Natural Language Inference, Sentence representation and Attention Mechanism

Cyprien de Lichy
Stanford University
Stanford, CA

Abstract

A characteristic of natural language is that there are many different ways to express a statement: several meanings can be contained in a single text and the same meaning can be conveyed by different texts. Understanding entailment becomes cruvial to understanding natural language. Natural language inference constitutes an effective way to evaluate machine reading algorithms. It is an interesting problem as it reflects our ability to extract and represent the meaning at a sentence level, and it takes into account several characteristics of natural language such as scope, syntactic structure, lexical ambiguity, semantic compositionality.

1 Introduction and Related work

1.1 The task of natural language inference

Natural Language Inference (or Recognizing Textual Entailment) is the task of determining whether two natural language sentences are in contradiction, not related (neutral), or whether the first sentence (called the premise) entails the second one (called the hypothesis). Thus, it can be seen as a sentence-pair classification task in which the model is fed two sentences (the premise and the hypothesis) and outputs one of the three possible labels. Here are examples of the different classes:

- **ENTAILMENT** : the truth of the premise sentence implies the truth of the hypothesis sentense.
  - **premise** : *Man in a black suit, white shirt and black bowtie playing an instrument with the rest of his symphony surrounding him.*
  - **hypothesis** : *A person in a suit.*
- **CONTRADICTION** : the truth of the premise sentence implies that the hypothesis is false.
  - **premise** : *Man in a black suit, white shirt and black bowtie playing an instrument with the rest of his symphony surrounding him.*
  - **hypothesis** : *Nobody has a suit.*
- **NEUTRAL** : the premise does not help in identifying whether the hypothesis is true or false.
  - **premise** : *Two women having drinks and smoking cigarettes at the bar.*
  - **hypothesis** : *Women are celebrating at a bar.*
  - Here we can't deduce from the premise that the women are celebrating.

Several important NLP applications, such as Question Answering, Information Retrieval, Summarization and Machine Translation evaluation, need to account for this ambiguity inherent to natural language in order to recognize that a particular target meaning can be inferred from different textual variants. In 2004, during the *Workshop on Learning Methods for Text Understanding and Mining* Natural Language Inference was proposed as a generic task that captures major semantic inference needs across many NLP applications.

1.2 Previous work

Several models have been proposed to tackle the Natural Language Inference Task. The recent introduction of the Stanford Natural Language Inference (SNLI) Corpus in 2015 made possible the training of complex models and their benchmark. The introduction paper for the dataset [2] established several baseline: a feature-based classifier using unlexicalized features reaches a test accuracy of 78.2%, an RNN model for sentence representation has a test accuracy of 72.2%, an LSTM encoder reaches 77.6%. Other papers proposed their models for this task: Mou et al. [6] used a Tree-based CNN for sentence representation reaching a test accuracy of 82.1%, Bowman et al. [1] proposed their Stack-augmented Parser-Interpreter Neural Network that combines parsing and representation in a tree-sequence hybrid model and reached a test accuracy of 83.2%. Other models not based on sentence encoding reached higher accuracy: Rocktschel et al. [7] used attention over LSTM states and reached 83.5% test accuracy, Wang and Jiang [8] improved their model to reach 86.1% accuracy, and finally, Cheng et al. [9] used a more sophisticated model called LSTM Network with Deep Attention Fusion and reached the test accuracy of 86.3% which is currently the state-of-the-art for this dataset.

2 Approach

We experimented with 2 different approaches. In the first and simplest approach we try to learn a distributed representation for the sentences (a sentence model) using a Recurrent Neural Network (RNN) and then apply this model to both the premise and the hypothesis and then feed the output to a traditional Multi-Layer Perceptron (MLP) for the classification task. In the second approach we don't try to learn distributed representations for the premise and hypothesis and use instead RNNs with attention before the MLP layer.

2.1 Sentence representation with Long Short-Term Memory

RNNs allow representing sequence of an arbitrary number of inputs in a fixed-size vector while taking into account the structural properties of the input sequence. This architecture allows to model temporal behavior and in NLP, distributed representations can be computed from sequences of word vectors. We used Long short-term memory (LSTM) RNNs because they better handle long term dependencies and are less susceptible to the vanishing gradient problem. LSTMs are more sophisticated RNNs that include gated activation functions (the input, forget and output gates) and they have two vectors at each timestep t: a memory cell c_t and a hidden state vector h_t. The LSTM computes the output h_t and cell state c_t at time step t from an input vector x_t, the previous output h_{t-1} and previous cell state c_{t-1} as:

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

Where the W's and U's matrices are parameters to be trained.

In our first model, sentence representations are learned using an LSTM for the premise and the hypothesis (the same LSTM is used for both). Where we used the final state of the LSTM to encode the sentence. Then the sentence vectors are concatenated and the result is fed in an MLP with 3 hidden layer, and a 3-way softmax layer is used to predict *entailment*, *neutral* or *contradiction*. We used 100D hidden size for the LSTM, and the inputs to the LSTM are 100D word vectors. We used 100D GloVe [10] pre-trained word vectors that are fine-tuned during training.

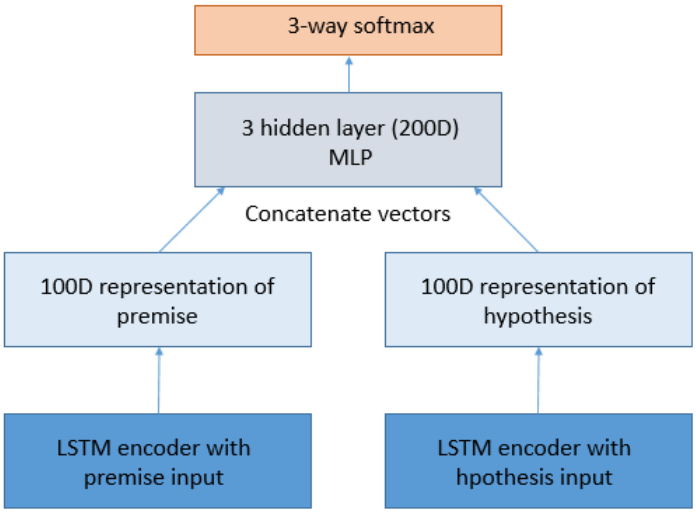

Figure 1: LSTM model

2.2 Attention

The problem with trying to encode a sentence with a single cell state vector is that this vector has to fully capture the meaning of a sentence (which can be very complex). Thus it seems unreasonable to be able to encode all information about a complex sentence into a single vector. Attention mechanisms can be used to alleviate this issue. With attention mechanism, we no longer try to capture all of the semantic meaning of a sentence into a fixed-length vector. Rather, we output vectors from the first LSTM and build a representation in the state cell as we go along the first LSTM to indicate the second LSTM (that of the hypothesis) which of the output it should attend to over to predict the class. Basically, we let the model learn what to attend to based on the two sentences and the class. The attention mechanisme is modeled as follows:

- The matrix $Y \in \mathbb{R}^{k \times L}$ (where k is the size of the hidden state of the LSTMs and L the sequence length) consists of the outputs vectors of the premise LSTM $[h_1, \cdots, h_L]$
- The matrix $H \in \mathbb{R}^{k \times L}$ consists of packing the final output of the hypothesis LSTM L times $[h_{\text{end}}, \cdots, h_{\text{end}}]$
- The attention mechanism will output a vector r that can be seen as a weighted representation of the premise outputs (contained in the Y matrix) $r \in \mathbb{R}^k$
- The attention weights $\alpha \in \mathbb{R}^L$ are computed from a non linear combination of the premise's LSTM outputs and the final output of the hypothesis' LSTM
- The final sentence-pair representation that is fed to the MLP for classification is obtained through a non-linear combination of r and h_{end}

The model equations are the following:

$$
\begin{aligned}
M &= \tanh\left(W^{(y)}Y + W^{(h)}H\right) \\
\alpha &= \text{softmax}\left(w^T M\right) \\
r &= Y\alpha^T \\
h^{(a)} &= \tanh\left(W^{(r)}r + W^{(x)}h_{\text{end}}\right)
\end{aligned}
$$

Where the W's matrices and the vector w are parameters to be trained.

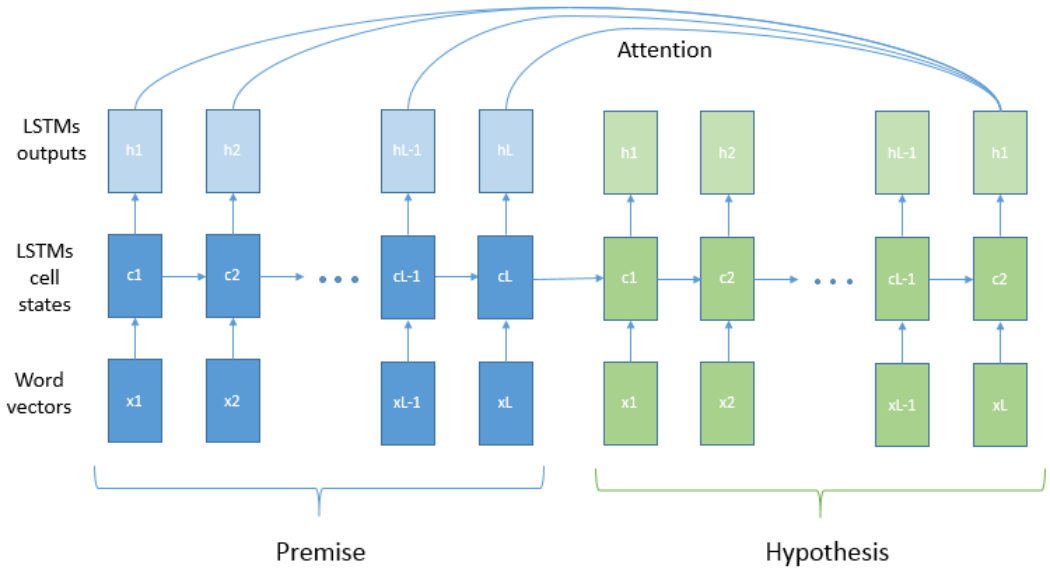

Figure 2: Attention model

3 Experiment

3.1 Dataset

We used the SNLI (Stanford Natural Language Inference) corpus. The SNLI corpus contains 570 152 human-labeled pairs of sentences, the distribution of labels is roughly balanced. Since the dataset is balanced, evaluation can be performed using the simple accuracy metric.

Dataset splits	
Training pairs	549367
Validation pairs	9842
Test pairs	9824

The sentences are rather short for most of them, premises are longer than hypotheses with a mean of 13 words (vs 7 for the hypotheses). The shortest sentence contains 1 word and the longest one contains 78 words. Premise sentences exhibit more variation in sentence length than hypothese. See below for a distribution of the sentence lengths for the premises and hypotheses:

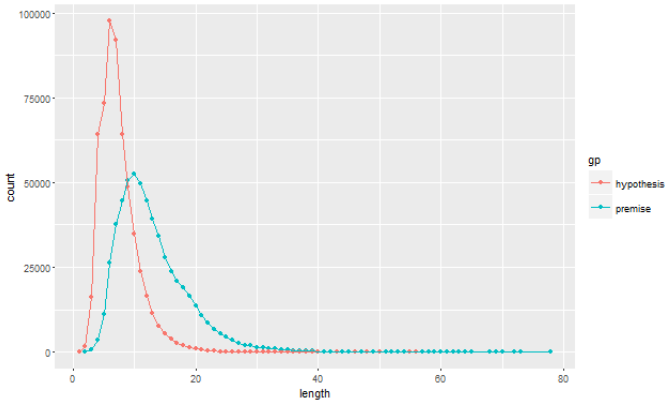

Figure 3: Sentence Length Distribution

3.2 A first baseline

A first experiment was conducted to get a grip at the problem. 100D GloVe word vectors are used for the word present in the training set. The word vectors are averaged to get a rough sentence vector representation for the premise and the hypothesis. Then both sentence vectors are concatenated and fed into a 2-hidden layer neural network with 60 hidden units, then a 3-way softmax classifier is used to output the class probabilities. The network on the whole training set and the performance evaluated on the validation set.

Hyperparameters used	
Batch size	64
Embedding size	100
Hidden size	60
Dropout probability	0.1
L2 Regularization factor	0.0005
Activation function	$\tanh$
Optimizer	Adam
Loss function	cross-entropy
Learning rate	0.001

After 16 epochs, the validation accuracy is 65.2%, and we got the following confusion matrix:

Table 1: Confusion matrix for baseline (2-hidden layer MLP)

Ref. / Pred.	ent	neut	contr
ent	2309	462	558
neut	633	1868	734
contr	560	479	2239

3.3 LSTM Model

Averaging the word vectors is not an accurate way to represent a sentence as it does not take into account position and relative importance of words in the sentence. There we used the LSTM model which first encodes both the premises and the hypothesis sentences using the same LSTM and then concatenate the representations to feed them into a traditional feed-forward neural network. The performance on the training set is improved substantially as expected. See below for a plot of the loss history:

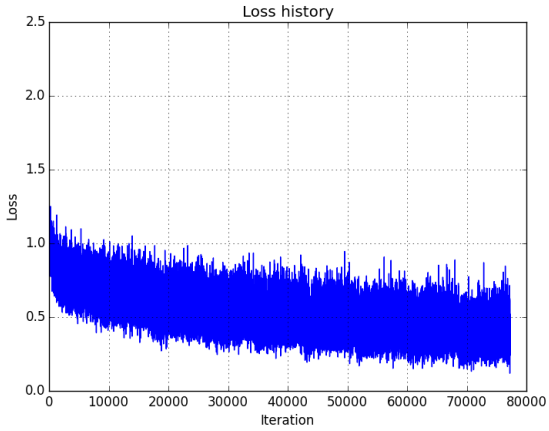

Figure 4: Loss history for the LSTM model

The model was trained using the Adam algorithm [11] for the stochastic gradient optimization with a batch size of 64. Because sentences don't have the same length, we cropped sentences of more

than 30 tokens and padded shorter sentences with a special ¡PAD¿ token. We used a learning rate of 0.001. Gradient clipping was applied to avoid exploding gradients and the threshold was set to 5. To avoid overfitting on the training data, a 20% dropout was used, L2 regularization didn't seem to improve the model and, thus, was not applied. The hidden size of the LSTM was set to 100. And all of the MLP layers sizes were 200. Multi-layers LSTM (with 2 and 3 layers) was tried but did not seem to improve the accuracy. We used pre-trained 100D GloVe word vectors that were fine tuned during training, this did not increase overfitting significantly and did not hurt the model.

We obtained a validation accuracy of 77.2% (training accuracy was 79.4%), see below for the confusion matrix:

Table 2: Confusion matrix for LSTM

Ref. / Pred.	ent	neut	contr
ent	2777	272	261
neut	552	2207	461
contr	310	376	2576

3.4 Attention Model

The problem when using encoding for sentences is that it is hard to capture the full meaning of a sentence into a single vector. That's what we tried implementing an attention-based model that will learn to attend over specific LSTM outputs during training. LSTMs are used not to encode the full sentence but rather to guide the attention model to inform the hypothesis LSTM which premise outputs it needs to attend over to predict the correct class. We used almost the same hyperparameters as before except that now our input to the MLP is 100D dimensional instead of 200D. We used the same learning rate but with exponential decay to help in the optimization process (we chose a learning rate decay parameter of 0.9). Also, instead of having the same LSTM parameters for the premise and the hypothesis we now have two LSTMs (one for the premise and one for the hypothesis). The final state of the premise LSTM becomes the initial state of the hypothesis LSTM. In this model we break the symmetry of the previous model, this seems relevant as the Natural Language Inference task is not symmetric. The performance on the training set is improved. See below for a plot of the training loss history:

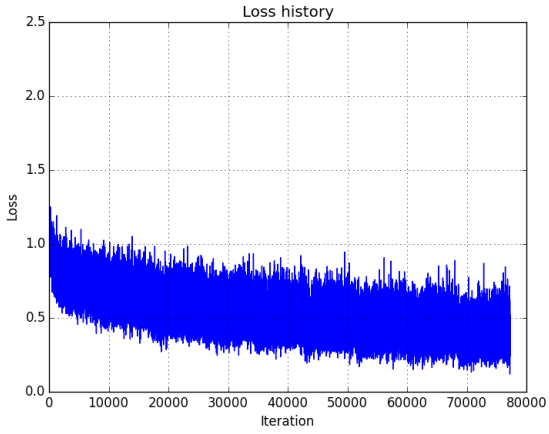

Figure 5: Loss history for the LSTM model

We obtained a validation accuracy of 79.7% (training accuracy was 81.2%), see below for the confusion matrix:

Training took more time and we found out that using annealing during the training process helped quite a bit. We used dropout for regularization and we didn't use L2 regularization (as before). The Adam optimizer was used and we used also gradient clipping to avoid exploding gradients.

Table 3: Confusion matrix for Attention model

Ref. / Pred.	ent	neut	contr
ent	2611	323	376
neut	310	2519	391
contr	330	261	2671

4 Conclusion

Recognising textual entailment is central task in NLP that embeds several characteristics of natural language such as semantic ambiguity, syntactic structure and compositionality. Improvement of Natural Language Inference models will benefit most NLP applications. By developing models that perform well on the task, we can expect to improve our understanding of NLP and contribute to the development of better machine reading algorithms.

Here, we have explored the effectiveness of RNNs and especially the LSTM arhitecture as well as attention mechanisms for the textual entailment task. Although, we could experience the limitations of RNNs for sentence representations as their fixed-size encoding greatly limits the ability of the model to *understand* complex sentences and represent their meaning accurately, as a consequence, we believe that trying to compute sentence embeddings might not be the right approach to the problem, and attention mechanisms seem promising.

References

[1] Bowman *et al* (2016) A Fast Unified Model for Parsing and Sentence Understanding

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

[3] Cheng *et al* (2016) Long Short-Term Memory-Networks for Machine Reading

[4] Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory.

[5] Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets.

[6] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, Zhi Jin. 2016. Natural Language Inference by Tree-Based Convolution and Heuristic Matching

[7] Tim Rocktschel, Edward Grefenstette, Karl Moritz Hermann, Tom Koisk, Phil Blunsom. 2015. Reasoning about Entailment with Neural Attention

[8] Shuohang Wang, Jing Jiang. 2015. Learning Natural Language Inference with LSTM

[9] Jianpeng Cheng, Li Dong, Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading

[10] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[11] Diederik Kingma, Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization

[12] Sepp Hochreiter, Jrgen Schmidhuber. 1997. Long short-term memory