Text entailment classification

Valle, Damian
s182091@student.dtu.dk

Barkat, Sacha
s182045@student.dtu.dk

November 1, 2018

1 Introduction

Natural Language Inference (NLI) is a main task in the scope of Natural Language Processing that has been around for more than a decade but in the last years has become more relevant due to the new big and high quality datasets. This datasets make it now feasible to train models such as deep neural networks which typically require a large amount of training data. The applications of NLI include question answering, semantic search, automatic text summarization, etc...

The task discussed in this paper is an entailment classification one, which tries to determine whether from a premise sentence P one can infer another hypothesis sentence H.

In order to train the models that will *hopefully* execute the task, we rely on the Stanford Natural Language Inference dataset (SNLI). This corpus contains around 570K sentence pairs with three labels: *entailment, contradiction* and *neutral*.

This dataset has been proven to outperform every other one that came before but it still has some problems like the bias that permited making hypothesis only predictions shown by Adam Poliak et al.[11]. Another problem would be the scenario contradiction that we will illustrate with an example:
Consider the pair of sentences *A boat sank in the Pacific Ocean* and *A boat sank in the Atlantic Ocean*, we might label this pair as a contradiction if we assume that they refer to the same single event, but on the other hand we could also label it as neutral if this assumption is not made.

2 Related Work

In order to implement three different variants of models that can solve the classification challenge we need to look at several papers: Bowman et al.[1] for the first two models (BoW and LSTM RNN) and McCann et al.[3] for the third model (BCN).

1. In his paper [9], Bill MacCartney note that despite its extreme simplicity, a bag of words model achieves surprisingly good results on NLI.

2. Building a model with an LSTM RNN Bowman et al.[1] succeed to get an accuracy goal of 75 percent.Compared to a classic RNN, the paper show that LSTM has a more robust ability to learn long-term dependencies , giving it a substantial advantage over the plain RNN, and resulting in performance that is essentially equivalent to the lexicalized classifier on the test set (LSTM performance near the stopping iteration varies by up to 0.5 percent between evaluation steps). While the lexicalized model fits the training set almost

perfectly, the gap between train and test set accuracy is relatively small for all three neural network models, suggesting that research into significantly higher capacity versions of these models would be productive.

3. McCann et al.[3] managed to achieve state of the art performance on a wide variety of common NLP tasks such as sentiment analysis (SST, IMDb), question classification (TREC) and, most importantly for the purpose of this paper, entailment (SNLI) thanks to it's context vectors (CoVe). Typically in NLP we only initialize the lowest layer of deep models with pretrained word vectors, but McCann et al.[3] propose a deep LSTM encoder from an attentional sequence-to-sequence model to contextualize vectors. In all cases, models that used CoVe from their best pretrained MT-LSTM performed better than baselines that used random word vector initialization.

3 Methods

3.1 Bag of Words (BoW)

In the Bag of Words, the non-linear hidden layer is removed and the projection layer is shared for all words. Thus, all words get projected into the same position (their vectors are averaged). Moreover, we also use words from the future, and the model uses continuous distributed representation of the context.

3.2 LSTM RNN

An LSTM uses a few gate vectors at each position to control the passing of information along the sequence and thus improves the modeling of long-range dependencies We are using an LSTM to perform word-by-word matching of the hypothesis with the premise. To achieve this goal, our LSTM processes the hypothesis, and for each position, it tries to match the current word in the hypothesis with an attention-weighted representationof the premise. Matching results that are critical for the final prediction will be remembered by the LSTM while less important matching results will be "forgotten". q

3.3 Biattentive Classification Network (BCN)

In order to describe the BCN model, first we have to define how the Context Vectors (CoVe) are defined. Let w be a sequence of words and GloVe(w) the corresponding sequence of word vectors produced by the GloVe model, we define

$$\text{CoVe}(w) = \text{MT-LSTM}(\text{GloVe}(w))$$

In our case, our output vector will be the concatenation of the two outputs

$$\tilde{w} = [\text{GloVe}(w); \text{CoVe}(w)]$$

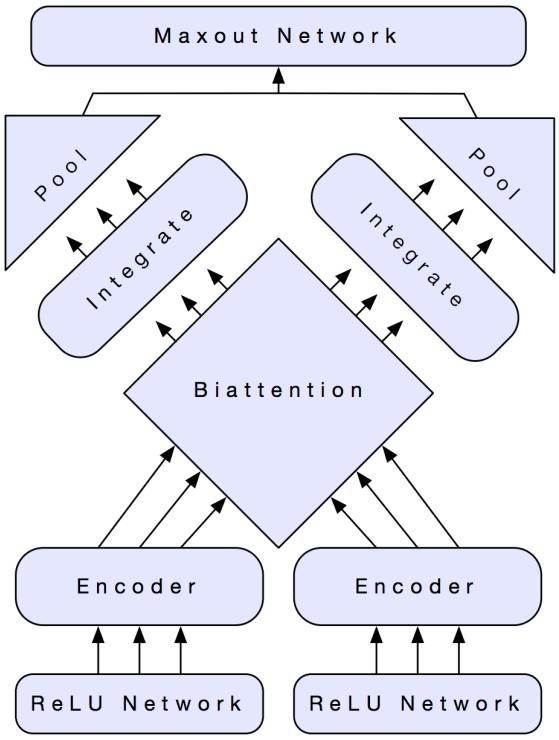
we can use this output to provide context for the BCN.

The model described here is a general Biattentive Classification Network, that means it's designed to handle both single-sentence and two-sentence classification tasks. In the case of the single-sentence task we duplicate it to input both to the model.

Given our input sequences w^x and w^y, we compute their sequences of vectors $\tilde{w}^x$ and $\tilde{w}^y$. We use those as an input to a feedforward neural network modelled by f with ReLU activation to each of them and feed that to a bidirectional LSTM.

$$x = \text{biLSTM}(f(\tilde{w}^x))$$
$$y = \text{biLSTM}(f(\tilde{w}^y))$$

We stack these sequences along the time axix to get matrixes X and Y. To compute interdependent representations we use a biattention mechanism that first computes an



affinity matrix $A = XY^T$, then extracts attention weights with column-wise normalization:

$$A_x = softmax(A)$$
$$A_y = softmax(A^T)$$

Next, it uses context summaries

$$C_x = A_x^T X$$
$$C_y = A_y^T Y$$

to condition each sequence on the other.

We then integrate the conditioning information into our representations for each sequence with two separate one-layer, bidirectional LSTMs that operate on the concatenation of the original representations. The outputs of the bidirectional LSTMs are aggregated by pooling along the time dimension. The pooled representations are combined to get one joined representation for all input and we feed this joined representation through a three-layer maxout network. Ian J. Goodfellow et al.[10].

4 Experimental setup

The code implementation of the models described will be done using python torch implementation (PyTorch) on python notebooks running on Amazon Web Services (AWS) GPUs.

To manage the version control of the code, dataset and papers we use a github repository hosted at: `https://github.com/DamianValle/deep-entailment`

References

[1] Samuel R. Bowman, Gabor Angeli, Christopher Potts and Christopher D. Manning. *A large annotated corpus for learning natural language inference*. Standford Natural Language Processing Group, 2015.

[2] Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, Christopher Potts. *A Fast Unified Model for Parsing and Sentence Understanding*. Standford Linguistics

[3] Bryan McCann, James Bradbury, Caiming Xiong and Richard Socher. *Learned in Translation: Contextualized Word Vectors*. Neural Information Processing Systems, Long Beach, CA, 2017

[4] Shuohang Wang and Jing Jiang. *Learning Natural Language Inference with LSTM*. Information Systems, Singapore Management University 2016

[5] Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. *Efficient estimation of Word Representations in Vector Space*. Google Inc., Mountain View, CA, 2013

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

[7] Chris McCormick. *Word2Vec: The Skip-Gram Model*. `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model`

[8] Sepp Hochreiter and Jurgen Schmidhuber. *Long Short Term Memory*. Technische Universitat Munchen.

[9] MacCartney, Bill. *Natural language inference*. `https://search.proquest.com/docview/305018371`

[10] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville and Yoshua Bengio. *Maxout Networks*. Departement d'Informatique et de Recherche Operationelle, Universite de Montreal

[11] Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. *Hypothesis Only Baselines in Natural Language Inference*.