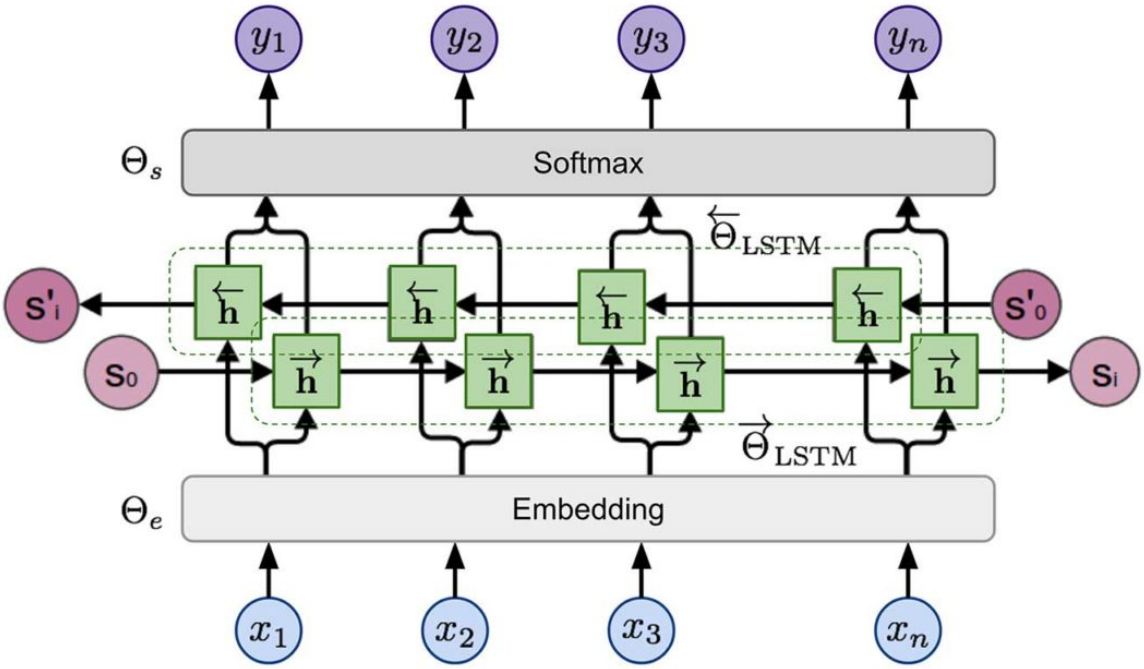
Generalized Language Models: CoVe, ELMo & Cross-View Training

April 24, 2019 by Lilian Weng



EDITOR'S NOTE: Generalized Language Models is an extensive four-part series by Lillian Weng of OpenAI.

- Part 1: CoVe, ELMo & Cross-View Training
- Part 2: ULMFiT & OpenAI GPT
- Part 3: BERT & OpenAI GPT-2
- Part 4: Common Tasks & Datasets

***Do you find this in-depth technical education about language models and NLP applications to be useful?** Subscribe below to be updated when we release new relevant content.*

We have seen amazing progress in NLP in 2018. Large-scale pre-trained language modes like OpenAI GPT and BERT have achieved great performance on a variety of language tasks using generic model architectures. The idea is similar to how ImageNet classification pre-training helps many vision tasks (*). Even better than vision classification pre-training, this simple and powerful approach in NLP does not require labeled data for pre-training, allowing us to experiment with increased training scale, up to our very limit.

(*) Although recently He et al. (2018) found that pre-training might not be necessary for image segmentation task.

In my previous NLP post on word embedding, the introduced embeddings are not context-specific — they are learned based on word concurrency but not sequential context. So in two sentences, "*I am eating an apple*" and "*I have an Apple phone*", two "apple" words refer to very different things but they would still share the same word embedding vector.

Despite this, early adoption of word embeddings in problem-solving is to use them as additional features for an existing task-specific model and in a way the improvement is bounded.

CoVe

CoVe (McCann et al. 2017), short for **Contextual Word Vectors**, is a type of word embeddings learned by an encoder in an attentional seq-to-seq machine translation model. Different from traditional word embeddings introduced here, CoVe word representations are functions of the entire input sentence.

NMT Recap

Here the Neural Machine Translation (NMT) model is composed of a standard, two-layer, bidirectional LSTM encoder and an attentional two-layer unidirectional LSTM decoder. It is pre-trained on the English-German translation task. The encoder learns and optimizes the embedding vectors of English words in order to translate them to German. With the intuition that the encoder should capture high-level semantic and syntactic meanings before transforming words into another language, the encoder output is used to provide contextualized word embeddings for various downstream language tasks.

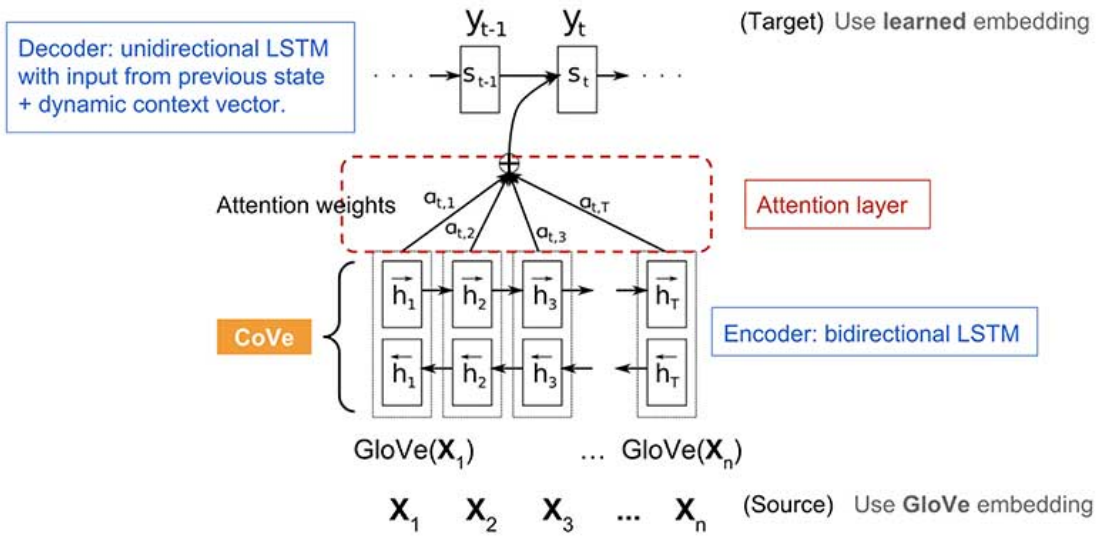


Fig. 1. The NMT base model used in CoVe.

- A sequence of *n* words in source language (English): $x = [x_1, \ldots, x_n]$.
- A sequence of *m* words in target language (German): $y = [y_1, \ldots, y_m]$.
- The GloVe vectors of source words: $\text{GloVe}(x)$.
- Randomly initialized embedding vectors of target words: $z = [z_1, \ldots, z_m]$.

time dimension:

$$\text{decoder hidden state: } s_t = \text{LSTM}([z_{t-1}; \tilde{h}_{t-1}], s_{t-1})$$
$$\text{attention weights: } \alpha_t = \text{softmax}(H(W_1 s_t + b_1))$$
$$\text{context-adjusted hidden state: } \tilde{h}_t = \tanh(W_2[H^\top \alpha_t; s_t] + b_2)$$
$$\text{decoder output: } p(y_t \mid H, y_1, \ldots, y_{t-1}) = \text{softmax}(W_{\text{out}} \tilde{h}_t + b_{\text{out}})$$

Use CoVe in Downstream Tasks

The hidden states of NMT encoder are defined as **context vectors** for other language tasks:

$$\text{CoVe}(x) = \text{biLSTM}(\text{GloVe}(x))$$

The paper proposed to use the concatenation of GloVe and CoVe for question-answering and classification tasks. GloVe learns from the ratios of global word co-occurrences, so it has no sentence context, while CoVe is generated by processing text sequences is able to capture the contextual information.

$$v = [\text{GloVe}(x); \text{CoVe}(x)]$$

Given a downstream task, we first generate the concatenation of GloVe + CoVe vectors of input words and then feed them into the task-specific models as additional features.

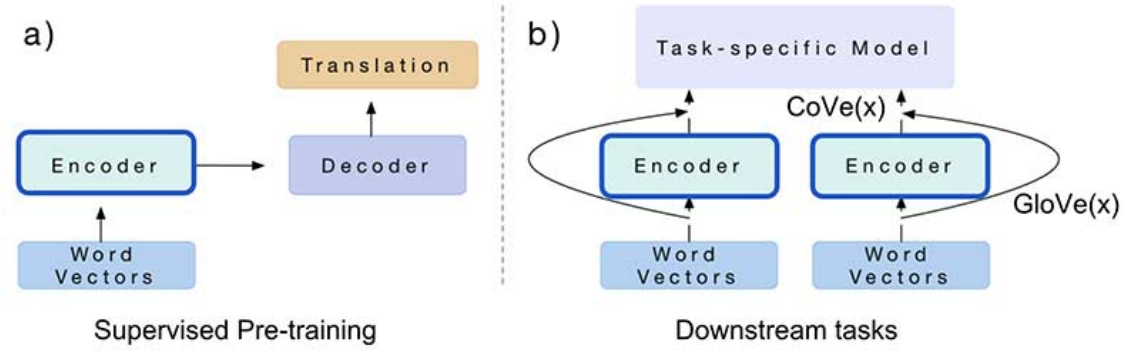


Fig. 2. The CoVe embeddings are generated by an encoder trained for machine translation task. The encoder can be plugged into any downstream task-specific model. (Image source: original paper)

Summary: The limitation of CoVe is obvious: (1) pre-training is bounded by available datasets on the supervised translation task; (2) the contribution of CoVe to the final performance is constrained by the task-specific model architecture.

In the following sections, we will see that ELMo overcomes issue (1) by unsupervised pre-training and OpenAI GPT & BERT further overcome both problems by unsupervised pre-training + using generative model architecture for different downstream tasks.

ELMo

ELMo, short for **Embeddings from Language Model** (Peters, et al, 2018) learns contextualized word representation by pre-training a language model in an *unsupervised* way.

model learns to predict the probability of next token given the history.

In the forward pass, the history contains words before the target token,

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

In the backward pass, the history contains words after the target token,

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_{i+1}, \ldots, x_n)$$

The predictions in both directions are modeled by multi-layer LSTMs with hidden states $\overrightarrow{\mathbf{h}}_{i,\ell}$ and $\overleftarrow{\mathbf{h}}_{i,\ell}$ for input token x_i at the layer level $\ell = 1, \ldots, L$. The final layer's hidden state $\mathbf{h}_{i,L} = [\overrightarrow{\mathbf{h}}_{i,L}; \overleftarrow{\mathbf{h}}_{i,L}]$ is used to output the probabilities over tokens after softmax normalization. They share the embedding layer and the softmax layer, parameterized by Θ_e and Θ_s respectively.

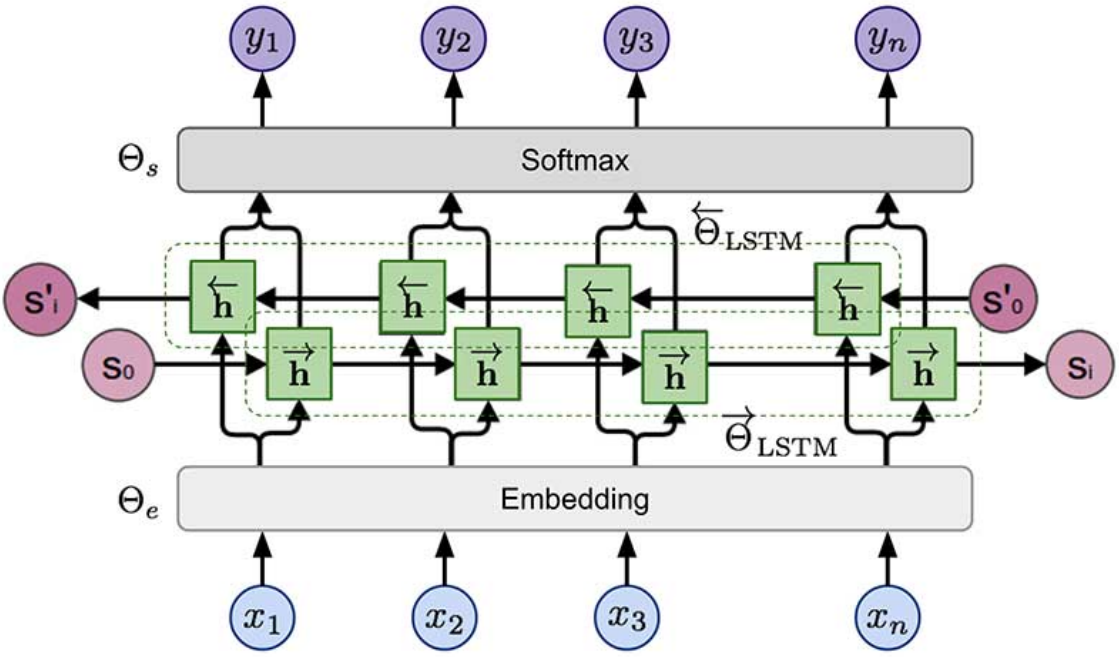


Fig. 3. The biLSTM base model of ELMo. (Image source: recreated based on the figure in "Neural Networks, Types, and Functional Programming" by Christopher Olah.)

The model is trained to minimize the negative log likelihood (= maximize the log likelihood for true words) in both directions:

$$\mathcal{L} = -\sum_{i=1}^{n} \Big(\log p(x_i \mid x_1, \ldots, x_{i-1}; \Theta_e, \overrightarrow{\Theta}_{\text{LSTM}}, \Theta_s) + \log p(x_i \mid x_{i+1}, \ldots, x_n; \Theta_e, \overleftarrow{\Theta}_{\text{LSTM}}, \Theta_s) \Big)$$

ELMo Representations

On top of a L−layer biLM, ELMo stacks all the hidden states across layers together by learning a task-specific linear combination. The hidden state representation for the token x_i contains $2L + 1$ vectors:

$R_i = \{\mathbf{h}_{i,\ell} \mid \ell = 0, \ldots, L\}$ where $\mathbf{h}_{0,\ell}$ is the embedding layer output and $\mathbf{h}_{i,\ell} = [\overrightarrow{\mathbf{h}}_{i,\ell}; \overleftarrow{\mathbf{h}}_{i,\ell}]$.

$$v_i = f(R_i; \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{\ell=0}^{L} s_i^{\text{task}} \mathbf{h}_{i,\ell}$$

To evaluate what kind of information is captured by hidden states across different layers, ELMo is applied on semantic-intensive and syntax-intensive tasks respectively using representations in different layers of biLM:

- **Semantic task**: The *word sense disambiguation (WSD)* task emphasizes the meaning of a word given a context. The biLM top layer is better at this task than the first layer.
- **Syntax task**: The *part-of-speech (POS) tagging* task aims to infer the grammatical role of a word in one sentence. A higher accuracy can be achieved by using the biLM first layer than the top layer.

The comparison study indicates that syntactic information is better represented at lower layers while semantic information is captured by higher layers. Because different layers tend to carry different type of information, *stacking them together helps*.

Use ELMo in Downstream Tasks

Similar to how CoVe can help different downstream tasks, ELMo embedding vectors are included in the input or lower levels of task-specific models. Moreover, for some tasks (i.e., SNLI and SQuAD, but not SRL), adding them into the output level helps too.

The improvements brought up by ELMo are largest for tasks with a small supervised dataset. With ELMo, we can also achieve similar performance with much less labeled data.

Summary: The language model pre-training is unsupervised and theoretically the pre-training can be scaled up as much as possible since the unlabeled text corpora are abundant. However, it still has the dependency on task-customized models and thus the improvement is only incremental, while searching for a good model architecture for every task remains non-trivial.

Cross-View Training

In ELMo the unsupervised pre-training and task-specific learning happen for two independent models in two separate training stages. **Cross-View Training** (abbr. **CVT**; Clark et al., 2018) combines them into one unified semi-supervised learning procedure where the representation of a biLSTM encoder is improved by both supervised learning with labeled data and unsupervised learning with unlabeled data on auxiliary tasks.

Model Architecture

The model consists of a two-layer bidirectional LSTM encoder and a primary prediction module. During training, the model is fed with labeled and unlabeled data batches alternatively.

- On *labeled examples*, all the model parameters are updated by standard supervised learning. The loss is the standard cross entropy.
- On *unlabeled examples*, the primary prediction module still can produce a "soft" target, even though we cannot know exactly how accurate they are. In a couple of auxiliary tasks, the predictor only sees and processes a restricted view of the input, such as only using encoder hidden state representation in one direction. The auxiliary task outputs are expected to match the primary prediction target for a full view of input.
  In this way, the encoder is forced to distill the knowledge of the full context into partial representation. At this stage, the biLSTM encoder is backpropagated but the primary prediction module is *fixed*. The loss is to minimize the distance between auxiliary and primary predictions.

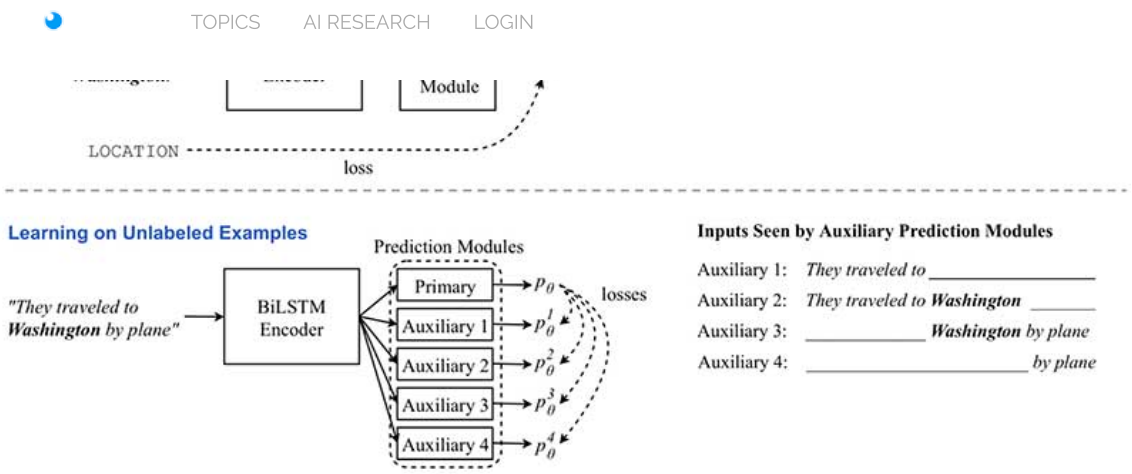


Fig. 4. The overview of semi-supervised language model cross-view training. (Image source: original paper)

Multi-Task Learning

When training for multiple tasks simultaneously, CVT adds several extra primary prediction models for additional tasks. They all share the same sentence representation encoder. During supervised training, once one task is randomly selected, parameters in its corresponding predictor and the representation encoder are updated. With unlabeled data samples, the encoder is optimized jointly across all the tasks by minimizing the differences between auxiliary outputs and primary prediction for every task.

The multi-task learning encourages better generality of representation and in the meantime produces a nice side-product: all-tasks-labeled examples from unlabeled data. They are precious data labels considering that cross-task labels are useful but fairly rare.

Use CVT in Downstream Tasks

Theoretically the primary prediction module can take any form, generic or task-specific design. The examples presented in the CVT paper include both cases.

In sequential tagging tasks (classification for every token) like NER or POS tagging, the predictor module contains two fully connected layers and a softmax layer on the output to produce a probability distribution over class labels. For each token $\mathbf{x}_i$, we take the corresponding hidden states in two layers, $\mathbf{h}_1^{(i)}$ and $\mathbf{h}_2^{(i)}$:

$$
\begin{aligned}
p_\theta(y_i \mid \mathbf{x}_i) &= \text{NN}(\mathbf{h}^{(i)}) \\
&= \text{NN}([\mathbf{h}_1^{(i)}; \mathbf{h}_2^{(i)}]) \\
&= \text{softmax}\big(\mathbf{W} \cdot \text{ReLU}(\mathbf{W}' \cdot [\mathbf{h}_1^{(i)}; \mathbf{h}_2^{(i)}]) + \mathbf{b}\big)
\end{aligned}
$$

The auxiliary tasks are only fed with forward or backward LSTM state in the first layer. Because they only observe partial context, either on the left or right, they have to learn like a language model, trying to predict the next token given the context. The `fwd` and `bwd` auxiliary tasks only take one direction. The `future` and `past` tasks take one step further in forward and backward direction, respectively.

$$
\begin{aligned}
p_\theta^{\text{fwd}}(y_i \mid \mathbf{x}_i) &= \text{NN}^{\text{fwd}}(\overrightarrow{\mathbf{h}}^{(i)}) \\
p_\theta^{\text{bwd}}(y_i \mid \mathbf{x}_i) &= \text{NN}^{\text{bwd}}(\overleftarrow{\mathbf{h}}^{(i)}) \\
p_\theta^{\text{future}}(y_i \mid \mathbf{x}_i) &= \text{NN}^{\text{future}}(\overrightarrow{\mathbf{h}}^{(i-1)}) \\
p_\theta^{\text{past}}(y_i \mid \mathbf{x}_i) &= \text{NN}^{\text{past}}(\overleftarrow{\mathbf{h}}^{(i+1)})
\end{aligned}
$$

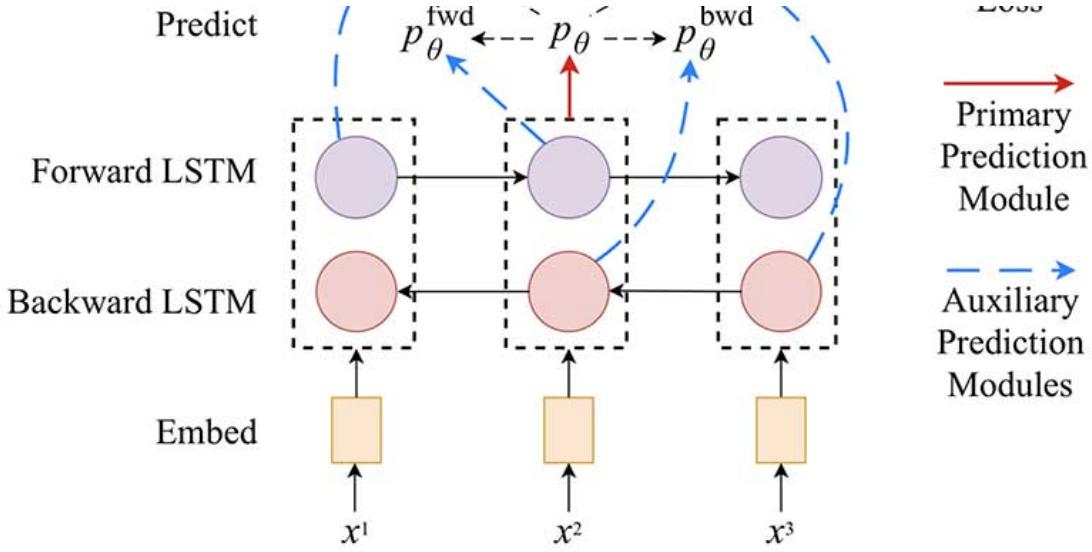


Fig. 5. The sequential tagging task depends on four auxiliary prediction models, their inputs only involving hidden states in one direction: forward, backward, future and past. (Image source: original paper)

Note that if the primary prediction module has dropout, the dropout layer works as usual when training with labeled data, but it is not applied when generating "soft" target for auxiliary tasks during training with unlabeled data.

In the machine translation task, the primary prediction module is replaced with a standard unidirectional LSTM decoder with attention. There are two auxiliary tasks: (1) apply dropout on the attention weight vector by randomly zeroing out some values; (2) predict the future word in the target sequence. The primary prediction for auxiliary tasks to match is the best predicted target sequence produced by running the fixed primary decoder on the input sequence with beam search.

References

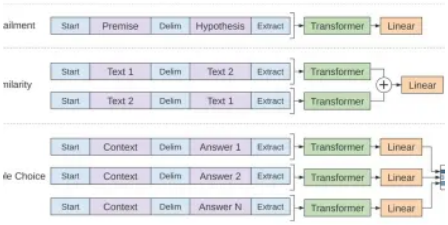
This article was originally published on Lil'Log and re-published to TOPBOTS with permission from the author.

- [ ] Chatbots & Conversational AI
- [ ] Computer Vision
- [ ] Ethics & Safety
- [ ] Robotics
- [ ] Machine Learning
- [ ] Deep Learning
- [ ] Reinforcement Learning
- [ ] Generative Models
- [ ] Other (Please Describe Below)

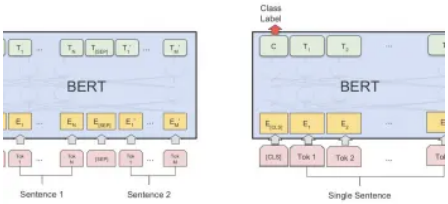
What is your biggest challenge with AI research? *

SUBMIT

Related

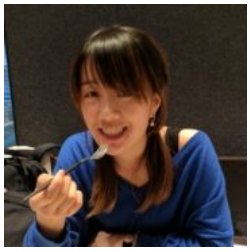


Generalized Language Models: ULMFiT & OpenAI GPT



Generalized Language Models: BERT & OpenAI GPT-2

Generalized Language Models: Common Tasks & Datasets



About Lilian Weng

Lilian Weng is on the Robotics team at OpenAI. She writes code, reads papers, does research on deep learning models, and works on physical machines.

Leave a Reply

Name

Email

Website

POST COMMENT

Search the site ...

Learn Applied AI

We create and source the best content about applied artificial intelligence for business. Be the FIRST to understand and apply technical breakthroughs to your enterprise.

Name

First

Last

Company

Email

SUBMIT

POPULAR ARTICLES

NeurIPS 2021 – 10 Papers You Shouldn't Miss

A Guide To Knowledge Graphs

Why Graph Theory Is Cooler Than You Thought

10 Leading Language Models For NLP In 2021

BERT Inner Workings

Pretrain Transformers Models in PyTorch Using Hugging Face Transformers

More Articles

TOPICS

Bots

Brands

Business

China

Commerce

Computer Vision

Conversational AI

Customer Service

Cybersecurity