Semi-supervised sequence tagging with bidirectional language models

Shubham Gupta

March 5, 2020

1 Introduction

- This is the TagLM paper mentioned in Lecture 13 in the CS224N course.
- This paper demonstrates how we can use context embeddings from BiLSTM models and use it for sequence labelling tasks

2 Paper Introduction

- Typically, RNN are used only on labelled data to learn the context embeddings of words.
- Semi supervised approach used in this paper.
  - Train LM on large unlabelled corpus
  - Compute embedding at each position in the sequence(LM embedding)
  - Use embedding in supervised sequence tagging model
- Using both forward and backward embeddings gives better performance than using forward only LM.

3 TagLM

3.1 Overview

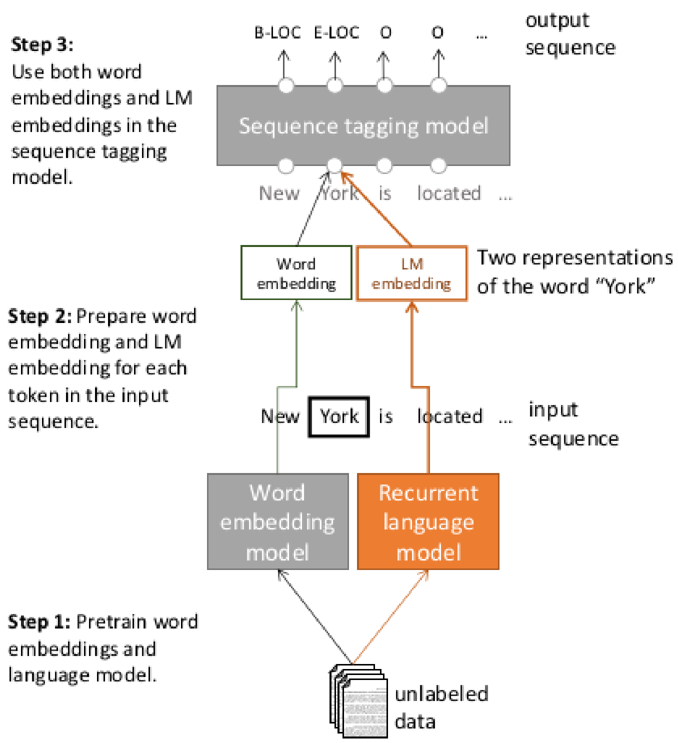

Figure 1: Main componenets in the TagLM model

- Extract word and LM embeddings for every token
- Prepare embedding by concatinating both embeddings
- se them in supervised sequence tagging model

3.2 Baseline

- Baseline model is hierachical neural tagging model
- Obtain word and character embeddings. Concatenate them to form final embedding. $x_k = [c_k; w_k]$
- Char embedding: Can be obtained via CNN or RNN
- Word embedding: Use pretrained embeddings
- Use multiple bidirectional RNN to learn context embedding
- For every token, concatenate forward and backward hidden states at each layer

- 2nd layer will use the above output and predict next hidden state
- Use GRU or LSTM depending on task
- Use output of final layer to predict score for each possible tag using dense layer
- Better to predict tags for full sequence than for a single token
- THEREFORE, add another layer with params for each label bigram
- Compute sentence conditional random field loss(CRF) using forward-backward algorithm
- Use Viterbi algorithm to find most likely sequence

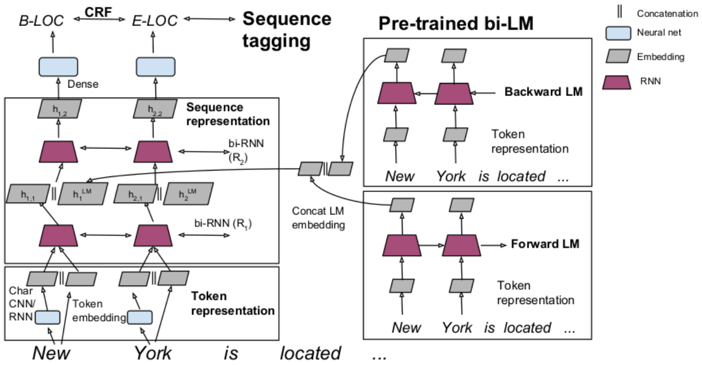

Figure 2: Overview of TagLM, our language model augmented sequence tagging architecture. The top level embeddings from a pre-trained bidirectional LM are inserted in a stacked bidirectional RNN sequence tagging model. See text for details.

Figure 2: Overview of architecture

- LM embedding will be created by concatenating forward and backward embeddings. No param sharing between these two embeddings

4 Experiments

- Evaulation done on CoNLL 2003 NER and CoNLL 2000 chunking
- Lot of detail around model architecture and training methods. Skipping this for now.

4.1 Analysis

- Second RNN captures interactions between task specific context
- Backward LM addition has significant performance gains
- Model size makes a difference. Using bigger CNN model lead to 0.3 percent improvement

- Also tried training the model JUST ON THE CoNLL data. Reduced model size
- Including embeddings from this model **decreased** performance compared to baseline system.
- Replacing task specific RNN with using LM embeddings with a dense layer and CRF **reduced** performance
- Improvement shown by transferring knowledge from other tasks **almost disappers** when the initial model is trained on a large dataset. TLDR: Moar data =¿ Moar performance gain
-