

ELMo and the Origins of Attention

Contextual embeddings from a language model

Ramaseshan Ramachandran

EMBEDDINGS FROM LANGUAGE MODELS (ELMO)

- ▶ ELMo is a deep contextualized word representation model.
- ▶ Proposed by Peters et al.[\[1\]](#), ELMo captures complex syntactic and semantic characteristics of words.
- ▶ ELMo embeddings are context-sensitive, varying for each instance of a word depending on its context.

ELMO ARCHITECTURE OVERVIEW

- ▶ ELMo uses a bidirectional LSTM language model with two layers to produce context-sensitive embeddings.
- ▶ The model is pretrained on a large text corpus in an unsupervised manner.
- ▶ Key components:
 - ▶ **Character-based Embeddings:** Word embeddings are computed from character-level CNNs.
 - ▶ **Bidirectional LSTMs:** Separate LSTMs for forward and backward contexts.
 - ▶ **Layer-wise Aggregation:** Different layers are combined for task-specific representations.

CHARACTER-BASED EMBEDDINGS

- ▶ Character-based embeddings allow ELMo to handle out-of-vocabulary (OOV) words and morphological variations.
- ▶ A character-level CNN processes each word as a sequence of characters to produce a word embedding.
- ▶ This embedding captures both morphological and orthographic information.

$$\mathbf{w} = \text{CNN}_{\text{char}}(c_1, c_2, \dots, c_n)$$

where $c_1, c_2, \dots, c_n$ are the characters of the word.

BIDIRECTIONAL LANGUAGE MODEL (BiLM)

- ▶ ELMo uses a bidirectional language model (BiLM) consisting of:
 - ▶ **Forward LSTM**: Processes words from left to right.
 - ▶ **Backward LSTM**: Processes words from right to left.
- ▶ Each LSTM layer l produces hidden states for each token t :

$$\vec{h}_t^{(l)} = \text{LSTM}_{\text{forward}}(\mathbf{w}_{1:t})$$

$$\overleftarrow{h}_t^{(l)} = \text{LSTM}_{\text{backward}}(\mathbf{w}_{t:T})$$

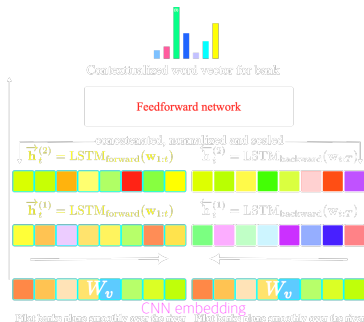
CONTEXTUALIZED ELMO EMBEDDINGS

ELMO embeddings are a weighted combination of hidden states from different layers:

$$\text{ELMO}_t = \gamma \sum_{l=0}^L s^{(l)} \cdot \left[\vec{h}_t^{(l)}; \overleftarrow{h}_t^{(l)} \right]$$

where $s^{(l)}$ are Layer-wise normalized weights using softmax, learned as task-specific parameters, γ is the scaling parameter to adjust the overall magnitude and $\left[\vec{h}_t^{(l)}; \overleftarrow{h}_t^{(l)} \right]$ are the concatenated hidden states from forward and backward LSTMs.

ELMO ILLUSTRATION



$$\text{ELMO}_t = \gamma \sum_{l=0}^L s^{(l)} \cdot [\vec{h}_t^{(l)}; \overleftarrow{h}_t^{(l)}]$$

where $s^{(l)}$ are Layer-wise normalized weights using softmax, learned as task-specific parameters, γ is the scaling parameter to adjust the overall magnitude and $[\vec{h}_t^{(l)}; \overleftarrow{h}_t^{(l)}]$ are the concatenated hidden states from forward and backward LSTMs.

Component	Description
Input	Raw tokens
Layer 1	Input Embedding
Layer 2	Forward
Layer 3	Backward
Output	Contextualized word embedding

Pilot banks plane to smoothly over the river

The forward LSTM processes "bank" in the context of "Pilot," while the backward LSTM processes it in the context of "plane smoothly over the river." This dual context helps disambiguate "bank" as a plane maneuver rather than a financial institution

Reference: Deep Contextualized Word Representations

TRAINING THE ELMO MODEL

- ▶ ELMO is pretrained as a language model on a large text corpus
- ▶ The loss is computed as the sum of the forward and backward language model losses

$$\mathbb{L}(\theta) = \sum_{t=1}^T (-\log p(w_t | w_{1:t-1}) - \log p(w_t | w_{t+1:T}))$$

This includes the parameters of the feed forward weights

- ▶ After pretraining, the model is fine-tuned by learning task-specific weights $s^{(l)}$ for each layer.

Feature Extraction: The pre-trained Bidirectional Language Model (biLM) is used as a fixed function to generate high-quality, context-sensitive word vectors for a given input sentence $S = (t_1, t_2, \dots, t_N)$.

1. Input Processing

- ▶ The input token t_k is first converted to an initial context-independent embedding x_k using a **Character Convolutional Neural Network (CNN)**.

2. Layer-wise Contextual Encoding

- ▶ The embeddings x_k are passed through the $L = 2$ layers of the pre-trained biLM.
- ▶ **Forward Pass ($\overrightarrow{\text{LM}}$):** A forward LSTM computes $\overrightarrow{h}_{k,j}$ (context from $t_1 \dots t_{k-1}$) for each layer $j \in \{1, 2\}$.
- ▶ **Backward Pass ($\overleftarrow{\text{LM}}$):** A backward LSTM computes $\overleftarrow{h}_{k,j}$ (context from $t_{k+1} \dots t_N$) for each layer $j \in \{1, 2\}$.
- ▶ The total representation at layer j for token t_k is the concatenation of the forward and backward states:

$$R_{k,j} = [\overrightarrow{h}_{k,j}; \overleftarrow{h}_{k,j}]$$

3. Final Contextual Embedding Calculation:

- ▶ ELMo defines a set of $L = 3$ layer representations for each token t_k :

$$R_k = \{R_{k,j} \mid j = 0, \dots, L\}, \quad \text{where } R_{k,0} \equiv x_k \text{ (initial embedding)}$$

- ▶ The final ELMo vector $ELMo_k$ for token t_k is a **task-specific weighted linear combination** of these layer representations:

$$ELMo_k = E(R_k; \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{j=0}^L s_j^{\text{task}} R_{k,j}$$

4. Downstream Integration (Inference):

- ▶ The resulting vector $ELMo_k$ is **concatenated** with the other inputs of the downstream NLP model (e.g., a classifier) and used for the final task prediction.
- ▶ The parameters s_j^{task} (softmax-normalized weights) and γ^{task} (scalar scaling factor) are the **only parameters learned** during the supervised training phase for the specific downstream task; the biLM weights are frozen.

BENEFITS OF ELMo

- ▶ ELMo is a task specific combination of the intermediate layer representations in the biLM[1]
- ▶ ELMo embeddings are context-sensitive, making them highly effective for tasks requiring nuanced word representations
- ▶ They capture complex syntactic and semantic relationships that static embeddings (e.g., Word2Vec, GloVe) cannot
- ▶ ELMo is designed to generalize across multiple NLP tasks, providing task-agnostic pre-trained embeddings

CONCLUSION

- ▶ ELMo represents a significant advancement in word embeddings by introducing context-dependent representations
- ▶ Its architecture, based on character-level embeddings and bidirectional LSTMs, provides a flexible and powerful approach for a range of NLP tasks
- ▶ ELMo set the stage for further innovations in contextual embeddings, leading to models like BERT and GPT

1. ELMo produces *contextual* word vectors: a word's vector depends on the whole sentence it appears in [1], so $w_i = f(w_1, w_2, w_3, \dots, w_T)$

$$\text{FORWARD: } p(w_1, w_2, \dots, w_T) = \prod_k^T p(w_k | w_1, w_2, \dots, w_{t-1})$$

$$\text{BACKWARD: } p(w_1, w_2, \dots, w_T) = \prod_k^T p(w_k | w_{k+1}, w_{k+2}, \dots, w_T)$$

2. If there are L layers in both forward and backward LSTMs, then the after L layers, $\vec{h}_{k,L}$ and $\overleftarrow{h}_{k,L}$ are computed
3. $\vec{h}_{k,L}$ and $\overleftarrow{h}_{k,L}$ are scaled (may use a layer norm)
4. $\vec{h}_{k,L}$ and $\overleftarrow{h}_{k,L}$ are concatenated for downstream applications
5. For each token, ELMo takes a task-specific weighted (linear) combination of its representations across the L layers
6. ELMo uses a character-based input layer, so there is no need for an **unknown-token** representation
7. Contrast: ELMo uses stacked biLSTMs; BERT and later models replaced these

WHAT IS LAYER NORMALIZATION?

- ▶ Layer Normalization (LayerNorm) is a technique used to normalize the inputs across the features for each training example.
 - ▶ It helps in stabilizing the learning process in deep neural networks.
-

Given a vector $x = [x_1, x_2, \dots, x_d]$, the layer norm is given by

$$\text{LayerNorm}(x) = \gamma \cdot \frac{x - \mu}{\sigma} + \beta \quad (1)$$

where μ is the mean of x , σ is the standard deviation of x and γ and β are learnable scale and shift parameters, respectively

STEPS OF LAYER NORMALIZATION

1. Compute the mean μ of the input vector x . $\mu = \frac{1}{d} \sum_{i=1}^d x_i$
2. Compute the variance σ^2 of the input vector x . $\sigma^2 = \frac{1}{d} \sum_{i=1}^d (x_i - \mu)^2$
3. Normalize the input by subtracting the mean and dividing by the square root of variance. $\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$ where ϵ is a small constant for numerical stability.
4. Scale and shift the normalized values. $y_i = \gamma \hat{x}_i + \beta$

LAYERNORM VS BATCHNORM

- ▶ **Batch normalization** normalizes each feature across the *batch* dimension. It depends on batch statistics and struggles with variable-length sequences and small batches.
- ▶ **Layer normalization** normalizes across the *features* of a single example. It is independent of batch size, which suits sequence models.
- ▶ This is why transformers use LayerNorm (or RMSNorm) rather than BatchNorm.

REFERENCES I

- [1] Matthew E. Peters et al. “Deep contextualized word representations”. In: *CoRR* abs/1802.05365 (2018). arXiv: [1802.05365](https://arxiv.org/abs/1802.05365). URL: <http://arxiv.org/abs/1802.05365>.