Generating Entailments and Contradictions to assist Natural Language Inference

Ryan Sie

1   Introduction

This paper explores the use of deep learning methods for two natural language processing tasks related to Natural Language Inference (NLI). NLI can be described as the general task of determining the particular semantic relationships between multiple sentences - for instance, whether one particular sentence can be inferred from another, or whether two sentences are contradictory. The ability to automate NLI can be very useful for aiding various NLP tasks like information extraction, question answering, and machine translation.

The first task is the generation of entailments and contradictions. Given an input sentence A, we wish to produce a sentence B which A **entails**, a sentence C which **entails** A, and a sentence D which A **contradicts**. Informally speaking, sentence a **entails** sentence b if when a is true, it follows that b is true. Conversely, sentence a **contradicts** sentence b if when a is true, it follows that b is false. If neither of these relationships hold, we say that a and b are **neutral**. A more formal representation of these ideas is discussed in section **4**.

The second task is the classification task of recognizing textual entailment (RTE). Given an input sentence A and hypothesis sentence H, we wish to predict if A **entails**, **contradicts**, or is **neutral to** H. We propose using the results of the generation task to directly assist this task, motivated by knowledge of basic semantic theory.

2   Previous Work

The models for the generation task are standard sequence-to-sequence models [1] which use the LSTM [2] recurrent unit in encoder and decoder recurrent neural networks (RNNs). The use of sequence-to-sequence models has seen great success in a wide range of NLP tasks, and is a natural choice for a generation task like this. Indeed, this particular model architecture has been previously successfully implemented for the same generation task described here by Kolesnyk et al. [3].

The model for the classification task also makes use of recurrent neural networks with the LSTM [2] recurrent unit to produce compact representations, along with an MLP for the actual classification. The base architecture used for our model is inspired by the work of Wang et al. [4], and makes use of two RNNs which separately process the

input and hypothesis sentences. Furthermore, the architecture makes use of word-by-word attention [5]. Our model builds on top of this architecture by utilizing the results of the generation model, as described in the **Model** section.

3 Data

For both tasks, use the Stanford Natural Language Inference (SNLI) corpus [6], which consists of pairs of human-produced input and hypothesis sentences, along with corresponding classification labels of entailment, contradiction, or neutrality between the two sentences. For the generation task, we filter the dataset to the class corresponding to the type of sentence we wish to generate. In particular, for reverse entailment generation, we can switch the order in which the sentences are processed.

To avoid potential inter-model conflicts arising from reuse of data, we randomly split the dataset 50/50 between the two tasks. The result is

We also make use of the GloVe [7] pre-trained 200 dimension word embeddings in each model.

4 Linguistic Motivation

In the linguistic subfield of semantics, the relationships of **entailments** and **contradictions** between sentences in natural language are well studied phenomena. The models discussed in this paper attempt to make use of various parts of semantic theory in their design. This section will provide a brief introduction and overview of the core ideas that motivate the models' architectures.

A common way to formalize the representation of 'declarative' natural language sentences are as sets of possible worlds in which they are true, in the mathematical sense of *set*. A possible world can simply be thought of any particular arrangement of circumstances in the real world that can be imagined. For our purposes it will suffice to describe a possible world w in words. For instance, in one world you might own a goldfish, while in another, you might not. Then a sentence $S = \{w_1, ..., w_k, ...\}$ is a simple set containing possible worlds w_i in which the sentence is true. Note that this set may be infinite (in fact, many are), and even empty. To help illustrate this formulation, below are some example sentences with descriptions of some possible worlds they contain.

(1) I ate breakfast
    a. A world where I ate a bagel for breakfast
    b. A world where I ate cereal for breakfast
(2) The sky is not blue
    a. A world where the sky is purple
    b. A world where the sky is white
(3) The sentence is true or false
    a. All possible worlds
(4) The sentence is both true and false

a. No possible world

Examples (3) and (4) would correspond to the sets W and $\emptyset$, respectively, where W is the set of all possible worlds. As a convention, for a sentence A we will denote S_A as the set of possible worlds in which A is true. With this formulation setup, we can now introduce the following definitions:

Definition 4.1. Let A, B be two sentences. We say that A **entails** B if when A is true, it follows that B is true. This occurs when every possible world in S_A is a possible world in S_B, or $S_A \subseteq S_B$.

Definition 4.2. Let A, B be two sentences. We say that A **contradicts** B if when A is true, it follows that B is false. This occurs precisely when every possible world in S_A is not a possible world in S_B, which is equivalent to saying every possible world in S_B is not a possible world in S_A. These conditions are satisfied precisely when $S_A \cap S_B = \emptyset$.

Definition 4.3. Let A, B be two sentences. We say that A and B are **neutral** if the truth value of one sentence has no bearing on the truth value of the other. This occurs when some but not all possible world(s) in S_A is a (are) possible world(s) in S_B, and vice versa. This is satisfied when $S_A \cap S_B \neq \emptyset, S_A \nsubseteq S_B$, and $S_B \nsubseteq S_A$.

With these definitions in place, we can state some useful facts about these relationships which follow from basic set theory.

Let A, B, and C be sentences.

Fact 4.1. If A entails B and B entails C, then A entails C.

Fact 4.2. If A contradicts B, then B contradicts A.

Fact 4.3. If A contradicts B, and C entails B, then A contradicts C.

5 Model

5.1 Generation Model

The generation task involves 3 sequence-to-sequence models with the same architecture, differing only in the subset of data used to train each one. As mentioned above, the architecture involves an encoder and decoder recurrent neural network (RNN), both of which use the LSTM [2] recurrent unit. We use the GloVe pre-trained word embeddings as well for our embedding layer [7].

An LSTM cell at time t takes in a previous cell state $\mathbf{c}_{t-1}$ and hidden state $\mathbf{h}_{t-1}$, as well as the input at time $t, \mathbf{x}_t$, and produces the next cell state $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$:

$$\mathbf{LSTM}((\mathbf{c}_{t-1}, \mathbf{h}_{t-1}), \mathbf{x}_t) = (\mathbf{c}_t, \mathbf{h}_t)$$

The encoder RNN can then be described by the following algorithm to process an input sentence $S = w_1 w_2 ... w_n$.

$$\text{Initialize } \mathbf{c}_0, \mathbf{h}_0 \tag{1}$$

$$\text{for i} = 1, ..., \text{n} \tag{2}$$

$$\mathbf{x}_i = \text{GloVe}(w_i) \tag{3}$$

$$(\mathbf{c}_i, \mathbf{h}_i) = \mathbf{LSTM}((\mathbf{c}_{i-1}, \mathbf{h}_{i-1}), \mathbf{x}_i) \tag{4}$$

The encoder RNN thus produces n pairs of cell and hidden states $(\mathbf{c}_i, \mathbf{h}_i)$, which will be useful in the decoder RNN. We implement word-by-word attention in the decoder with an attention MLP, in order to capture more intricate relationships between particular lexical items. This MLP can be described as a function $\text{MLP}_{attn} : \mathbb{R}^{d_{enc}+d_{dec}} \to \mathbb{R}$, where d_{enc}, d_{dec} are the hidden state dimensions in the encoder and decoder, respectively (In our case we set them to be equal). This MLP takes in the concatenation of the previous hidden state before decoder timestep j, and some hidden state from the encoder outputs at timestep i, to eventually produce a normalized attention weight α_{j_i} with a final softmax layer.

The following algorithm describes the behavior of the decoder RNN during training, assuming that $T = u_1 u_2 ... u_m$ is the corresponding observed output sentence corresponding to S, and t_j is the index of the nonzero entry of the corresponding one-hot vector of u_j.

$$\text{Take } \mathbf{b}_0 = \mathbf{c}_n, \mathbf{g}_0 = \mathbf{h}_n \tag{5}$$

$$\text{for j} = 1, ..., \text{m} \tag{6}$$

$$\text{for i} = 1, ..., \text{n} \tag{7}$$

$$\bar{\alpha_{j_i}} = \text{MLP}_{attn}(\text{concat}(\mathbf{g}_{j-1}, \mathbf{h}_i)) \tag{8}$$

$$\alpha_{j_i} = \text{softmax}(\bar{\alpha_{j_1}}, ..., \bar{\alpha_{j_n}})_{(i)} \tag{9}$$

$$\mathbf{v}_j = \sum_{i=1}^{n} \alpha_{j_i} \mathbf{h}_i \tag{10}$$

$$\mathbf{x}_j = \text{GloVe}(u_j) \tag{11}$$

$$(\mathbf{b}_j, \mathbf{g}_j) = \mathbf{LSTM}((\mathbf{b}_{j-1}, \mathbf{g}_{j-1}), \text{concat}(\mathbf{x}_j, \mathbf{v}_j)) \tag{12}$$

$$\mathbf{y_j} = \text{MLP}_{pred}(\mathbf{g}_j) \tag{13}$$

$$\text{loss}_j = -\log(\mathbf{y_{j}}_{(t_j)}) \tag{14}$$

$$\text{loss}_{total} = \sum_{j=1}^{m} \text{loss}_j \tag{15}$$

We can then perform

5.2 Inference Model

6 Results

7 Conclusion

References

[1] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS14, page 31043112, Cambridge, MA, USA, 2014. MIT Press.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):17351780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8. 1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

[3] Vladyslav Kolesnyk, Tim Rocktäschel, and Sebastian Riedel. Generating natural language inference chains. *CoRR*, abs/1606.01404, 2016. URL `http://arxiv.org/abs/1606.01404`.

[4] Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. *CoRR*, abs/1512.08849, 2015. URL `http://arxiv.org/abs/1512.08849`.

[5] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664, 2015.

[6] 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 (EMNLP)*. Association for Computational Linguistics, 2015.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.