Pre-training Language Representation (ELMo, BERT, OpenAI GPT)

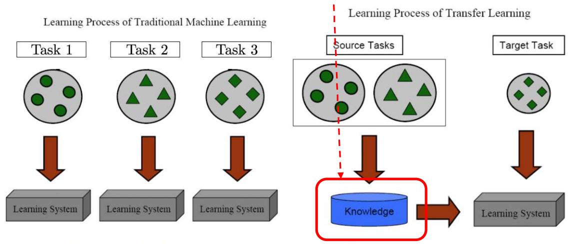
Ko, Youngjoong

Sungkyunkwan University

Contents

Introduction : Transfer Learning

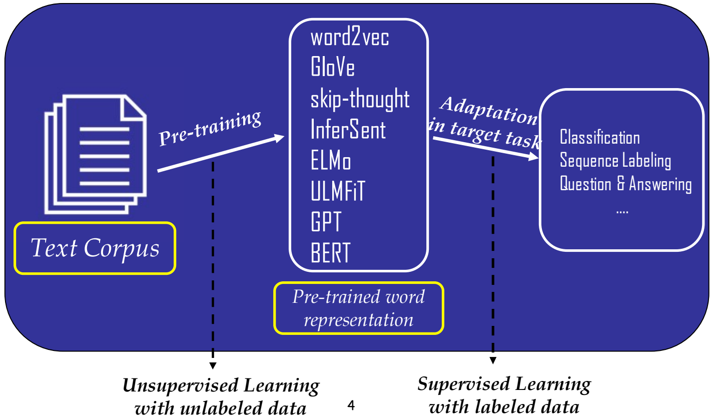
- **How can we take advantage of distributed word representation?**
  - Transfer Learning
- **What is Transfer Learning?**



(a) Traditional Machine Learning (b) Transfer Learning

Introduction : Transfer Learning

- **Transfer Learning using word representations**
  - Pre-trained Word Representation



Introduction : Pre-trained Language Representations

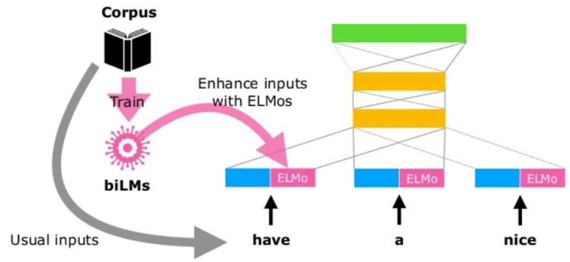
Let's Dive into Pre-trained Language Representation

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

5

Introduction : Pre-trained Language Representations

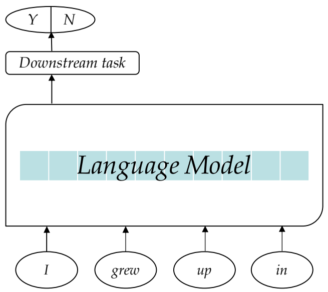
- ❖ **Feature-based approach**
  - ➢ Use task-specific architectures that include the pre-trained representations as additional features
    - ▪ Learned representations are used as features in a downstream model
  - ➢ ELMo



6

Introduction : Pre-trained Language Representations

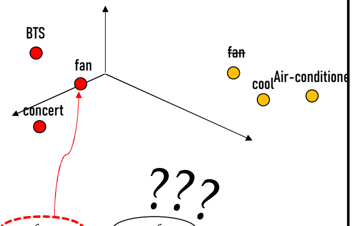
- ❖ **Fine-tuning approach**
  - ➢ Introduce minimal task-specific parameters
  - ➢ Trained on the downstream tasks by simply fine-tuning the pre-trained parameters
  - ➢ OpenAI GPT, BERT

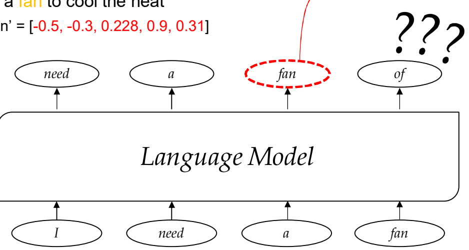


7

Feature-based approach : ELMo

- ❖ **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]



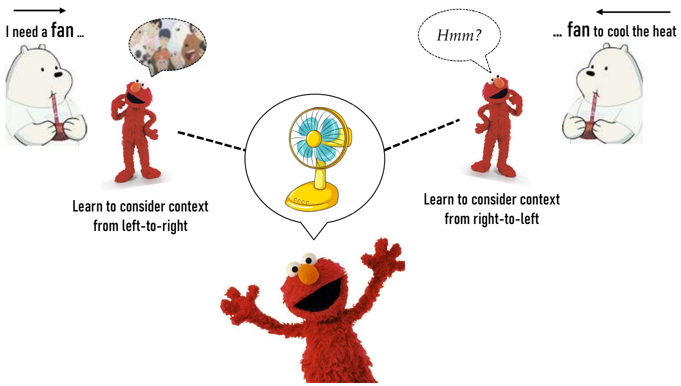


8

Feature-based approach : ELMo

❖ **ELMo (Peters et al. 2018) (Cont'd)**

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

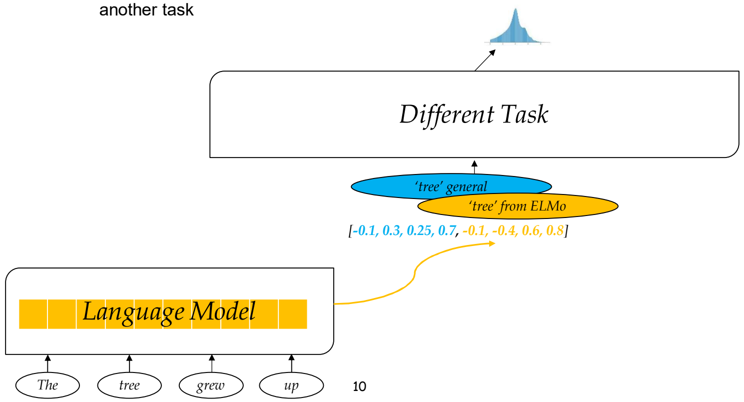


Feature-based approach : ELMo

❖ **ELMo (Peters et al. 2018) (Cont'd)**

➢ Trained task-specifically

- Learned parameters(weights) from language model are used as a **feature** for another task

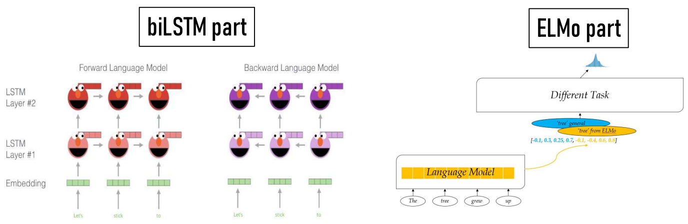


Feature-based approach : ELMo

❖ **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



Feature-based approach : ELMo

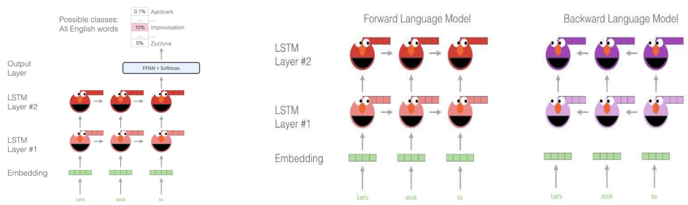
❖ **biLSTM part**

➢ Two objectives : predicting word in forward direction, backward direction

- Forward : $p(t_1, t_2, \dots, t_N) = \prod_{k=1}^{N} p(t_1 | t_1, t_2, \dots, t_{k-1})$
  - ✓ Task of predicting next token
- backward : $p(t_1, t_2, \dots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \dots, t_N)$
  - ✓ Task of predicting previous token

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

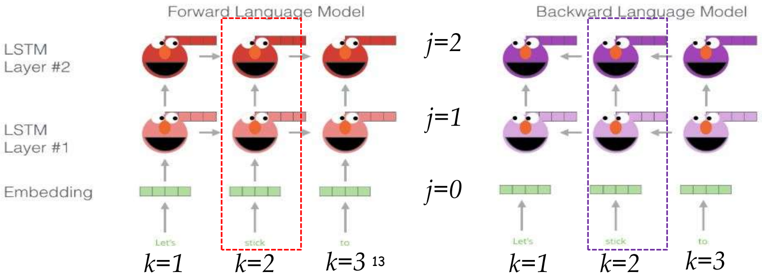
- $J(\boldsymbol{\theta}) = \sum_{k=1}^{N} (\log p(t_k | t_1, \dots, t_{k-1}; \theta_x, \vec{\theta}_{LSTM}, \theta_s) + \log p(t_k | t_{k+1}, \dots, t_N; \theta_x, \overleftarrow{\theta}_{LSTM}, \theta_s))$



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

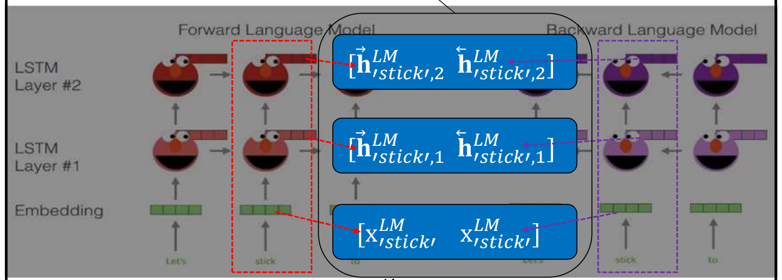
$$R_k = \{h^{LM}_{timestep(k),layer(j)} \mid j = 0, \ldots, L\}$$



Feature-based approach : ELMo

- **ELMo part (Cont'd)**
  - Layer representation trained from biLSTM part, R_k
  - ELMo collapses all layers in R into a single vector, $\boldsymbol{ELMo}_K = E(R_k; \Theta_e)$

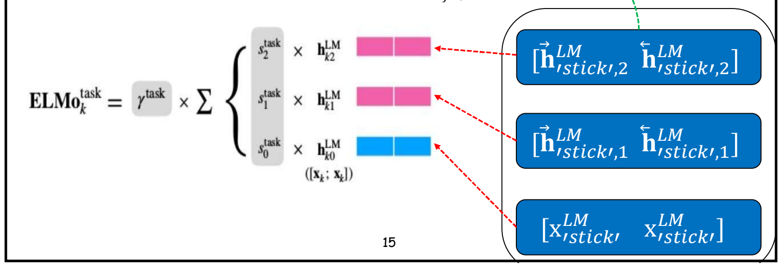
$$R_k = \{h^{LM}_{timestep(k),layer(j)} \mid j = 0, \ldots, L\}$$



Feature-based approach : ELMo

- **ELMo part (Cont'd)**
  - ELMo collapses all layers in R into a single vector,
  - Choices of $\boldsymbol{ELMo}_K = E(R_k; \Theta_e)$
    - Simplest case : $\boldsymbol{ELMo}_K = E(R_k) = h^{LM}_{k,L}$ (top layer) (Peters et al. 2017, McCann et al. 2017)
    - General case : compute a task specific weighting of all biLM layers down-stream task learns weighting parameters

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



Feature-based approach : ELMo

- **ELMo Evaluation**

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

- 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

Feature-based approach : ELMo

❖ **ELMo Evaluation : effects of 'Deep biLM' part**

- Deep biLM effects

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

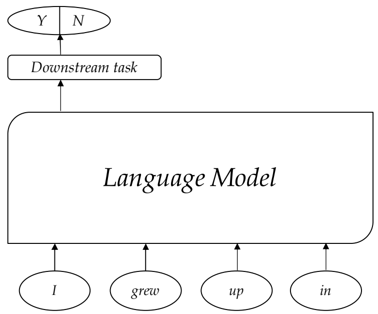
- using biLM's context representation,
  - Disambiguate word sense in the source sent (Word Sense Disambiguity test)
    - ✓ Deeper layers catch more of *semantic information*
  - Disambiguate part of speech in the source sent (POS tagging test)
    - ✓ Shallower layers catch more of *syntactic information*

17

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

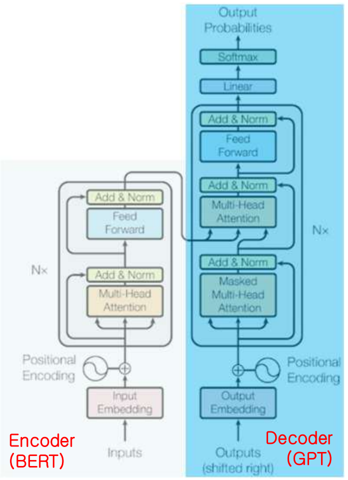


18

Fine-tuning approach : Open-AI GPT

❖ **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)



19

Fine-tuning approach : Open-AI GPT

❖ **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**

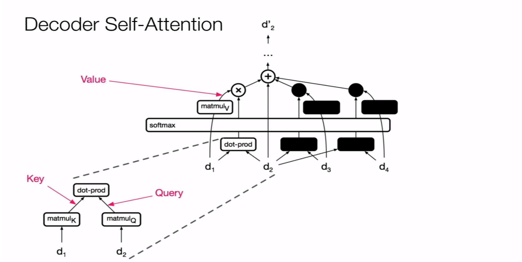
- Language model objective with large corpus of unlabeled data
- $L_1(\mathcal{U}) = \sum_i log\boldsymbol{P}(u_i|u_{i-k}, \dots, u_{i-1};\ \boldsymbol{\theta})$
  - k : *size of the context window*
  - $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$: *unsupervised corpus of tokens*
  - $\boldsymbol{P}$: *modeled using a neural network with parameters* $\boldsymbol{\theta}$
- Multi-layer Transformer decoder block for the language model

20

Fine-tuning approach : Open-AI GPT

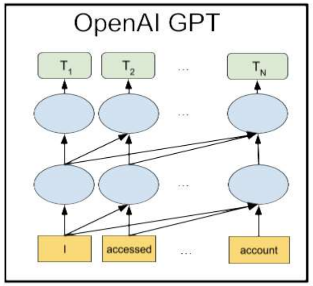
- **Transformer (Decoder Block)**
  - Linearly transform d_2 : Q (Query) , linearly transform the others : K (Key)
  - Dot product of every neighboring position
  - (mask out future words logits by multiplying 10e-9)
  - Softmax the logits Convex Combination of the softmax result then put through FFNN

    => Becomes the re-representation of $d_2 = d'_2$



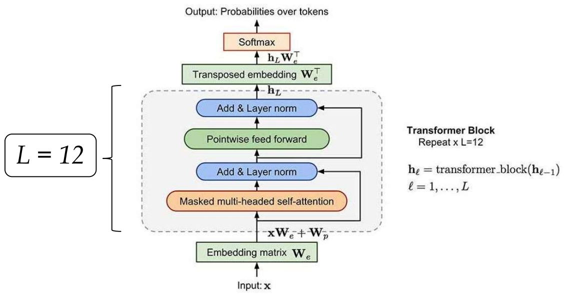
Fine-tuning approach : Open-AI GPT

- **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)



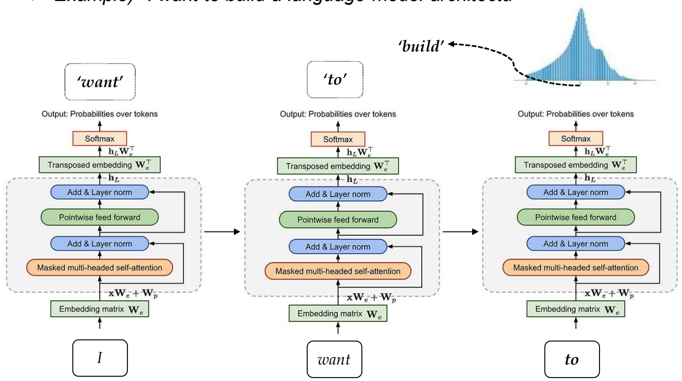
Fine-tuning approach : Open-AI GPT

- **First Stage, Unsupervised pre-training (Cont'd)**
  - 12 layers of Transformer blocks
    - i) Masked multi-headed self-attention over the input context tokens
    - ii) Followed by position-wise feedforward layers
    - iii) Softmax of feedforward result over the target tokens



Fine-tuning approach : Open-AI GPT

- **First Stage, Unsupervised pre-training (Cont'd)**
  - *Example) "I want to build a language model architecture …"*

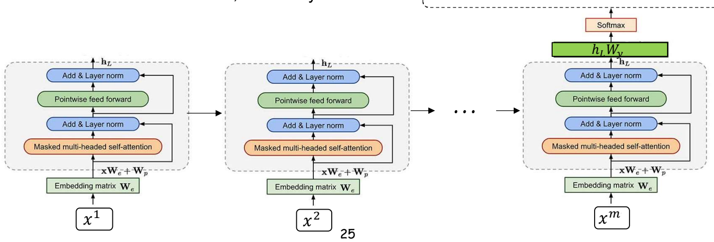


Fine-tuning approach : Open-AI GPT

❖ 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 $\mathcal{C} = \{c_1, \ldots, c_n\}$ $where\ c_1 = \{x^1, \ldots, x^m, y_1\}$
- The inputs are passed through our pre-trained model to obtain the final transformer block's activation h_l^m
  - m : last token index, l : last layer index

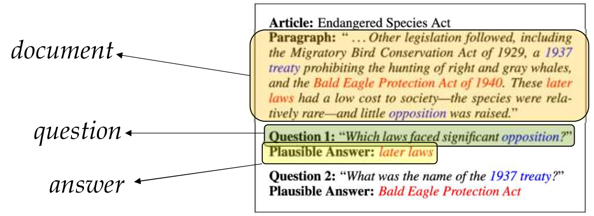
$P(y|x^1, \ldots, x^m) = softmax(h_l^m W_y)$



Fine-tuning approach : Open-AI GPT

❖ There is one problem in Second Stage …

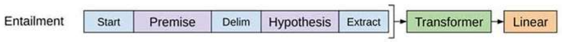
- Input form of the Open-AI GPT looks like
  - $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$,
    ex) {'I', 'am', 'a', 'student', 'I', 'like', 'studying', … }
- In task-specific task such as question answering, textual entailment
  - They have structured inputs
    ex) {document, question, answers}



Fine-tuning approach : Open-AI GPT

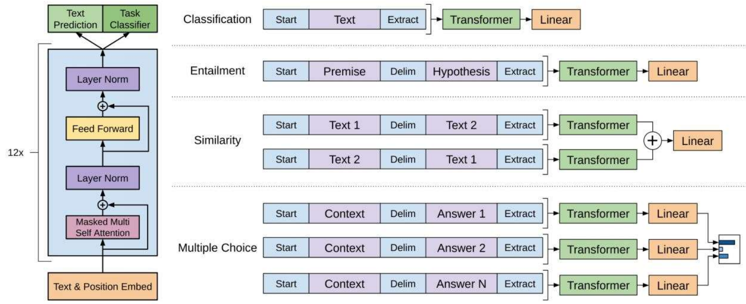
❖ There is one problem in Second Stage … (Cont'd)

- How to align those structured inputs so as to fine-tune with the structured inputs in a pre-trained input manner?
  - Fit **form of the inputs** to **the model**
  - 'Start,' 'Delim,' 'Extract' tokens !!
  - Ex) Entailment task, input has structured form of Premise and Entailment
  - Align the input as below
- 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



Fine-tuning approach : Open-AI GPT

❖ Second Stage, Supervised fine-tuning (Cont'd)



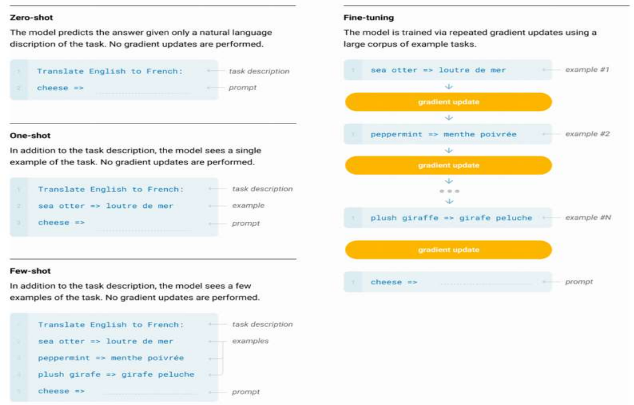
Open-AI GPT 3

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



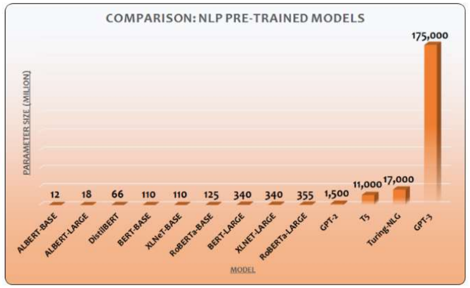

Open-AI GPT 3

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



Open-AI GPT 3

- **Model: GPT-2**와 동일한 구조
  - 파라미터 수 증가 **(175B** 파라미터**)**
- 데이터 소개
  - **45TB**나 되는 **150Billion Token (500GB** 전처리된 텍스트**)**



Compartison: Size of paremeters between models.

Open-AI GPT 3

- 문장 생성 및 **Cloze** 퀴즈 맞추기 태스크에 대한 성능

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

Alice was friends with Bob. Alice went to visit her friend ______. → Bob

George bought some baseball equipment, a ball, a glove, and a ______. →

- **Translation**

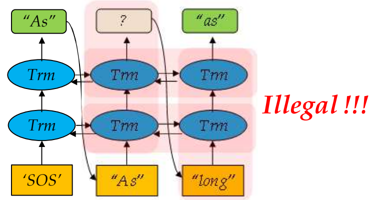
Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0[b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	37.5	34.9	28.3	35.2	35.2	33.1
mBART [LGG+20]	-	-	29.8	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	39.2	29.7	40.6	21.0	39.5

Fine-tuning approach : BERT

- **BERT (B**idirectional **E**ncoder **R**epresentations from **T**ransformers**)**
  - Paper published in NAACL 2019 by Google AI
  - "BERT:Pre-training of Deep Bidirectional Transformers for Language Understanding," Devlin et al. 2019, NAACL.
  - Won Best Long Paper

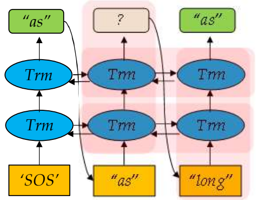
Fine-tuning approach : BERT

- **BERT (B**idirectional **E**ncoder **R**epresentations from **T**ransformers**)**
  - 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



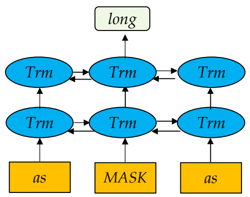
Fine-tuning approach : BERT

- **BERT (B**idirectional **E**ncoder **R**epresentations from **T**ransformers**)**
  - **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



Illegally injecting future label information to LM.

vs.

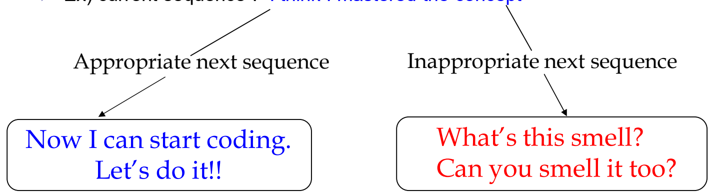


Masked language model task

Fine-tuning approach : BERT

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

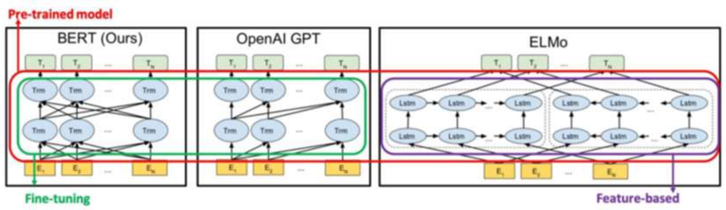
- **Next Sentence Prediction task?**
  - Guessing appropriate sequence after which follows
  - Ex) current sequence : "I think I mastered the concept"



Fine-tuning approach : BERT

❖ Overall Architecture

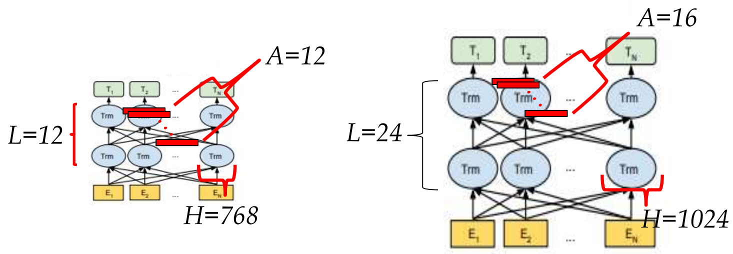
- ➢ Multi-layer bidirectional Transformer Encoder
  - ▪ Transformer is now able to refers to the right-to-left context due to the changed train objectives
- ➢ Task specific layer on top of the model



Fine-tuning approach : BERT

❖ Overall Architecture

- ➢ 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

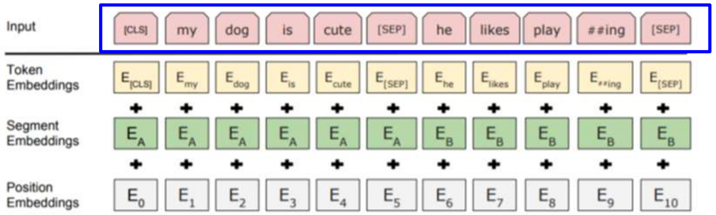


$\text{BERT}_{\text{BASE}}$ $\text{BERT}_{\text{LARGE}}$

Fine-tuning approach : BERT

❖ Overall Procedure

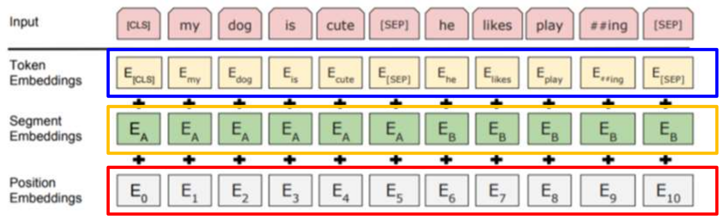
- ➢ How to construct an input?
  - ▪ [CLS] + sentence A + [SEP] + sentence B
  - ▪ Just like 'start', 'delim' tokens : 'CLS', 'SEP' tokens
- ➢ Input example
  - ▪ Ex) ['CLS', 'my', 'dog', 'is', 'cute', 'SEP', 'he', 'likes', 'playing', 'SEP']



Fine-tuning approach : BERT

❖ Overall Procedure (Cont'd)

- ➢ Input is represented and fed to the model summing
  - ▪ 1) WordPiece embeddings (Wu et al. 2016)
  - ▪ 2) Segment embeddings
  - ▪ 3) Learned positional embeddings

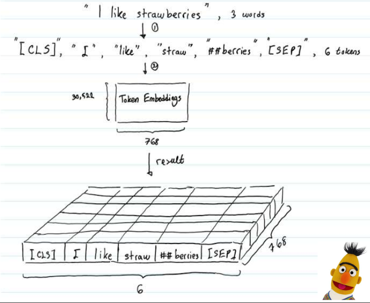


Fine-tuning approach : BERT

❖ 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

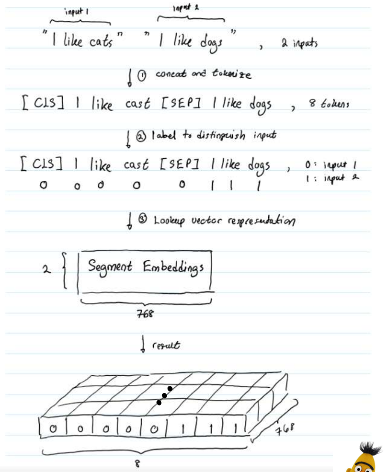


Fine-tuning approach : BERT

❖ Overall Procedure (Cont'd)

➢ 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



Fine-tuning approach : BERT

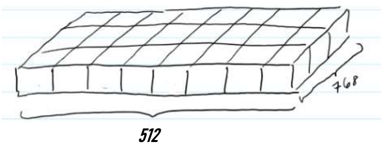
❖ Overall Procedure (Cont'd)

➢ 2) Positional Embeddings

> *I think, therefore I am*

the first "I" should not have the same vector representation as the second "I".

- BERT is designed to process input sequences of up to length 512
- BERT learns a vector representation for each position



Architecture of Transformer

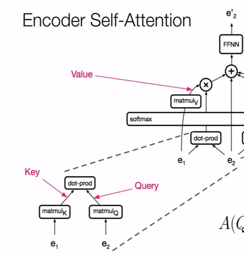
❖ Multihead attention in Transformer

➢ Sentence is about 'who,' 'did what,' and 'to whom'

➢ In CNN, different filters learn the concept of 'who,' 'did what,' and 'to whom.'

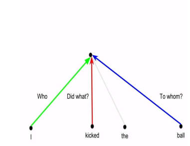
➢ Self-attention can't pick out different information from different places

- It's just a linear combination everywhere

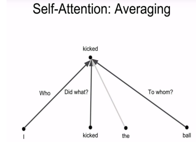


$$A(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Convolutions



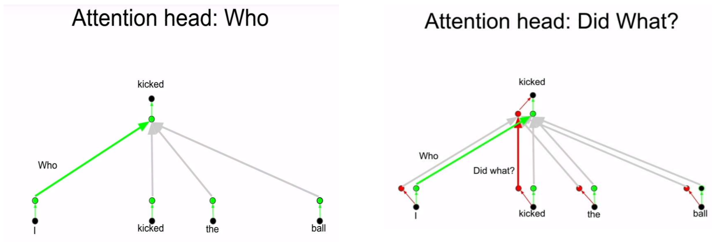
Self-Attention: Averaging



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

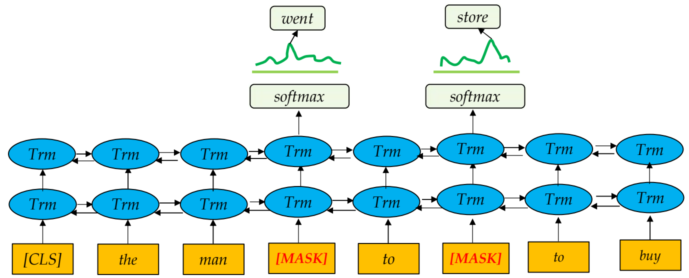


Fine-tuning approach : BERT

❖ **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



Fine-tuning approach : BERT

❖ **Task #1 : Masked LM (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

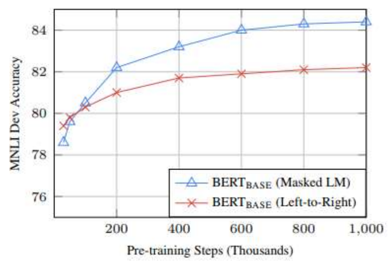
- 80% of the time ($0.8 \times \alpha$%) : MASK
  - ✓ "my dog is [MASK]"
- 10% of the time ($0.1 \times \alpha$%) : Replace with a random word
  - ✓ "my dog is apple"
- 10% of the time ($0.1 \times \alpha$%) : Keep the word unchanged
  - ✓ "my dog is hairy"

Fine-tuning approach : BERT

❖ **Task #1 : Masked LM (Cont'd)**

➢ Two downsides of this approach

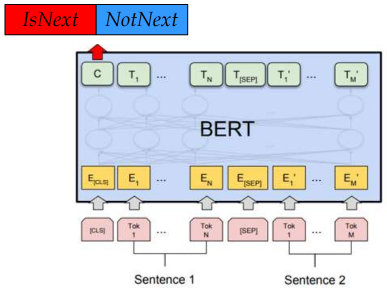
- 2nd , only 15% of tokens are predicted in each batch
  - ✓ which suggests that more pre-training steps may be required for the model to converge
  - ✓ Left-to-right model predicts every token so it converges faster
- However, empirical improvements of the MLM model far outweigh the increased training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



Fine-tuning approach : BERT

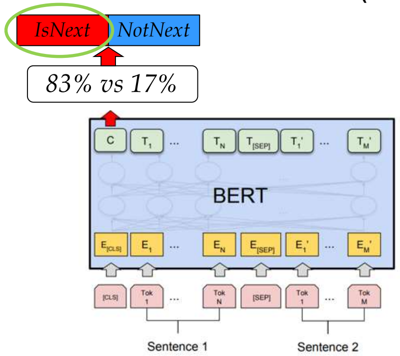
❖ **Task #2 : Next Sentence Prediction (NSP)**

- 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



Fine-tuning approach : BERT

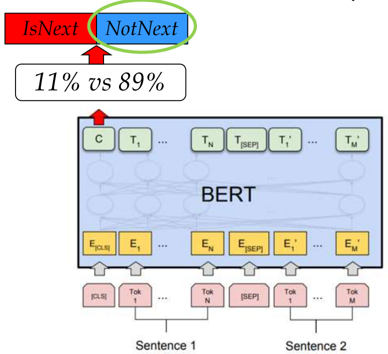
❖ **Task #2 : Next Sentence Prediction (NSP) (Cont'd)**



[CLS] the man went to [MASK] store ***[SEP]*** *he bought a [MASK] chocolate milk [SEP]*

Fine-tuning approach : BERT

❖ **Task #2 : Next Sentence Prediction (NSP) (Cont'd)**



[CLS] the man went to [MASK] store ***[SEP]*** *starry [MASK] night [SEP]*

Fine-tuning approach : BERT

❖ **Task #2 : Next Sentence Prediction (NSP) (Cont'd)**

- Effect of training with the task of NSP

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

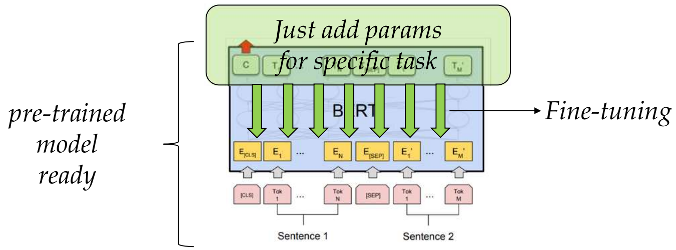
  - No NSP : trained without the NSP task
  - LTR & No NSP : trained without the NSP task + only left-to-right LSTM

- 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 !**

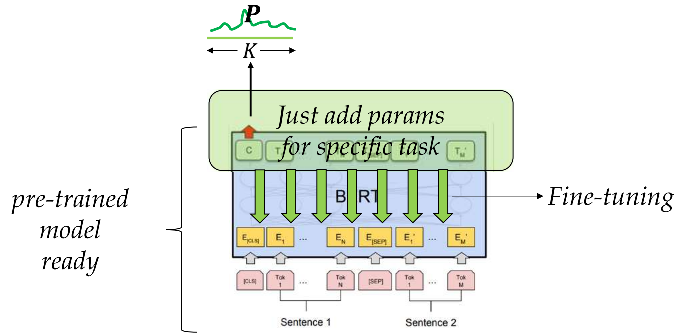
- No need to construct another model for another task
- Just add the output layer parts !



Fine-tuning approach : BERT

❖ **Fine-tuning**

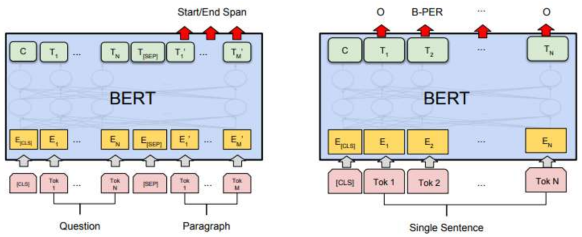
- [CLS] embedding ($\boldsymbol{C} \in \mathbb{R}^{H}$) is mostly used for fine-tuning task
- Only new parameter ($\boldsymbol{W} \in \mathbb{R}^{K \times H}$) for classification layer
  - K is the # of classifier lables, ex. 2 for ['IsNext', 'NotNext']
- Label probabilities ($\boldsymbol{P} \in \mathbb{R}^{K} = \mathrm{softmax}(CW^{T})$)



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]



(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

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

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\mathrm{BERT}_{\mathrm{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\mathrm{BERT}_{\mathrm{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

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.[8] 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)**

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

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

57

Fine-tuning approach : BERT

❖ **Result SQuAD 2.0**

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

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$\text{BERT}_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

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

58

References

❖ **Chris McCormick, "Word2Vec Tutorial – The Skip-Gram Model" :** ***http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/***

❖ ***Mikolov et al. 2013, "Efficient Estimation of Word Representations in Vector Space")***

❖ ***G.E. Hinton, J.L. McClelland, D.E. Rumelhart. Distributed representations. In: Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations, MIT Press, 1986.***

❖ ***Distributed Representations of Words and Phrases and their Compositionality (Mikolov et al. 2013)***

❖ ***LM picture :*** ***https://upload.wikimedia.org/wikipedia/commons/d/dd/CBOW_eta_Skipgram.png***

❖ ***Transfer Learning Tutorial by Sebastian Ruder, Matthew Peters, Swabha Swayamdipta, Thomas Wolf :*** ***https://docs.google.com/presentation/d/1fIhGikFPnb7G5kr58OvYC3GN4io7MznnM0aAgadvJfc/edit#slide=id.g58bdd596a1_0_0***

❖ **Semi-supervised Sequence Learning - NIPS Proceedings,** ***Dai and Le, 2015***

❖ ***Semi-supervised sequence tagging with bidirectional language models, Peters et al. 2017***

❖ ***Deep contextualized word representations, Peters et al. 2018***

59

References

❖ **ELMo explanation blog : https://www.slideshare.net/shuntaroy/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 Classfication (Jeremy Howard and Sebastian Ruder, 2018)**

❖ **BERT:Pre-training of Deep Bidirectional Transformers for Language Understanding," Devlin et al. 2019, NAACL**

❖ **Improving Language Understanding with Unsupervised Learning (Radford et al. 2018)**

❖ **Attention Is All You Need, Vaswani et al. 2017**

❖ **Stanford CS224N: NLP with Deep Learning | Winter 2019 | Lecture 14 – Transformers and Self-Attention**

❖ **https://www.youtube.com/watch?v=5vcj8kSwBCY&t=811s**

❖ **OpenAI-GPT figure : https://www.topbots.com/generalized-language-models-ulmfit-openai-gpt/**

❖ **Input representation of BERT : https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a**

60

Thank you for your attention!

고 영 중 (Ko, Youngjoong)

nlplab.skku.edu

61