ELMo

AI2, 2018

Lu Liu, Peter Schoener, Einar Horn

Motivation

ELMo

- ELMo - Deep Contextualised Word Representations
  - Embeddings from Language Models
- 2018, AllenNLP
- Deep bi-directional LSTM model to create word representations
- Uses a word **and** context to create an embedding

Why not use word embeddings?

- Word embeddings (word2vec/GloVe) are like a dictionary
    - maps string -> vector
- But words can have multiple different meanings
    - **suit** yourself
    - wore a **suit**
- A word’s meaning changes on **context**
- Solution: contextualized word-embeddings


Hey ELMo, what's the embedding of the word "stick"?
There are multiple possible embeddings! Use it in a sentence.
Oh, okay. Here: "Let's stick to improvisation in this skit"
Oh in that case, the embedding is: -0.02, -0.16, 0.12, -0.1 ....etc

Pretrained Model

- Contains a language model trained on the 1B word benchmark (800 million unique tokens)
  - 93.6 million parameters
- Then we fix LM weights, fine-tune additional parameters for a specific NLP task

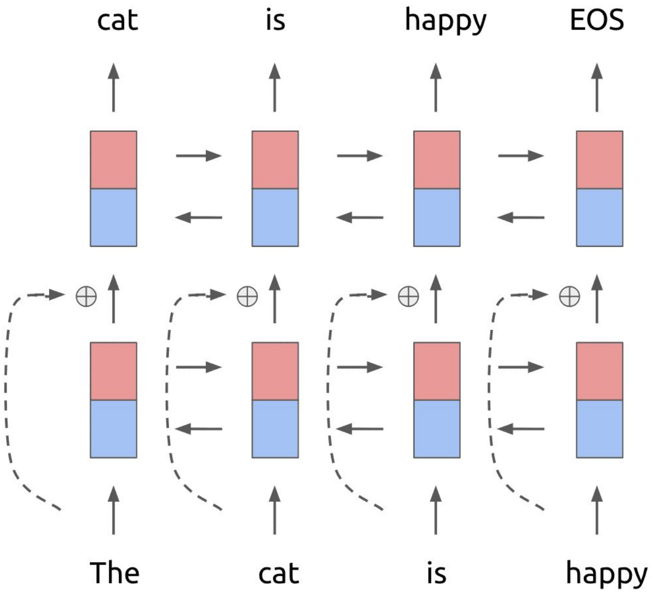
Why pretraining?

- Why separate into two training phases?
- Pretraining is done on a large **unlabelled** corpus
    - General task of language modelling
        - I.e. what's the next word given the previous words?
    - Model gains a fundamental understanding of the language (syntax/semantics)
- Fine-tuning is done on a smaller labelled corpus
    - Teach the model how it should apply it’s understanding of language
        - Should it be focussing on sentiment? Sentence similarity? Etc.

Model-Design

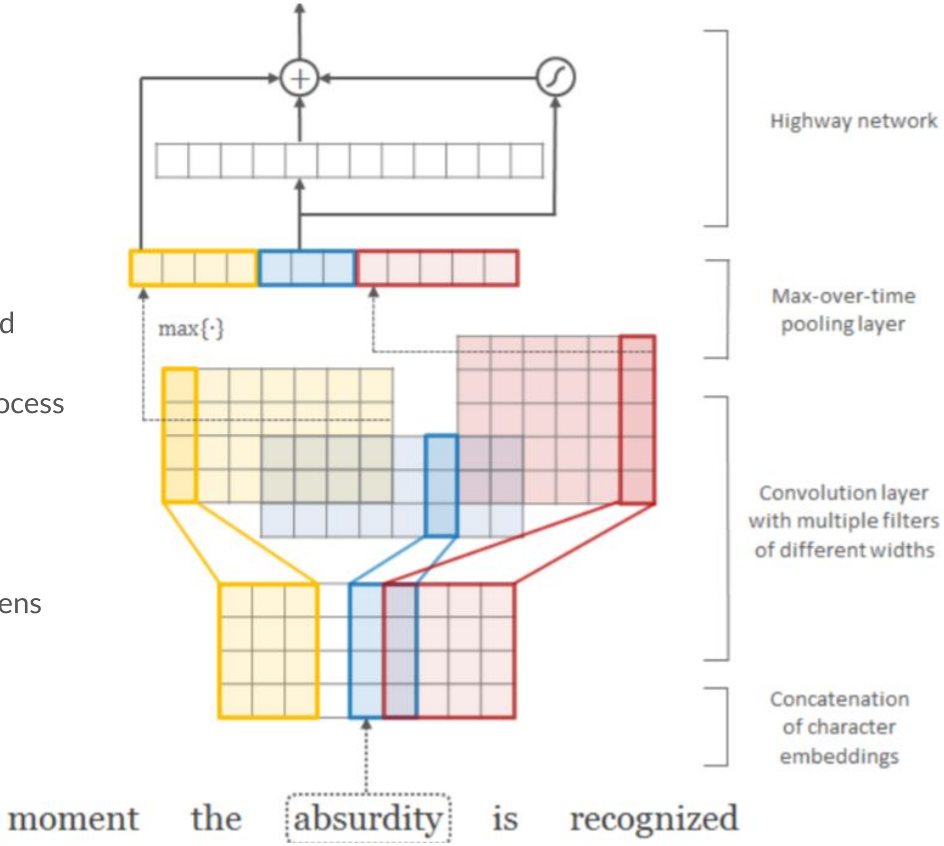
ELMo Architecture

- ELMo architecture contains a 2 layer bi-LSTM language model
  - language model computes the probability of a word, given some prior history + future words
  - Trained on large unsupervised corpus
  - Higher layer learns semantics, lower layer learns syntax

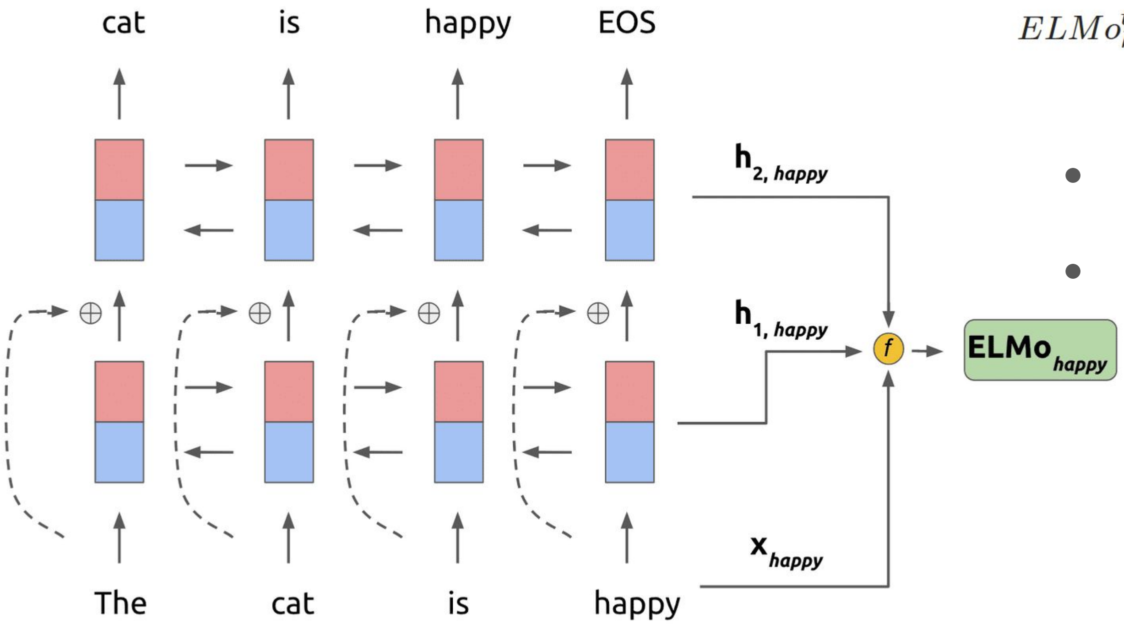


ELMo Input

- Input layer
  - Until now, we've used 1H embeddings or word embeddings
  - ELMo uses a special **character** embedding process
  - Character embeddings
    - Capture morphological information
      - i.e. subword information
      - e.g. **pre-**determine
    - Still creates embeddings for OOV tokens



ELMo Architecture



$$ELMo_k^{task} = \gamma_k \cdot (s_0^{task} \cdot x_k + s_1^{task} \cdot h_{1,k} + s_2^{task} \cdot h_{2,k})$$

- Task specific weights are learned by training on specific task
- Word embedding is a linear combination of each layer’s output

Model-How to learn it

biLM

- Forward LM

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}).$$

- A context-independent token representation $\mathbf{x}_k^{LM}$, then pass it through L layers of forward LSTMs.
- At each position k, each LSTM layer outputs a context-dependent representation

$$\overrightarrow{\mathbf{h}}_{k,j}^{LM} \text{ where } j = 1, \ldots, L$$

- The top layer LSTM output $\overrightarrow{\mathbf{h}}_{k,L}^{LM}$ is used to predict the next token t_{k+1} with a Softmax layer.

biLM

- Backward LM
- Similar to a forward LM, except it runs over the sequence in reverse, predicting the previous token given the future context:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N).$$

- It can be implemented in an analogous way to a forward LM, with each backward LSTM layer j in an L layer deep model producing representations $\overleftarrow{\mathbf{h}}_{k,j}^{LM}$ of t_k given $(t_{k+1}, \ldots, t_N)$

Objectives

- A biLM combines both a forward and backward LM. The ELMo formulation jointly maximizes the log likelihood of the forward and backward directions:

$$\sum_{k=1}^{N} \left(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right)$$

- The parameters of token representatior Θ_x and Softmax layer Θ_s are tied in forward and backward directions while the parameters of LSTM are maintained separate.
- ELMo word representations are computed on top of two-layer biLMs with character convolutions, as a linear function of the internal network states.

ELMo

- For each token t_k, an L-layer biLM computes a set of 2L + 1 representations

$$R_k = \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} = \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},$$

where $\mathbf{h}_{k,0}^{LM}$ is the token layer and $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$, for each biLSTM layer.

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

$$\mathbf{ELMo}_k = E(R_k; \mathbf{\Theta}_e)$$

- In the simplest case, ELMo just selects the top layer $E(R_k) = \mathbf{h}_{k,L}^{LM}$ (e.g. TagLM, CoVe).

ELMo

- More generally, ELMo computes a task specific weighting of all biLM layers:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

- s^{task} are softmax-normalized weights.
- γ^{task} is the scalar parameter allows the task model to scale the entire ELMo vector.
- γ can aid the optimization process, in some cases it also help to apply layer normalization to each biLM layer before weighting considering that the activations of each biLM layer have a different distribution,

Using biLMs for supervised NLP tasks

- First, simply 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.
  - First consider the lowest layers. Given a sequence of tokens $(t_1, \ldots, t_N)$, form a context-independent token representation x_k for each token position (pre-trained word embeddings, optionally character-based representations).
  - Then, the model forms a context-sensitive representation h_k (bidirectional RNNs, CNNs, or feed forward networks).
  - Add ELMo to the supervised model.
    - first freeze the weights of the biLM
    - then concatenate the ELMo vector $\mathbf{ELMo}_k^{task}$ with x_k
    - pass the ELMo enhanced representation $[\mathbf{x}_k; \mathbf{ELMo}_k^{task}]$ into the task RNN.

Usage-Performance

Model testing — results

Task	Previous SOTA		Our baseline	ELMo + baseline
SQuAD	Liu et al. (2017)	84.4	81.1	85.8
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17
SRL	He et al. (2017)	81.7	81.4	84.6
Coref	Lee et al. (2017)	67.2	67.2	70.4
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5

Model testing

- Tested on various standardized semantic tasks
  - question answering
  - entailment
  - semantic role labelling
  - coreference resolution
  - named entity extraction
  - sentiment analysis
- Direct integration in otherwise unmodified state-of-the-art methods was consistently effective
- Effects are small but consistent
- More accurate embedding modelling

Component testing

- Semantic layer is competitive with state-of-the-art for word sense disambiguation
- Syntactic layer is better than semantic layer and CoVe at POS tagging

Model	F_1
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	**70.1**
CoVe, First Layer	59.4
CoVe, Second Layer	*64.7*
biLM, First layer	67.4
biLM, Second layer	*69.0*

Table 5: All-words fine grained WSD F_1. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Model	Acc.
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

Table 6: Test set POS tagging accuracies for PTB. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Discussion

Discussion

- Directly useful drop-in replacement
- Captures semantics very well
- Separates embeddings into meaningful components
- Fast to train even purpose-built models due to pretraining

Reference Paper

- Peters, Matthew E., et al. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).