Appunti del paper *Deep Contextualized Word Representations*

A.A. 2023/2024

Rosso Carlo

Contents

1 Introduction

We introduce a new type of *deep contextualized* word representation that models:

- syntax;
- semantics;
- polysemy: how the words uses vary across linguistic contexts.

Our word vectors are learned functions of the internal stated of a deep bidirectiona language model (biLM). We also present an analysis showing that exposign the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision seignals.

Pre-trained word representations are a key compoennt in many neural language understanding models.
Our representations differ from traditional word type embeddings in that each token is assigned a representation that is a function of the entire input sentence. We use vectors derived from a bidirectional LSTM (Long Short-Term Memory) that is trained with a coupled language model. We call them ELMo (Embeddings from Language Models) representations. ELMo representations are deep: we learn a linear combination of the vectors stacked above above each input word for each end task.
We show that higher-level LSTM capture semantic; while lower-level states model capture syntax. It allows the finetuned model over ELMo select the type of information that is most useful for end task.

2 Related Work

Due to their ability to capture syntactic and semantic information of words from large scale unlabeled text, pretrained word vectors are a standard component of most state-of-the-art NLP architectures. However, these approaches for learning word vectors only allow a single context-independent representation for each word. Other recent work has also focused on learning context-dependent representations.
In this paper, we take full advantage of access to plentiful monolingual data, and train our biLM on a corpus with approximately 30 million sentences.
Previous work has also shown that different layers of deep biRNNs encode different types of information. We show that similar signals are also induced by the modified language model objective of our EMLo representations, and it can be very beneficial to learn models for downstream tasks that mix these different types of semi-supervision.
After pretraining the biLM with unlabeled data, we fix the weights and add additional task-specific model capacity, allowing us to leverage large, rich and universal biLM representations for cases where downstream training data size dictates a smaller supervised model (note that it is the case for SST-5 and BERT is used in the same way).

3 Model

ELMo word representations are functions of the entire input sentence. They are computed on top of two-layer biLMs with character convolutions, as a linear function of the internal network states.

3.1 BiLM

Given a sentence of n tokens, $(t_1, \ldots, t_n)$, a forward language model computes the probability of the sequence by modeling the probability of token t_k given the history $(t_1, \ldots, t_{k-1})$:

$$p(t_1, \ldots, t_n) = \prod_{k=1}^{n} p(t_k | t_1, \ldots, t_{k-1}) \tag{3.1}$$

To compute $p(t_k | t_1, \ldots, t_{k-1})$, we first compute a context-independent token representation x_k^{LM} via token embeddings or a CNN over characters, then we pass it through L layers of forward LSTMs. At each position k, each LSTM layer outputs a context-dependent representation $\overleftarrow{h}_{k,j}^{LM}$ where $j \in \{1, \ldots, L\}$. Finally $h_{k,L}^{LM}$ is used to predict the next token t_{k+1} wiht a softmax layer.

A backward LM is similar to a forward LM, except it runs over the sequence in reverse, predicting the previous token given the future context:

$$p(t_1, \ldots, t_n) = \prod_{k=1}^{n} p(t_k | t_{k+1}, \ldots, t_n). \tag{3.2}$$

A biLM combines both a forward and backward LM. Our fomulation jointly maximizes the log likelihood of the forward and backward directions:

$$\sum_{k=1}^{n} \left(\log p\left(t_k | t_1, \ldots, t_{k-1}; \theta_x, \theta^{forward}, \theta_s\right) + \log p\left(t_k | t_{k+1}, \ldots, t_n; \theta_x, \theta^{backward}, \theta_s\right) \right) \tag{3.3}$$

We tie the parameters for both the token representation θ_x and softmax layer θ_s in the forward and backward directions, while maintaining separate parameters for the LSTMs in each direction.

3.2 ELMo

ELMo is a task specific combination of the intermediate layer representations in the biLM. Fro each token t_k, a L-layer biLM computes a set of $2L + 1$ representations:

$$\begin{aligned} R_k &= \{x_k^{LM}, \overrightarrow{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} \mid j \in \{1, \ldots, L\}\} \\ &= \{h_{k,j}^{LM} \mid j \in \{0, \ldots, L\}\} \end{aligned}$$

For inclusion in a downstream model, ELMo collapses all layers in R into a single vector:

$$\text{ELMo}_k^{task} = E(R_k; \theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM} \tag{3.4}$$

where s^{task} are softmax-normalized weights and the scalar parameter γ^{task} allows the task model to scale the entire ELMo vector.

3.3 Using ELMo

Given a pre-trained biLM and a supervised architecture for a target NLP task, it is a simple process to use the biLM to improve the task model. We run the biLM and record all of the layer representations for each word. Then, we let the end task model learn a linear combination of these representations, as descrived below.

First consider the lowest layers of the supervised model without the biLM. Given a sequence of tokens $(t_1, \ldots, t_n)$, it is a standard to form a context-independent token representation x_k for each token position using pre-trained word embeddings.

To add ELMo to the supervised model, we first freeze the weights of the biLM and then concateneate the ELMo vecotr ELMo_k^{task} with x_k and pass the ELMo enhances representation into the task RNN.

Finally, we found it beneficial to add a moderate amount of dropout to ELMo and in some cases to regularize the ELMo weights by adding $\lambda ||w||_2^2$ to the loss.

3.4 Pre-trained biLM architecture

The final model uses $L = 2$ biLMs with 4096 units and 512 dimension projections and a residual connection from the first to second layer. The context insensitive type representation uses 2048 character n-gram convolutional filters followed by two highway layers and a linear projection down to a 512 representation. As a result, the biLM provides three layers of representations for each input token.

Once pretrained, the biLM can compute representations fro any task. In some cases, fine tuning the biLM on domain specific data leads to significant drops in perplexity and an increase in downstream task performance.