Enhanced LSTM for Natural Language Inference

Qian Chen
University of Science and Technology of China
cq1231@mail.ustc.edu.cn

Xiaodan Zhu
National Research Council Canada
xiaodan.zhu@nrc-cnrc.gc.ca

Zhenhua Ling
University of Science and Technology of China
zhling@ustc.edu.cn

Si Wei
iFLYTEK Research
siwei@iflytek.com

Hui Jiang
York University
hj@cse.yorku.ca

Diana Inkpen
University of Ottawa
diana@site.uottawa.ca

Abstract

Reasoning and inference are central to human and artificial intelligence. Modeling inference in human language is very challenging. With the availability of large annotated data (Bowman et al., 2015), it has recently become feasible to train neural network based inference models, which have shown to be very effective. In this paper, we present a new state-of-the-art result, achieving the accuracy of 88.6% on the Stanford Natural Language Inference Dataset. Unlike the previous top models that use very complicated network architectures, we first demonstrate that carefully designing sequential inference models based on chain LSTMs can outperform all previous models. Based on this, we further show that by explicitly considering recursive architectures in both local inference modeling and inference composition, we achieve additional improvement. Particularly, incorporating syntactic parsing information contributes to our best result—it further improves the performance even when added to the already very strong model.

1 Introduction

Reasoning and inference are central to both human and artificial intelligence. Modeling inference in human language is notoriously challenging but is a basic problem towards true natural language understanding, as pointed out by MacCartney and Manning (2008), *"a necessary (if not sufficient) condition for true natural language understanding is a mastery of open-domain natural language inference."* The previous work has included extensive research on recognizing textual entailment.

Specifically, natural language inference (NLI) is concerned with determining whether a natural-language hypothesis h can be inferred from a premise p, as depicted in the following example from MacCartney (2009), where the hypothesis is regarded to be entailed from the premise.

p: *Several airlines polled saw costs grow more than expected, even after adjusting for inflation.*

h: *Some of the companies in the poll reported cost increases.*

The most recent years have seen advances in modeling natural language inference. An important contribution is the creation of a much larger annotated dataset, the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). The corpus has 570,000 human-written English sentence pairs manually labeled by multiple human subjects. This makes it feasible to train more complex inference models. Neural network models, which often need relatively large annotated data to estimate their parameters, have shown to achieve the state of the art on SNLI (Bowman et al., 2015, 2016; Munkhdalai and Yu, 2016b; Parikh et al., 2016; Sha et al., 2016; Paria et al., 2016).

While some previous top-performing models use rather complicated network architectures to achieve the state-of-the-art results (Munkhdalai and Yu, 2016b), we demonstrate in this paper that enhancing sequential inference models based on chain

models can outperform all previous results, suggesting that the potentials of such sequential inference approaches have not been fully exploited yet. More specifically, we show that our sequential inference model achieves an accuracy of 88.0% on the SNLI benchmark.

Exploring syntax for NLI is very attractive to us. In many problems, syntax and semantics interact closely, including in semantic composition (Partee, 1995), among others. Complicated tasks such as natural language inference could well involve both, which has been discussed in the context of recognizing textual entailment (RTE) (Mehdad et al., 2010; Ferrone and Zanzotto, 2014). In this paper, we are interested in exploring this within the neural network frameworks, with the presence of relatively large training data. We show that by explicitly encoding parsing information with recursive networks in both local inference modeling and inference composition and by incorporating it into our framework, we achieve additional improvement, increasing the performance to a new state of the art with an 88.6% accuracy.

2 Related Work

Early work on natural language inference has been performed on rather small datasets with more conventional methods (refer to MacCartney (2009) for a good literature survey), which includes a large bulk of work on recognizing textual entailment, such as (Dagan et al., 2005; Iftene and Balahur-Dobrescu, 2007), among others. More recently, Bowman et al. (2015) made available the SNLI dataset with 570,000 human annotated sentence pairs. They also experimented with simple classification models as well as simple neural networks that encode the premise and hypothesis independently. Rocktäschel et al. (2015) proposed neural attention-based models for NLI, which captured the attention information. In general, attention based models have been shown to be effective in a wide range of tasks, including machine translation (Bahdanau et al., 2014), speech recognition (Chorowski et al., 2015; Chan et al., 2016), image caption (Xu et al., 2015), and text summarization (Rush et al., 2015; Chen et al., 2016), among others. For NLI, the idea allows neural models to pay attention to specific areas of the sentences.

A variety of more advanced networks have been developed since then (Bowman et al., 2016; Vendrov et al., 2015; Mou et al., 2016; Liu et al., 2016; Munkhdalai and Yu, 2016a; Rocktäschel et al., 2015; Wang and Jiang, 2016; Cheng et al., 2016; Parikh et al., 2016; Munkhdalai and Yu, 2016b; Sha et al., 2016; Paria et al., 2016). Among them, more relevant to ours are the approaches proposed by Parikh et al. (2016) and Munkhdalai and Yu (2016b), which are among the best performing models.

Parikh et al. (2016) propose a relatively simple but very effective decomposable model. The model decomposes the NLI problem into subproblems that can be solved separately. On the other hand, Munkhdalai and Yu (2016b) propose much more complicated networks that consider sequential LSTM-based encoding, recursive networks, and complicated combinations of attention models, which provide about 0.5% gain over the results reported by Parikh et al. (2016).

It is, however, not very clear if the potential of the sequential inference networks has been well exploited for NLI. In this paper, we first revisit this problem and show that enhancing sequential inference models based on chain networks can actually outperform all previous results. We further show that explicitly considering recursive architectures to encode syntactic parsing information for NLI could further improve the performance.

3 Hybrid Neural Inference Models

We present here our natural language inference networks which are composed of the following major components: input encoding, local inference modeling, and inference composition. Figure 1 shows a high-level view of the architecture. Vertically, the figure depicts the three major components, and horizontally, the left side of the figure represents our sequential NLI model named ESIM, and the right side represents networks that incorporate syntactic parsing information in tree LSTMs.

In our notation, we have two sentences $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_{\ell_a})$ and $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_{\ell_b})$, where $\mathbf{a}$ is a premise and $\mathbf{b}$ a hypothesis. The $\mathbf{a}_i$ or $\mathbf{b}_j \in \mathbb{R}^l$ is an embedding of l-dimensional vector, which can be initialized with some pre-trained word embeddings and organized with parse trees. The goal is to predict a label y that indicates the logic relationship between $\mathbf{a}$ and $\mathbf{b}$.

3.1 Input Encoding

We employ bidirectional LSTM (BiLSTM) as one of our basic building blocks for NLI. We first use it

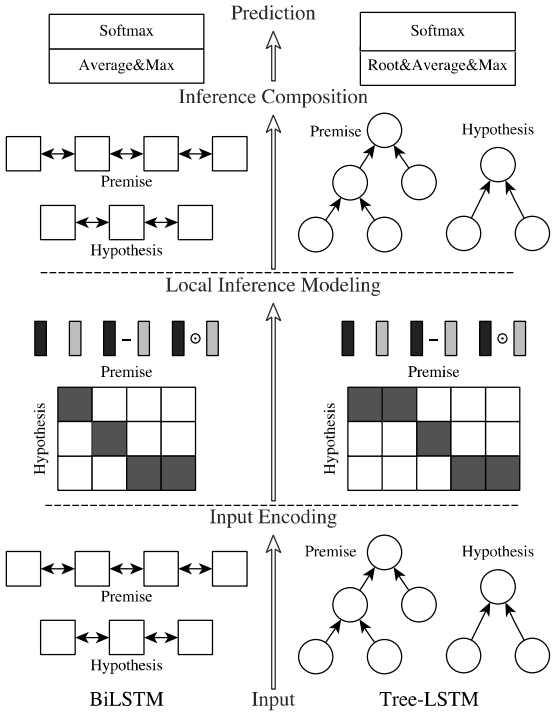

Figure 1: A high-level view of our hybrid neural inference networks.

to encode the input premise and hypothesis (Equation (1) and (2)). Here BiLSTM learns to represent a word (e.g., $\mathbf{a}_i$) and its context. Later we will also use BiLSTM to perform *inference composition* to construct the final prediction, where BiLSTM encodes local inference information and its interaction. To bookkeep the notations for later use, we write as $\bar{\mathbf{a}}_i$ the hidden (output) state generated by the BiLSTM at time i over the input sequence $\mathbf{a}$. The same is applied to $\bar{\mathbf{b}}_j$:

$$\bar{\mathbf{a}}_i = \text{BiLSTM}(\mathbf{a}, i), \forall i \in [1, \ldots, \ell_a], \quad (1)$$

$$\bar{\mathbf{b}}_j = \text{BiLSTM}(\mathbf{b}, j), \forall j \in [1, \ldots, \ell_b]. \quad (2)$$

Due to the space limit, we will skip the description of the basic chain LSTM and readers can refer to Hochreiter and Schmidhuber (1997) for details. Briefly, when modeling a sequence, an LSTM employs a set of soft gates together with a memory cell to control message flows, resulting in an effective modeling of tracking long-distance information/dependencies in a sequence.

A bidirectional LSTM runs a forward and backward LSTM on a sequence starting from the left and the right end, respectively. The hidden states generated by these two LSTMs at each time step are concatenated to represent that time step and its context. Note that we used LSTM memory blocks in our models. We examined other recurrent memory blocks such as GRUs (Gated Recurrent Units) (Cho et al., 2014) and they are inferior to LSTMs on the heldout set for our NLI task.

As discussed above, it is intriguing to explore the effectiveness of syntax for natural language inference; for example, whether it is useful even when incorporated into the best-performing models. To this end, we will also encode syntactic parse trees of a premise and hypothesis through tree-LSTM (Zhu et al., 2015; Tai et al., 2015; Le and Zuidema, 2015), which extends the chain LSTM to a recursive network (Socher et al., 2011).

Specifically, given the parse of a premise or hypothesis, a tree node is deployed with a tree-LSTM memory block depicted as in Figure 2 and computed with Equations (3–10). In short, at each node, an input vector $\mathbf{x}_t$ and the hidden vectors of its two children (the left child $\mathbf{h}^L_{t-1}$ and the right $\mathbf{h}^R_{t-1}$) are taken in as the input to calculate the current node's hidden vector $\mathbf{h}_t$.

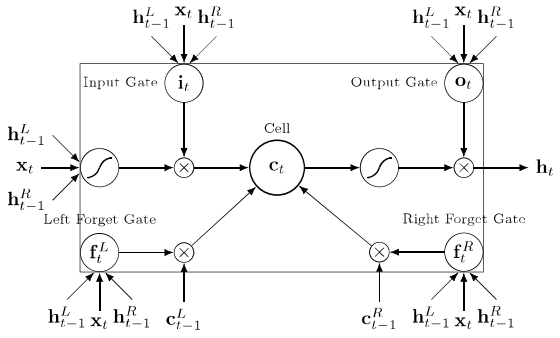

Figure 2: A tree-LSTM memory block.

We describe the updating of a node at a high level with Equation (3) to facilitate references later in the paper, and the detailed computation is described in (4–10). Specifically, the input of a node is used to configure four gates: the input gate $\mathbf{i}_t$, output gate $\mathbf{o}_t$, and the two forget gates $\mathbf{f}^L_t$ and $\mathbf{f}^R_t$. The memory cell $\mathbf{c}_t$ considers each child's cell vector, $\mathbf{c}^L_{t-1}$ and $\mathbf{c}^R_{t-1}$, which are gated by the left forget