PRE-TRAINED MODELS AND SELF-SUPERVISED LEARNING FOR NLP: MODELS AND METHODS

Junyi Du
Department of Computer Science
University of Southern California
941 Bloom Walk, Los Angeles, CA 90089
junyidu@usc.edu

1 PRE-TRAINED WORD REPRESENTATIONS - WORD EMBEDDINGS

The most bottom step across all modern NLP models is converting symbolic words to input vectors. To perform well on most NLP tasks, we want to capture the similarity and difference between words. There's a very successful idea about representations of word called **distributional semantics**: A words meaning is given by the words that frequently appear close-by.

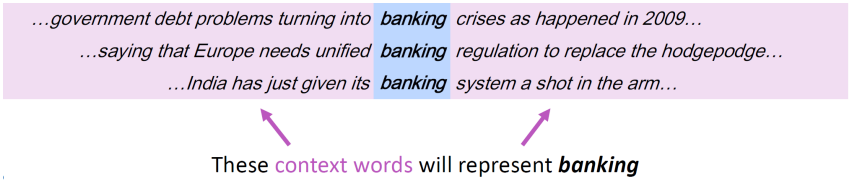

Figure 1: Context of a pivot word "banking"

To obtain word representations, it's a common idea to encode the context of a word with a language model on a large corpus, then capture the dense word embeddings in the model.

1.1 SKIP-GRAM MODEL

The Skip-gram Model (Mikolov et al., 2013a) used by the famous word embedding Word2Vec (Mikolov et al., 2013b) follows the idea of distributional semantics. Given a center word, the Skip-gram model try to predict the context words for some window size around the center word. The objective of the Skip-gram model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p\left(w_{t+j}|w_t\right) \tag{1}$$

The Skip-gram defines $p\left(w_{t+j}|w_t\right)$ as:

$$p\left(w_O|w_I\right)=\frac{\exp\left(v'_{w_O}{}^{\top}v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left(v'_w{}^{\top}v_{w_I}\right)} \tag{2}$$

where v_w and v'_{uv} are the word embeddings in the model. They train the model on 100B words from various news articles, then capture the word embeddings in the model.

1.2 GLOVE

GloVe is another algorithm for obtaining vector representations for words. It's a count-based model, motivated by the idea that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. The embeddings generated using the Word2Vec and Glove tend to perform very similarly in downstream NLP tasks.

1.3 UTILIZING WORD EMBEDDINGS

Since the advent of Word2Vec, the standard way of conducting NLP projects has been – word embeddings pretrained on large amounts of unlabeled data are used to initialize the first layer of a neural network, the rest of which is then trained on data of a particular task. This technique is so success that it's a standard component of many NLP architectures.

1.4 BEYOND WORD EMBEDDINGS

However, these word embeddings for learning word vectors only allow a single context independent representation for each word. The word embeddings doesn't address polysemy- they presume that a words meaning is relatively stable across sentences. Since then, some works focused on context-dependent representations learned by language model (Melamud et al., 2016; Peters et al., 2017) or encoder of neural machine translation (MT) system model (CoVe) (McCann et al., 2017).

Recently advances of ULMFiT (Howard & Ruder, 2018), ELMo (Peters et al., 2018), GPT (Radford et al.) and BERT (Devlin et al., 2018) lead the shift of training paradigm in NLP: going from just initializing the first layer of models with pre-trained word embeddings, to pre-training the entire model with deep hierarchical representations from language model, to achieve state-of-the-art on a diverse range of tasks in NLP.

2 EMBEDDINGS FROM LANGUAGE MODELS (ELMO)

ELMo is the first pre-trained language model that attracted attention of the whole NLP community, since it significantly improves the state of the art in every considered case across a range of challenging language understanding problems. The motivation for ELMo is that word embeddings should incorporate both word-level characteristics as well as contextual semantics.

2.1 BI-LSTM LANGUAGE MODEL

Instead of taking just the final layer of a deep bi-LSTM language model as the word representation (e.g., CoVe), ELMo representations are a function of all of the internal layers of the bi-LSTM language model. Here, a bi-LSTM language model is to maximizes the log likelihood of token in both forward and backward directions:

$$\begin{aligned}\sum_{k=1}^{N}(&\log p\left(t_k|t_1,\ldots,t_{k-1};\Theta_x,\vec{\Theta}_{LSTM},\Theta_s\right)\\ &+\log p\left(t_k|t_{k+1},\ldots,t_N;\Theta_x,\overleftarrow{\Theta}_{LSTM},\Theta_s\right)\Big)\end{aligned} \tag{3}$$

2.2 UTILIZING ELMO

For inclusion in a downstream model, ELMo collapses all layers into a single vector, by assigning each layer a task-specific weight:

$$\mathbf{ELMo}_k^{task} = E\left(R_k;\Theta^{task}\right) = \gamma^{task}\sum_{j=0}^{L} s_j^{task}\mathbf{h}_{k,j}^{LM} \tag{4}$$

We can inject this vector into downstream NLP models as a deep contextualized word vector. Similar to word embeddings, the pre-trained bi-LSTM can be fine-tuned for specific tasks.

2.3 WORD SENSE DISAMBIGUATION

Since adding ELMo improves task performance over word vectors alone, the biLMs contextual representations must encode information generally useful for NLP tasks that is not captured in word vectors. Intuitively, the bi-LSTM must be disambiguating the meaning of words using their context. Consider play, a highly polysemous word. The Figure 2 shows nearest neighbors to play using GloVe vectors. They are spread across several parts of speech (e.g., played, playing as verbs, and

player, game as nouns) but concentrated in the sports related senses of play. In contrast, the bottom two rows show nearest neighbor sentences using the biLMs context representation of play in the source sentence. In these cases, the biLM is able to disambiguate both the part of speech and word sense in the source sentence.

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Figure 2: Nearest neighbors to play using GloVe and the context embeddings from a biLM.

2.4 Hierarchical Representations

An observation is that the higher level states of the bi-LSTM capture context, while the lower level captures syntax well. Figure 4 shown empirically by comparing the performance of 1st layer and 2nd layer embeddings. While the 1st layer performs better on POS tagging, the 2nd layer achieves better accuracy for a word-sense disambiguation task.

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*

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

Figure 3: (Left)Fine grained WSD(Word sense disambiguation) F1. (Right) Test set POS tagging accuracies for PTB.

3 Universal Language Model Fine-tuning for Text Classification (ULMFiT)

The idea of using generative pre-trained LM + task-specific fine-tuning was first explored in ULMFiT. While basically similar to ELMo (except using a direct fine-tune on last layer output), ULMFiT achieve good transfer learning results on downstream language classification tasks with some interesting techniques:

1. **Discriminative fine-tuning**: tune each layer with different learning rates;
2. **Gradual unfreezing**: gradually unfreezing the model layers starting from the last one to avoid catastrophic forgetting.

4 Generative Pre-Training Transformer (OpenAI GPT)

Following the similar idea of ELMo, OpenAI GPT expands the unsupervised language model to a much larger scale by training on a giant collection of free text corpora. Despite of the similarity, GPT has two major differences from ELMo. The model architectures are different: ELMo uses a

concatenation of independently trained left-to-right and right-to-left multi-layer LSTMs, while GPT is a multi-layer uni-directional Transformer Language Model (Vaswani et al., 2017).

4.1 Bi-LSTM and Transformer: Differences

Remember RNN and LSTM and derivatives use mainly sequential processing over time, long-term information has to sequentially travel through all cells before getting to the present processing cell. This is the cause of vanishing gradients, and also the cause of the hardware-unfriendly.

Compared to RNN, a self-attention layer in Transformer connects all positions with a constant number of sequentially executed operations. In terms of training speed, Transformer is more hardware-friendly since it is easier to parallelize at training time.

4.2 Utilizing GPT

The use of contextualized embeddings in downstream tasks are slightly different: ELMo feeds embeddings into models as additional features, while GPT fine-tunes the same base model for all end tasks, while adding a task-specific layer to the final layer output of the last token.

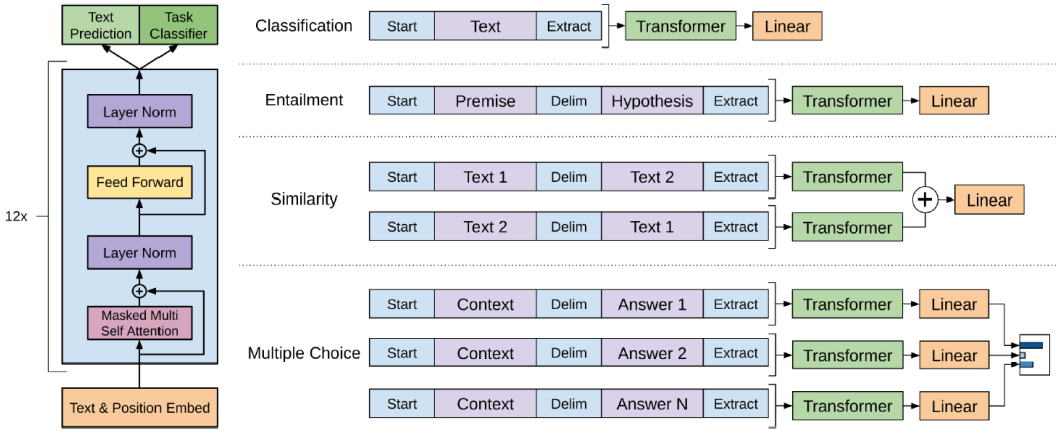

Figure 4: GPT architecture and training objectives

4.3 Uni-directional Transformer Language Model

Despite of the success of Transformer architecture on Seq2Seq tasks, experiments of GPT shows slightly performance improvement to ELMo in some NLP tasks. One of the reason could be the uni-directional Transformer language model used in GPT.It is a forward language model to predict future word based on existing words, without capability of capturing backward information.

4.4 GPT-2

GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data, and shows great potential in zero-shot learning. GPT-2 outperforms other language models trained on specific domains (like Wikipedia, news, or books) without needing to use these domain-specific training datasets. On language tasks like question answering, reading comprehension, summarization, and translation, GPT-2 begins to learn these tasks from the raw text, using no task-specific training data.

5 BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT)

BERT is a direct descendent to GPT train a large language model on free text and then fine-tune on specific tasks without customized network architectures. Compared to GPT, the largest difference and improvement of BERT is to make training bi-directional. The model learns to predict both context on the left and right. The model architecture of BERT is a multi-layer bidirectional Transformer encoder.

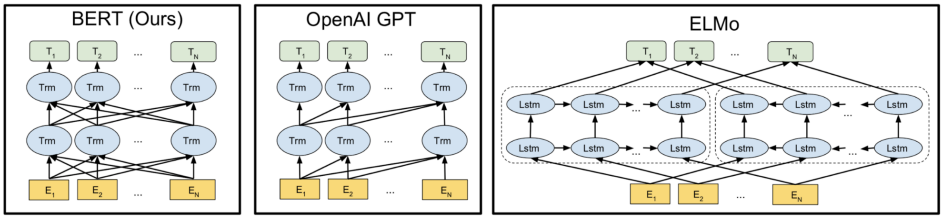

Figure 5: Comparison of BERT, OpenAI GPT and ELMo model architectures.

5.1 MASKED LANGUAGE MODEL

A major novelty of BERT could be the masked language modeling task. It is unsurprising to believe that a representation that learns the context around a word (bi-directional) rather than just after the word (uni-directional) is able to better capture its meaning, both syntactically and semantically. BERT encourages the model to do so by training on the mask language modeling task:

1. Randomly mask 15% of tokens in each sequence. Because if we only replace masked tokens with a special placeholder [MASK], the special token would never be encountered during fine-tuning. Hence, BERT employed several heuristic tricks:
   (a) with 80% probability, replace the chosen words with [MASK];
   (b) with 10% probability, replace with a random word;
   (c) with 10% probability, keep it the same.
2. The model only predicts the missing words, but it has no information on which words have been replaced or which words should be predicted. The output size is only 15% of the input size.

5.2 UTILIZING BERT

BERT fine-tuning requires only a few new parameters added, just like OpenAI GPT. For classification tasks, we get the prediction by taking the final hidden state of the special first token [CLS] as the input of the task-specific layer, shown in Figure 6 and 7.

REFERENCES

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018. doi: 10.18653/v1/p18-1031. URL `http://dx.doi.org/10.18653/v1/p18-1031`.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pp. 6294–6305, 2017.

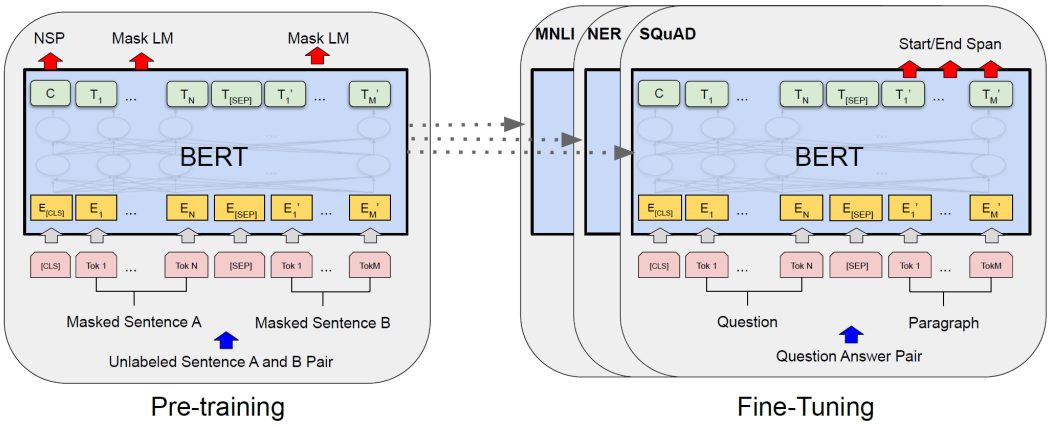

Figure 6: Overall pre-training and fine-tuning procedures for BERT

Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pp. 51–61, 2016.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013b.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

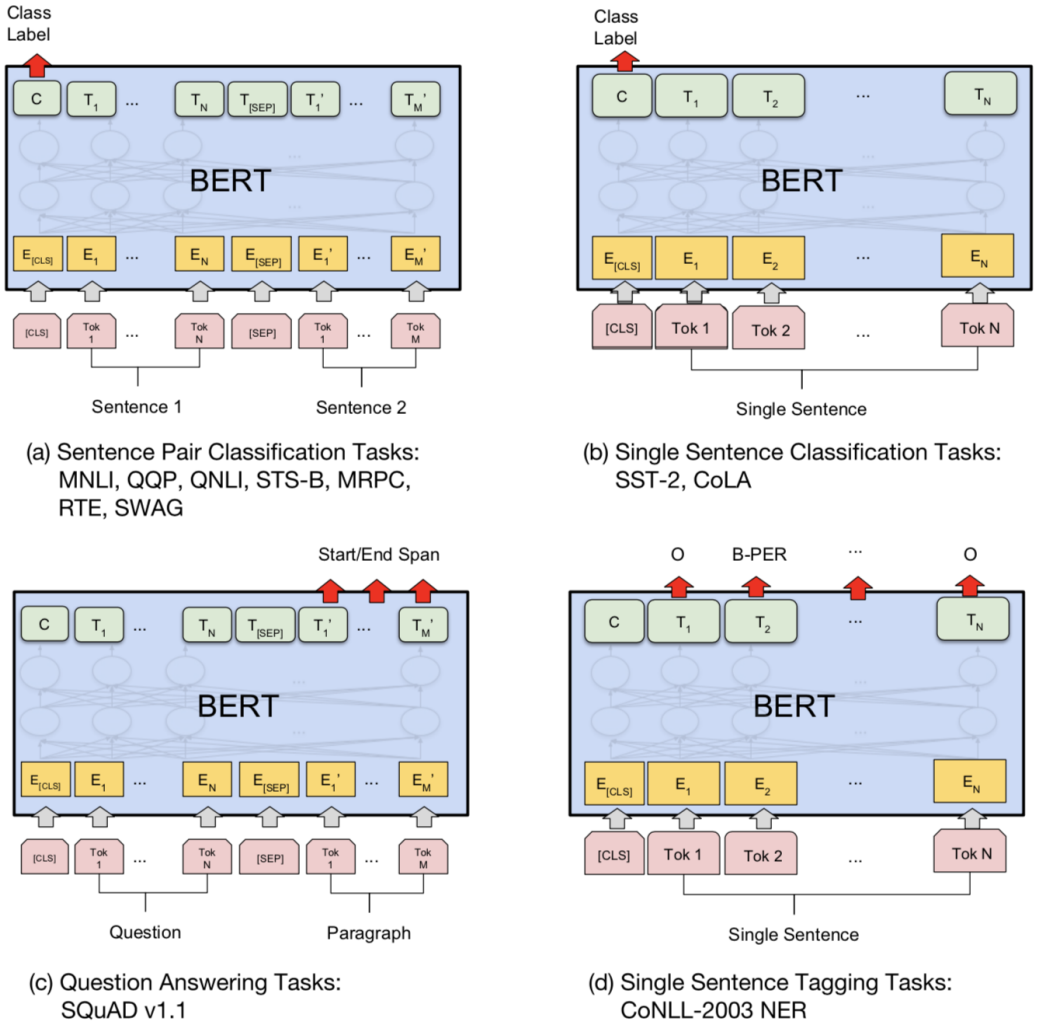

Figure 7: Finetune BERT models for downstream tasks