Contents

1. Introduction to Pre-trained Language Representations
2. (Feature-based approach) ELMo
3. (Fine-tuning approach) OpenAI GPT
4. OpenAI GPT3
5. (Fine-tuning approach) BERT
6. Summary

Introduction: Transfer Learning

- How can we take advantage of distributed word representation?
  - Transfer Learning

- What is Transfer Learning?

Transfer Learning using word representations
- Pre-trained Word Representation

Text Corpus

Pre-training

Unsupervised Learning with unlabeled data

Supervised Learning with labeled data

Adaptation in target task

Classification Sequence Labeling Question & Answering...
Introduction: Pre-trained Language Representations

Let’s Dive into Pre-trained Language Representation

Two kinds of Pre-trained Language Representations
- 1) Feature-based approach
- 2) Fine-tuning approach

Feature-based approach
- Use task-specific architectures that include the pre-trained representations as additional features
  - Learned representations are used as features in a downstream model
- ELMo (Peters et al. 2018)
  - In GloVe, Word2vec method,
  - Polysemous words refer to same representation no matter the context
    - “I am a big fan of Mozart”
      - fan = [0.5, -0.3, 0.228, 0.9, 0.31]
    - “I need a fan to cool the heat”
      - fan = [0.5, -0.3, 0.228, 0.9, 0.31]
ELMo (Peters et al. 2018) (Cont’d)

- Let the words be represented according to the context !!!

Feature-based approach: ELMo

- ELMo (Peters et al. 2018) (Cont’d)
  - Two components of the ELMo
    - biLSTM pre-training part
      - Use vectors derived from a bidirectional LSTM trained with a coupled LM Objective
    - ELMo part
      - Task specific combination of the representations in the biLM

biLSTM part

- Two objectives: predicting word in forward direction, backward direction
  - Forward: \( p(t_1, t_2, \ldots, t_n) = \prod_{i=1}^{n} p(t_i | t_1, t_2, \ldots, t_{i-1}) \)
    - Task of predicting next token
  - Backward: \( p(t_1, t_2, \ldots, t_n) = \prod_{i=1}^{n} p(t_i | t_{i+1}, t_{i+2}, \ldots, t_n) \)
    - Task of predicting previous token

Overall objective is to jointly maximizes the log likelihood of the forward and backward directions

\[
\mathcal{L}(\theta) = \sum_{i=1}^{n} (\log p(t_i | t_1, t_2, \ldots, t_{i-1}; \theta_2, \theta_{\text{biLM}}, \theta_2) + \log p(t_i | t_{i+1}, t_{i+2}, \ldots, t_n; \theta_2, \theta_{\text{biLM}}, \theta_2))
\]
**Feature-based approach : ELMo**

- **ELMo part**
  - Part where the ELMo word representation is defined
  - Task specific combination of the intermediate layer representations in the bILM

$$R_k = \{h_{timestep}(k)_{layer(j)} \}_{j=0, \ldots, L}$$

![ELMo Diagram](image)

- **ELMo (Cont'd)**
  - ELMo collapses all layers in $R$ into a single vector,
  - $ELMo_k = E(R_k; \Theta)$

- **ELMo Evaluation**
  - Question Answering, SQuAD: average $F_1$ score - +1.4% than SOTA
  - Textual Entailment, SNLI: accuracy score - +0.7% when SOTA + ELMo
  - Semantic Role Labeling, SRL: average $F_1$ score - +3.2% when SOTA reimplementation + ELMo
  - Coreference resolution, Coref: average $F_1$ score - +3.2% when SOTA reimplementation + ELMo
  - Named Entity Extraction, NER: average $F_1$ score - +0.3% when SOTA + ELMo
  - Sentiment Analysis, SST-5: accuracy score - +1% when SOTA reimplementation + ELMo
ELMo Evaluation : effects of ‘Deep biLM’ part

- Deep biLM effects

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>Model</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet 1st Sense Baseline</td>
<td>65.9</td>
<td>Collobert et al. (2011)</td>
<td>97.3</td>
</tr>
<tr>
<td>Raganato et al. (2017a)</td>
<td>69.9</td>
<td>Ma and Hovy (2016)</td>
<td>97.6</td>
</tr>
<tr>
<td>Iacobucci et al. (2016)</td>
<td>76.1</td>
<td>Ling et al. (2015)</td>
<td>97.8</td>
</tr>
<tr>
<td>CoVe, First Layer</td>
<td>59.4</td>
<td>CoVe, First Layer</td>
<td>93.7</td>
</tr>
<tr>
<td>CoVe, Second Layer</td>
<td>64.7</td>
<td>CoVe, Second Layer</td>
<td>92.8</td>
</tr>
<tr>
<td>blLM, First Layer</td>
<td>67.4</td>
<td>blLM, First Layer</td>
<td>97.3</td>
</tr>
<tr>
<td>blLM, Second Layer</td>
<td>66.0</td>
<td>blLM, Second Layer</td>
<td>96.8</td>
</tr>
</tbody>
</table>

- using biLM’s context representation,
  - Disambiguate word sense in the source sent (Word Sense Disambiguation test)
    - Deeper layers catch more of **semantic information**
  - Disambiguate part of speech in the source sent (POS tagging test)
    - Shallower layers catch more of **syntactic information**

Feature-based approach : ELMo

Fine-tuning approach to pre-trained Word Representation

- Fine-tuning approach
  - Trained on the downstream tasks by simply fine-tuning the pre-trained params
    - Minimal task-specific parameters
    - “fine-tune effectively with minimal changes to the architecture of the pre-trained model.” (Radford et al. 2018)
  - OpenAI GPT, BERT

OpenAI-GPT

- From ‘Improving Language Understanding by Generative Pre-Training’ paper, Radford et al. 2018
- Make use of Transformers model into unsupervised pre-training
- It is then transferred to discriminative tasks (downstream task)

- Multi-layer Transformer decoder block for the language model

Framework

- First stage, learning a high-capacity language model on a large corpus of text(BooksCorpus dataset)
- Followed by a fine-tuning stage where the model adapts to a discriminative task with labeled data

First Stage, Unsupervised pre-training

- Language model objective with large corpus of unlabeled data
  \[ L_1(\theta) = \sum_{t \in U} \log P(u_t|u_{t-k}, ..., u_{t-1}, \theta) \]
  - \( k \): size of the context window
  - \( U = \{ u_1, u_2, ..., u_n \} \): unsupervised corpus of tokens
  - \( P \): modeled using a neural network with parameters \( \theta \)
- Multi-layer Transformer decoder block for the language model
Fine-tuning approach: Open-AI GPT

- **Transformer (Decoder Block)**
  - Linearly transform \( d_2 : Q \) (Query), linearly transform the others: \( K \) (Key)
  - Dot product of every neighboring position
  - (mask out future words logits by multiplying 10e-9)
  - Softmax the logits Convex Combination of the softmax result then put through FFNN
    \[ d_2' = d_2 \]

  ![Decoder Self-Attention Diagram](image)

- **First Stage, Unsupervised pre-training (Cont'd)**
  - Overview
    - Inputs tokenized by spaCy tokenizer
    - Inputs are fed into 12 layers of Transformer blocks in each time step
    - Last layer produce probability distribution over BPE based vocabulary (40,000)

  ![OpenAI GPT Diagram](image)

  \[ L = 12 \]

  ![Transformer Block Diagram](image)

  **Example:** "I want to build a language model architecture …"
Second Stage, Supervised fine-tuning

- Once the first stage is finished with unsupervised corpus of tokens, our language model parameter is pre-trained thus it is available as pre-trained word representation.
- Adapt the parameters from first stage to the supervised target task with labeled dataset \( \mathcal{C} = \{c_1, \ldots, c_n\} \) where \( c_i = \{x^i_1, \ldots, x^i_m, y^i\} \).
- The inputs are passed through our pre-trained model to obtain the final transformer block’s activation \( h^m \).
  - \( m \) : last token index, \( i \) : last layer index

\[
P(y|x^1, \ldots, x^n) = \text{softmax}(h^m W_f)
\]

There is one problem in Second Stage ...

- Input form of the Open-AI GPT looks like
  - \( \mathcal{U} = \{u_1, u_2, \ldots, u_m\} \)
- In task-specific tasks such as question answering, textual entailment
  - They have structured inputs
    - ex) (document, question, answers)

See the difference between Feature-based & Fine-tuning approach?

- Feature-based: You fit your representation to the task
- Fine-tuning: task is fitted to the representation learning
Open-AI GPT 3

- **Task-agnostic Language Model**
  - Fine-tuning 없는 범용성이 좋은 Task-agnostic NLP 모델
  - Zero-shot Learning?

Open-AI GPT 3

- **Task-agnostic Language Model**
  - Few-shot Learning

Open-AI GPT 3

- **Model**: GPT-2와 동일한 구조
  - 파라미터 수 증가 (175B 파라미터)

- **데이터 소개**
  - 45TB나 되는 150 Billion Token (500GB 전처리된 텍스트)

Open-AI GPT 3

- **문장 생성 및 Cloze 퀴즈 맞추기 태스크에 대한 성능**

- **Translation**

Aline was friends with Bob. Aline went to visit her friend ____ → Bob
George bought some baseball equipment; a ball, a glove, and a ____ →
**Fine-tuning approach : BERT**

- **BERT (Bidirectional Encoder Representations from Transformers)**
  - Paper published in NAACL 2019 by Google AI
  - Won Best Long Paper

**BERT** is designed to pre-train representations by jointly conditioning on both left and right context in all layers.

- **How?** By training on two new tasks!
  - Word Representation learning via "masked language model" task
  - "Next Sentence Prediction" task

**Masked Language Model?**
- How about mask one of the tokens in a sentence and guess what that is
- Ex) As long \( \text{MASK} \) you love me : "as"
- Can take account of the context after the target token

**Next Sentence Prediction task?**
- Guessing appropriate sequence after which follows
- Ex) current sequence : "I think I mastered the concept"

Appropriate next sequence

Inappropriate next sequence

**Open-AI GPT cannot take on right to left context**

- Deep bidirectional model is more powerful than either a left-to-right model (GPT) or the shallow concatenation of a left-to-right and right-to-left model (ELMo)
- Every token can only attend to previous tokens in the self-attention layers of the Transformer
- This is due to the fact that standard Language Models can only be trained left-to-right or right-to-left
  - Since bidirectional conditioning would allow each word to indirectly "see itself" in a multi-layered context.
  - Ex) language model training "As long as you love me" from left to right

Illegal !!!

**Open-AI GPT cannot take on right to left context**

- Deep bidirectional model is more powerful than either a left-to-right model (GPT) or the shallow concatenation of a left-to-right and right-to-left model (ELMo)
- Every token can only attend to previous tokens in the self-attention layers of the Transformer
- This is due to the fact that standard Language Models can only be trained left-to-right or right-to-left
  - Since bidirectional conditioning would allow each word to indirectly "see itself" in a multi-layered context.
  - Ex) language model training "As long as you love me" from left to right

Illegal !!!

**Open-AI GPT cannot take on right to left context**

- Deep bidirectional model is more powerful than either a left-to-right model (GPT) or the shallow concatenation of a left-to-right and right-to-left model (ELMo)
- Every token can only attend to previous tokens in the self-attention layers of the Transformer
- This is due to the fact that standard Language Models can only be trained left-to-right or right-to-left
  - Since bidirectional conditioning would allow each word to indirectly "see itself" in a multi-layered context.
  - Ex) language model training "As long as you love me" from left to right

Illegal !!!
Fine-tuning approach : BERT

- **Overall Architecture**
  - Multi-layer bidirectional Transformer Encoder
    - Transformer is now able to refer to the right-to-left context due to the changed training objectives
  - Task specific layer on top of the model

- **Overall Procedure**
  - How to construct an input?
    - [CLS] + sentence A + [SEP] + sentence B
    - Just like 'start', 'delim' tokens: 'CLS', 'SEP' tokens
  - Input example
    - Ex) ['CLS', 'my', 'dog', 'is', 'cute', 'SEP', 'he', 'likes', 'playing', 'SEP']

Two model sizes
- \( \text{BERT}_{\text{BASE}} \): \( L=12, H=768, A=12 \)
- \( \text{BERT}_{\text{LARGE}} \): \( L=24, H=1024, A=16 \)
- Where \( L \) = number of layers, \( H \) = hidden size, \( A \) = self-attention head

Input is represented and fed to the model summing
- 1) WordPiece embeddings (Wu et al. 2016)
- 2) Segment embeddings
- 3) Learned positional embeddings
Overall Procedure (Cont'd)

1) WordPiece embeddings (Wu et al. 2016)

- Use embeddings trained with objectives that selects D wordpieces such that the resulting corpus is minimal in the number of wordpieces when segmented according to the chosen wordpiece model.

- Data-driven tokenization method that aims to achieve a balance between vocab size and out-of-vocab words
  
  - “strawberries” = “straw” + “berries”

- Enables BERT to only store 30,522 “words” in its vocab and very rarely encounter out-of-vocab words

2) Segment Embeddings

- Segment Embeddings help BERT distinguish the tokens in input pair
  
  - Sent A: Index 0 → 768 vec
  - Sent B: index 1 → 768 vec

- Index 0 when input only contains one input sentence

Overall Procedure (Cont'd)

2) Positional Embeddings

- BERT is designed to process input sequences of up to length 512

- BERT learns a vector representation for each position

Multihead attention in Transformer

- Sentence is about ‘who,’ ‘did what,’ and ‘to whom’
- In CNN, different filters learn the concept of ‘who,’ ‘did what,’ and ‘to whom.’
- Self-attention can’t pick out different information from different places
  
  - It’s just a linear combination everywhere

Architecture of Transformer

- Encoder Self-Attention

- Self-Attention: Averaging

\[ A(Q, K, V) = \text{softmax} \left( \frac{Q K^T}{\sqrt{d_k}} \right) V \]
**Architecture of Transformer**

- **Multihead attention in Transformer (Cont’d)**
  - Apply self-attention multiple times, each of them linearly transform token so that it conveys different information of interests.

![Attention head: Who](image1)

![Attention head: Did What?](image2)

**Fine-tuning approach : BERT**

- **Task #1 : Masked Language Model (MLM)**
  - Mask α% of the input tokens to be predicted.
  - The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocab.

![BERT Diagram](image3)

**Task #1 : Masked Language Model (MLM) (Cont'd)**

- Two downsides of this approach
  - 1st, we are creating a mismatch between pre-training and fine-tuning
    - 'MASK' token is never seen during fine-tuning time.
  - Take special steps
    - Ex) “my dog is hairy” and ‘hairy’ is randomly selected
      - 80% of the time (0.8 × α%) : MASK
        - "my dog is [MASK]"
      - 10% of the time (0.1 × α%) : Replace with a random word
        - "my dog is apple"
      - 10% of the time (0.1 × α%) : Keep the word unchanged
        - "my dog is hairy"

![Diagram showing masking](image4)

**Task #1 : Masked LM (Cont’d)**

- Two downsides of this approach
  - 2nd, only 15% of tokens are predicted in each batch
    - which suggests that more pre-training steps may be required for the model to converge.
    - Left-to-right model predicts every token so it converges faster.
  - However, empirical improvements of the MLM model far outweigh the increased training cost.

![Graph showing training improvements](image5)
Fine-tuning approach: BERT

Task #2: Next Sentence Prediction (NSP)
- To equip with ability to understand the relationship between two text sentences which is not directly captured by LM.
  - Question Answering (QA), Natural Language Inference (NLI) tasks
- Pre-train binary next sentence prediction task

Task #2: Next Sentence Prediction (NSP) (Cont’d)

Effect of training with the task of NSP
- No NSP: trained without the NSP task
- LTR & No NSP: trained without the NSP task + only left-to-right LSTM

Removing NSP hurts performance significantly on QNLI, MNLI, SQuAD which depend largely on the relationship between two sentences.
Fine-tuning approach: BERT

- Now that the pre-trained model is ready, start fine-tuning!
  - No need to construct another model for another task
  - Just add the output layer parts!

Pre-trained model ready

Just add params for specific task

Fine-tuning

Fine-tuning approach: BERT

- Fine-tuning
  - [CLS] embedding ($C \in \mathbb{R}^d$) is mostly used for fine-tuning tasks
  - Only new parameter ($W \in \mathbb{R}^K \times d$) for classification layer
    - $K$ is the number of classifier labels, e.g., 2 for ['IsNext', 'NotNext']
  - Label probabilities ($P \in \mathbb{R}^K = \text{softmax}(CW')$)

Pre-trained model ready

Just add params for specific task

Fine-tuning

Fine-tuning approach: BERT

- Fine-tuning in spanning or token-level task
  - Modified slightly to use different number and different location of the hidden-states other than [CLS]

Fine-tuning approach: BERT

- Result
  - GLUE (General Language Understanding Evaluation) Dataset
    - Obtains 4.5% and 7.0% respective average accuracy improvement over the prior SOTA

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-m (mm)</th>
<th>QQP</th>
<th>QNLI</th>
<th>SST-2</th>
<th>CoLA</th>
<th>STS-B</th>
<th>MRPC</th>
<th>RTE</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66.1</td>
<td>82.3</td>
<td>95.2</td>
<td>35.0</td>
<td>81.0</td>
<td>86.0</td>
<td>61.7</td>
<td>74.0</td>
</tr>
<tr>
<td>BiLSTM+MLM+Att</td>
<td>76.4/76.4</td>
<td>64.8</td>
<td>70.6</td>
<td>90.4</td>
<td>36.0</td>
<td>77.3</td>
<td>84.9</td>
<td>66.8</td>
<td>77.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>82.1/81.1</td>
<td>70.3</td>
<td>87.4</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.1</td>
</tr>
<tr>
<td>BERTBASE</td>
<td>84.1/83.4</td>
<td>72.1</td>
<td>90.5</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>87.6</td>
</tr>
<tr>
<td>BERTBASE2023 NER</td>
<td>86.7/85.9</td>
<td>72.1</td>
<td>92.7</td>
<td>94.9</td>
<td>60.0</td>
<td>86.5</td>
<td>89.3</td>
<td>78.1</td>
<td>82.1</td>
</tr>
</tbody>
</table>

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard).

The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. BERT and OpenAI GPT are single-model, single-task. F1 scores are reported for QQP and MRPC. Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.
Fine-tuning approach : BERT

Result (QA test)

<table>
<thead>
<tr>
<th>System</th>
<th>Dev EM</th>
<th>F1</th>
<th>Test EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>-</td>
<td>-</td>
<td>82.3</td>
<td>91.2</td>
</tr>
<tr>
<td>4 Ensemble - mnet</td>
<td>-</td>
<td>-</td>
<td>86.0</td>
<td>91.7</td>
</tr>
<tr>
<td>#2 Ensemble - QANet</td>
<td>-</td>
<td>-</td>
<td>84.5</td>
<td>90.6</td>
</tr>
<tr>
<td>Published</td>
<td>-</td>
<td>-</td>
<td>85.6</td>
<td>85.8</td>
</tr>
<tr>
<td>BDAF + ELMo (Single)</td>
<td>-</td>
<td>-</td>
<td>81.2</td>
<td>87.9</td>
</tr>
<tr>
<td>R M Reader (Ensemble)</td>
<td>-</td>
<td>-</td>
<td>82.3</td>
<td>93.6</td>
</tr>
<tr>
<td>Ours</td>
<td>-</td>
<td>-</td>
<td>80.8</td>
<td>88.5</td>
</tr>
<tr>
<td>BERTBASE (Single)</td>
<td>-</td>
<td>-</td>
<td>84.1</td>
<td>90.9</td>
</tr>
<tr>
<td>BERTLARGE (Single)</td>
<td>-</td>
<td>-</td>
<td>85.8</td>
<td>91.8</td>
</tr>
<tr>
<td>BERTLARGE (Ensemble)</td>
<td>-</td>
<td>-</td>
<td>84.2</td>
<td>91.1</td>
</tr>
<tr>
<td>BERTLARGE (Ogg+TriviaQA)</td>
<td>-</td>
<td>-</td>
<td>86.2</td>
<td>92.2</td>
</tr>
<tr>
<td>BERTLARGE (Em+TriviaQA)</td>
<td>-</td>
<td>-</td>
<td>87.6</td>
<td>93.2</td>
</tr>
</tbody>
</table>

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

Result SQuAD 2.0

- SQuAD 1.1 + ‘No Answer’ task
- +5.1 F1 improvement over the previous best system

<table>
<thead>
<tr>
<th>System</th>
<th>Dev EM</th>
<th>F1</th>
<th>Test EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>-</td>
<td>-</td>
<td>80.3</td>
<td>89.0</td>
</tr>
<tr>
<td>#1 Single - MIR-MRC (FM-Net)</td>
<td>-</td>
<td>-</td>
<td>86.9</td>
<td>89.5</td>
</tr>
<tr>
<td>#2 Single - mnet</td>
<td>-</td>
<td>-</td>
<td>74.8</td>
<td>78.0</td>
</tr>
<tr>
<td>Published</td>
<td>-</td>
<td>-</td>
<td>71.4</td>
<td>74.9</td>
</tr>
<tr>
<td>Ours</td>
<td>-</td>
<td>-</td>
<td>71.4</td>
<td>74.4</td>
</tr>
</tbody>
</table>

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

References

- Mikolov et al. 2013, “Efficient Estimation of Word Representations in Vector Space”
- Distributed Representations of Words and Phrases and their Compositionality (Mikolov et al. 2013)
- LM picture : https://upload.wikimedia.org/wikipedia/commons/d/dd/CBOW_etc_Skipgram.png
- Transfer Learning Tutorial by Sebastian Ruder, Matthew Peters, Swabha Swayamdipta, Thomas Wolf : https://docs.google.com/presentation/d/1fIhGikFPnb7G5kr58OvYC3GN4o7MznMoaAgaTvFCvAg/preview?hl=en&slideid=gs8bd596a1_0_0
- Semi-supervised Sequence Tagging with bidirectional language models, Peters et al. 2013
- Semi-supervised sequence tagging with bidirectional language models, Peters et al. 2017
- Deep contextualized word representations, Peters et al. 2018

- ELMo explanation blog : https://www.slideshare.net/shuntaro_y/a-review-of-deep-contextualized-word-representations-peters-2018
- ELMo, BERT, OpenAI GPT explanation blog : http://jalammar.github.io/illustrated-bert/
- Universal Language Model Fine-tuning for Text Classification (Jeremy Howard and Sebastian Ruder, 2018)
- Improving Language Understanding with Unsupervised Learning (Radford et al. 2018)
- Attention Is All You Need, Vaswani et al. 2017
- https://www.youtube.com/watch?v=5vqjKhSwBCY&t=811s
- OpenAI-GPT figure : https://www.topbots.com/generalized-language-models-ulmfit-openai-gpt/
- Input representation of BERT : https://medium.com/@_intel_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a
Thank you for your attention!

고 영 종 (Ko, Youngjoong)
nlplab.skku.edu