on the test set, and the top ten models achieve between 88.8% and 86.1% accuracy on the test set. In comparison, our best single model achieves 85.9% accuracy, and our ensemble model achieves 86.9% accuracy on the test set.

3.3 Qualitative Results

Although the our models did not beat the state-of-the-art models, we can indirectly demonstrate that they did learn interesting features. For example, the heat maps below show the word-by-word attention on different examples, using the Chen model. The heat maps indicate the model correctly places greater attention between relevant words, such as ‘quiet’ and ‘busy’, which indicate a contradiction.

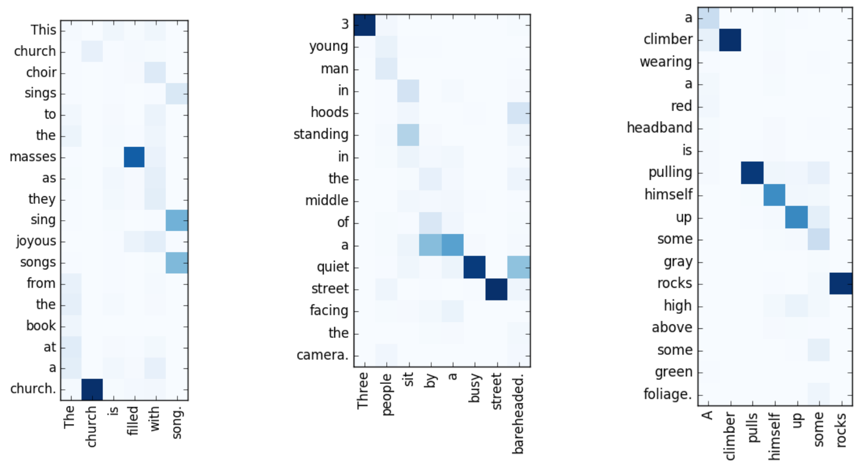
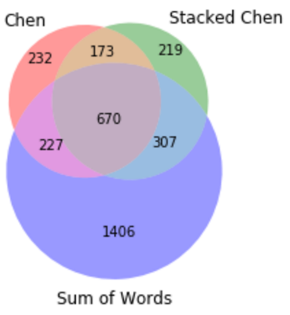


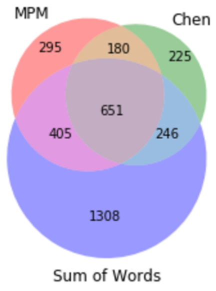
Figure 1: Heat map showing distribution of word-by-word attention on test examples, using the Chen model.

One of the interesting qualitative results from our experiments is that the many of the models made mistakes on the same sentence pairs. The Venn diagrams below show how many of the mistakes made on the test set were shared by different models. Interestingly, a large majority of the mistakes made by our best models were also made by the much simpler sum-of-words baseline. This suggests that there are certain examples which are consistently difficult to classify.

After investigating the examples on which our models consistently made mistakes, we identified several categories of examples to explain these results. One large category of examples includes sentence pairs whose relationship is ambiguous. For example, the premise “25 people are marching” and the premise “a large group of people are marching” have the true label entailment. On this example, our models predict ‘neutral’. Depending on what one considers to be a large group,



(a) Errors made by Chen, Stacked Chen, and SOW models



(b) Errors made by Chen, MPM, and SOW models

Figure 2: Venn diagrams describing the errors made by two sets of different models

this statement could be a contradiction. It is also worth mentioning that in the original SNLI paper, Bowman states that even human annotators only agreed with the gold-standard labels on 89% of the examples. This suggests that since a significant fraction of the sentences have ambiguous relationships, we cannot reasonably expect any system to perform perfectly on this dataset.

Another category of mistakes involves sentences containing irrelevant details. For example, consider the premise “a middle-aged man in a green jacket is talking to another middle-aged man in a light-blue tee-shirt”, and the hypothesis “two old friends catching up with each other”. The true label is entailment, but our model predicts it to be neutral. One possible cause for these types of labels is that the original annotators were asked to describe pictures, and may have used additional information from the image to make their judgment rather than relying only on the premise sentence.

Finally, one last category of mistakes involves sentences with complex grammatical structure. Consider the premise “a soccer player in white kicks a ball as another soccer player lunges towards the ball and a third soccer player in red follows close by,” and the hypothesis “the players are all trying to get the ball from the opposing team.” The correct label in this case is entailment. In order to correctly classify this system, however, the model would have to pay appropriate attention to the three main ideas of the sentence, and then infer that players dressed in different colors are on opposing teams. The models we have described and implemented seem poorly suited for this task. However, this suggests that other models that can leverage the structure of the sentence (for example, using parse trees) may be the best hope for exceeding the current state-of-the-art.

4 Conclusion

In this paper, we implemented several models to handle natural language inference, reproducing state-of-the-art results. While these models represent a considerable improvement over traditional symbolic methods, they make common mistakes as a result of irrelevant detail, syntactic complexity, and ambiguity.
We believe that the first two limitations can be explored through further research into attention-based models. However, the indeterminacy and assumption of contextual knowledge in many sentences may impose an upper bound in the high 80% range to model performance. At the same time, the structure of most sentences is simple, and logical relations can be inferred by simply matching individual words, which is at the heart of all our models. Therefore, we believe that more complex datasets may be needed for further advancement.

Acknowledgments

We thank Ignacio Cases (`cases@stanford.edu`) for his guidance throughout the development of this project.

References

[1] Bowman et al. (2015) A large annotated corpus for learning natural language inference. *arXiv:1508.05326 [cs.cL]*.

[2] Chen et al. (2016) Enhancing and Combining Sequential and Tree LSTM for Natural Language Inference. *arXiv:1609.06038v1 [cs.CL]*

[3] Cheng et al. (2016) Long Short-Term Memory-Networks for Machine Reading. *arXiv:1601.06733v7 [cs.CL]*

[4] Parikh et al. (2016) A Decomposable Attention Model for Natural Language Inference. *arXiv:1606.01933v1 [cs.cL]*

[5] Pennington et al. (2014) GloVe: Global Vectors for Word Representation. *https://nlp.stanford.edu/projects/glove/*

[6] Rocktschel et al. (2016) Reasoning about Entailment with Neural Attention. *arXiv:1509.06664v1 [cs.cL]*.

[7] Wang et al. (2015) Learning natural language inference with LSTM. *arXiv:1512.08849v1 [cs.cL]*.

[8] Wang et al. (2017) Bilateral Multi-Perspective Matching for Natural Language Sentences. *arXiv:1702.03814v2 [cs.AI]*