自然语言处理: BERT、XLNET的原理

ELMO/GPT/BERT/XLNET

Language Models

- **Language model** is a **probability distribution** over sequences of words.

$$P(w_1, w_2, ..., w_m) = P(w_1)\ P(w_2|w_1)\ P(w_3|w_1, w_2)$$
$$\ldots P(w_i \mid w_1, w_2, ..., w_{i-1}) \ldots P(w_m \mid w_1, w_2, ..., w_{m-1})$$

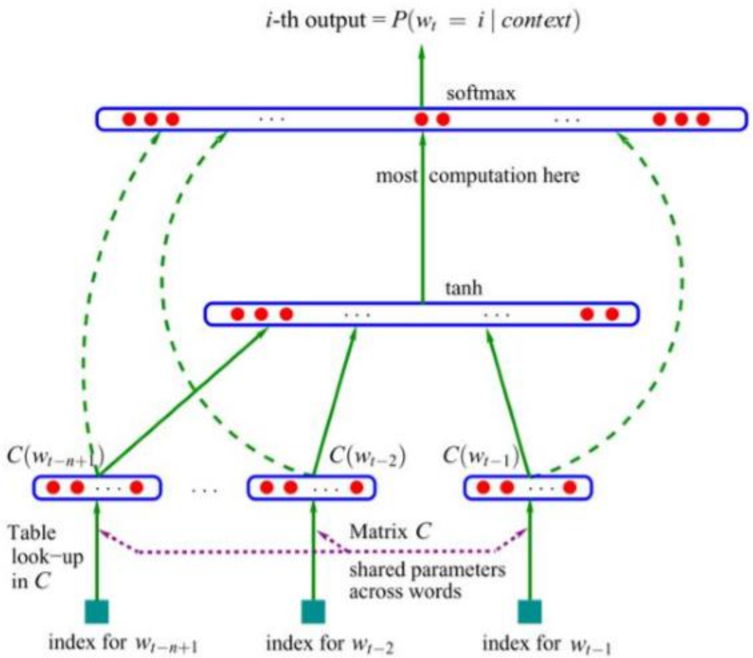
  - *n*-Gram Models (unigram, bigram, trigram)

$$P(w_i \mid w_1, w_2, ..., w_{i-1}) \approx P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

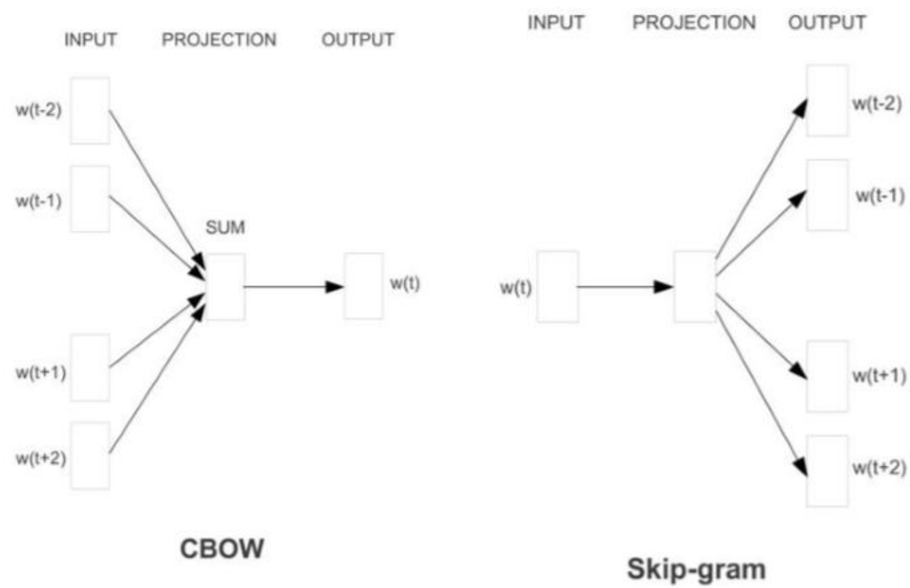
$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1})}$$

  - Neural Network Language Model (NNLM)

NNLM

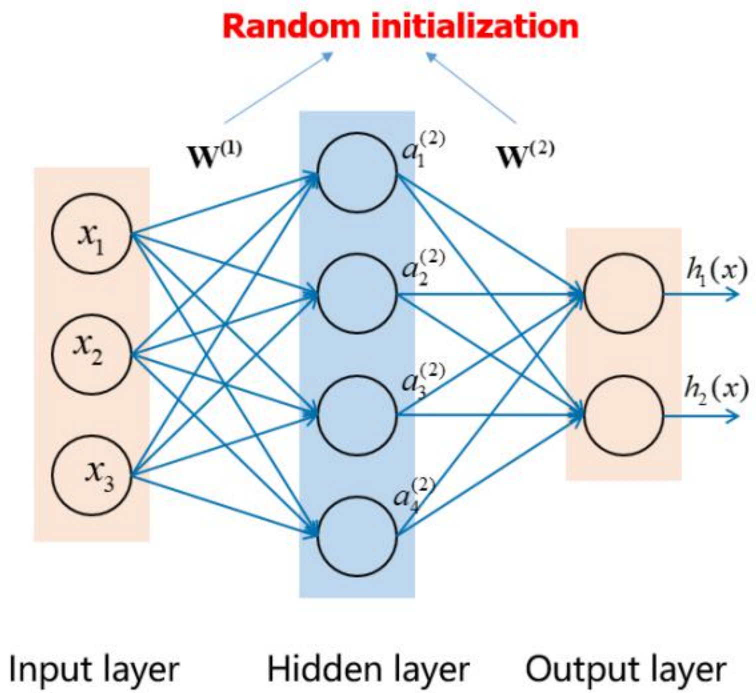


Word2vec(2013)



Pre-training

- Stacked Autoencoders (SAE)
- Word Embedding
- Transfer learning



ELMO: Deep Contextualized Word Representations

Motivation

- Pre-trained word representations should model both:
  - Complex characteristics of word use (e.g., **syntax and semantics**)
  - How these uses vary across linguistic contexts (i.e., to model **polysemy**)
- Traditional word embedding
  - These approaches for learning word vectors only allow a **single context-independent** representation for each word

Example:

1. Jobs was the CEO of **apple**.
2. He finally ate the **apple**.

Contribution

- Leveraging **Language Modeling** to get **pre-trained contextualized representation.**
- ELMo: **E**mbeddings from **L**anguage **Mo**dels
- Highlight:
  - The higher-level LSTM **internal states** capture **context-dependent** aspects of word meaning.
  - These representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems.

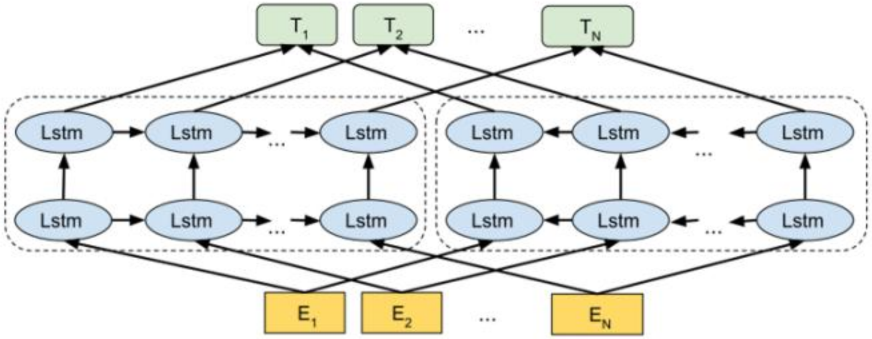
Bidirectional language models

- A forward LM

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1})$$

- A backward LM

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N)$$



- Jointly maximize the log likelihood of the forward and backward directions

$$\sum_{k=1}^{N} \left(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right)$$

Embeddings from Language Models

- ELMo is a task specific combination of the **intermediate layer representations in the biLM**.
- For k-th token, L-layer bi-directional Language Models computes 2L+1 representations

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

- For a specific down-stream task, ELMo would learn a weight to combine these representations (In the simplest case, ELMo just selects the top layer $E(R_k) = \mathbf{h}_{k,L}^{LM}$)

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

scalar parameter   softmax-normalized weights

Transformer

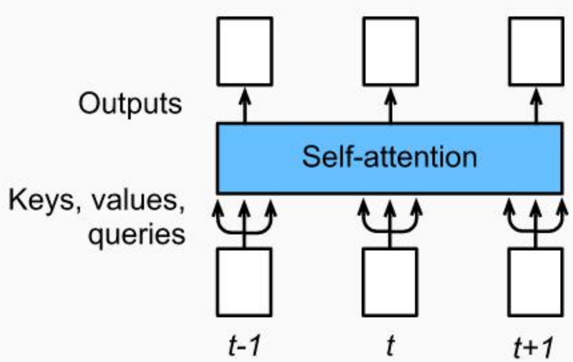


Fig. 9.3.1 Self-attention architecture.

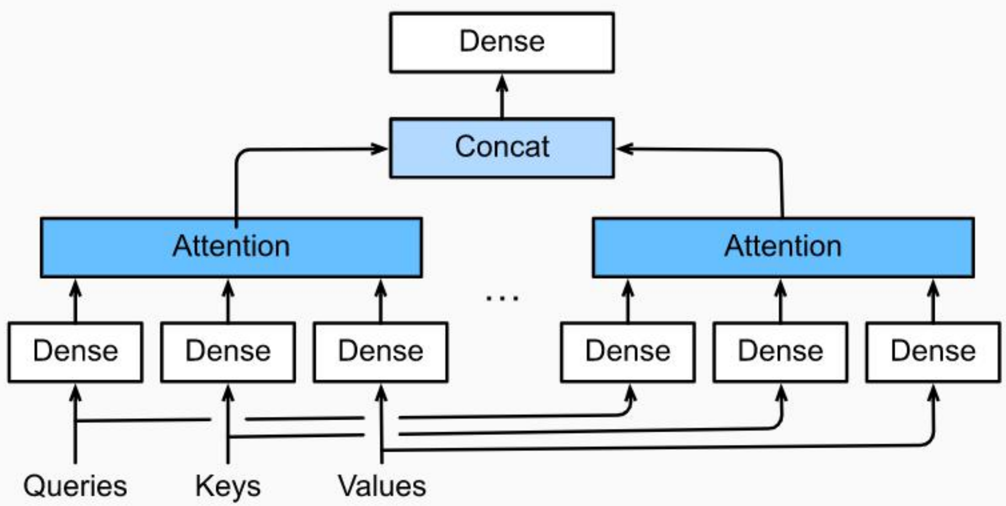
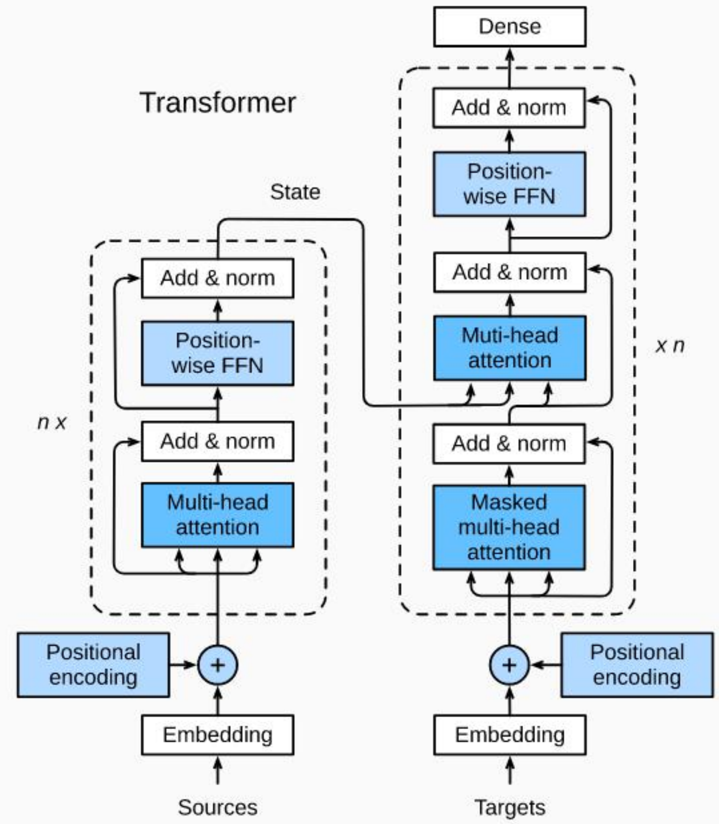


Fig. 9.3.3 Multi-head attention



GPT: Improving Language Understanding by Generative Pre-Training

Contribution

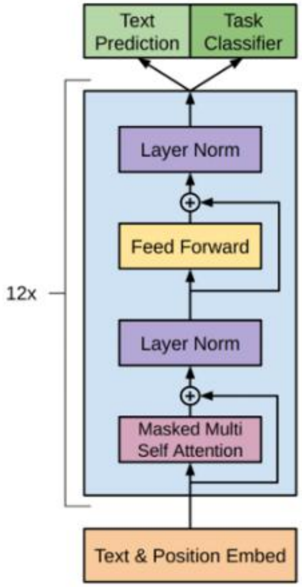
- Their goal is to learn **a universal representation** that transfers with little adaptation to a wide range of tasks.
- Highlight:
  - Use **transformer networks** instead of LSTM to achieve better capture long-term linguistic structure.
  - Include **auxiliary training objectives** (e.g. language modeling) in addition to the task objective when fine-tuning.
  - Demonstrate the effectiveness of the approach on a wider range of tasks. (significantly improving upon the state of the art in **9 out of the 12 tasks** studied)

Unsupervised pre-training

- Use a **standard language modeling objective** to maximize the following likelihood

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

- **A multi-layer Transformer decoder** for the language model



$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

U is token index matrix
We is embedding matrix

Supervised fine-tuning

- The final transformer block's activation is fed into an added linear output layer

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

classfy

- objective function

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m)$$

- We additionally found that including **language modeling as an auxiliary objective** to the fine-tuning helped learning by (a) improving generalization of the supervised model, and (b) accelerating convergence.

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Motivation

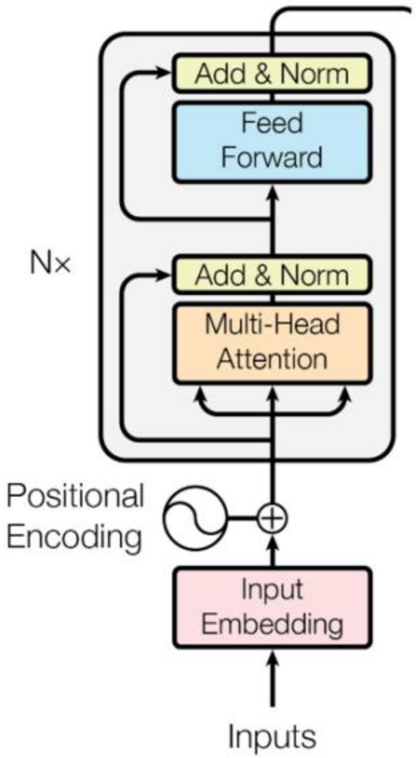
- Pre-trained language representations
  - **Feature-based**: ELMo
  - **Fine-tuning**: OpenAI GPT
- Both approaches share the same objective function during pre-training, where they use **unidirectional language models** to learn general language representations

Contributions

- BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.
  - We demonstrate the importance of bidirectional pre-training for language representations.
  - Use Transformer encoder as the LM and a new pre-training objective: the "masked language model" (MLM).
  - Introduce a "next sentence prediction" task that jointly pre-trains text-pair representations.
  - They show that pre-trained representations eliminate the needs of many heavily-engineered task-specific architectures. BERT advances the state-of-the-art for eleven NLP tasks.

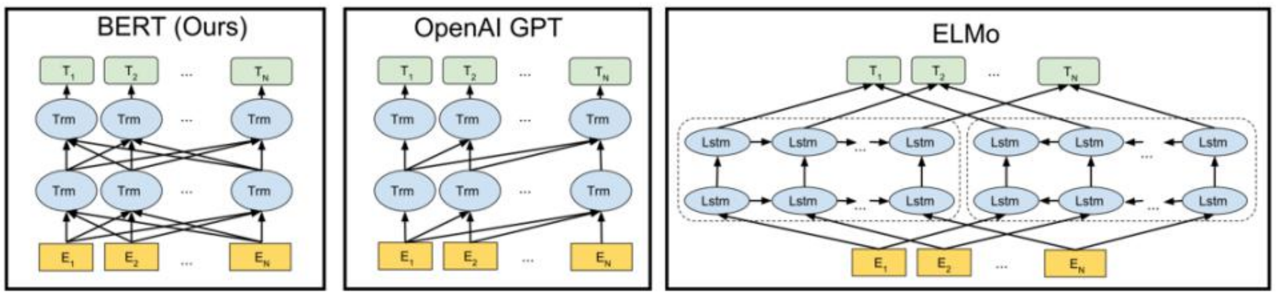
Model Architecture

- BERT's model architecture is a **multi-layer bidirectional Transformer encoder**.
  - $BERT_{BASE}$: L=12, H=768, A=12, Total Parameters=110M. (It was chosen to have an identical model size as OpenAI GPT for comparison purposes.)
  - $BERT_{LARGE}$: L=24, H=1024, A=16, Total Parameters=340M.



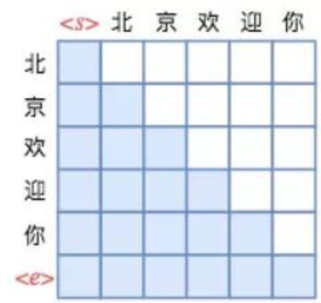
Model Architecture

- Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

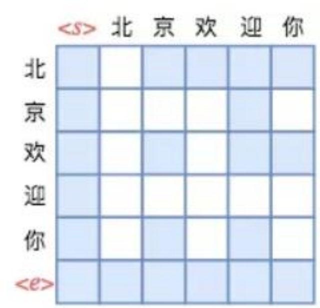


difference: mask

- **BERT** uses a bidirectional Transformer.
- **OpenAI GPT** uses a left-to-right Transformer.
- **ELMo** uses the concatenation of independently trained left-to-right and right- to-left LSTM.

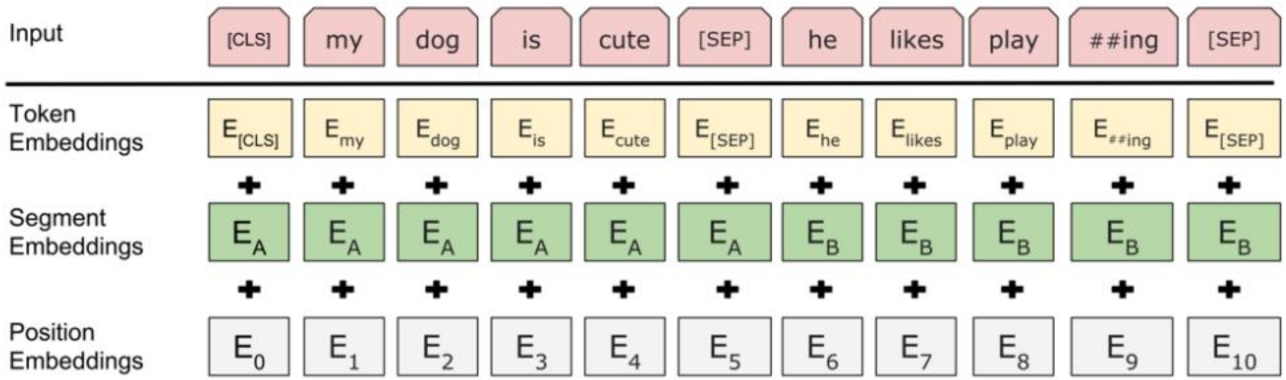


正向语言模型的Mask



乱序语言模型的Mask

Input Representation



- WordPiece embeddings with a 30,000 token vocabulary.
- Learned positional embeddings with supported sequence lengths up to 512 tokens.
- The first token of every sequence is always the special classification embedding ([CLS]).
- Sentence pairs are packed together into a single sequence.

Pre-training Task1: Masked LM (MLM)

- Masking some percentage of the input tokens at random, and then predicting only those masked tokens.
- The MLM objective allows the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer.
- Downsides:
  - Create a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.
  - Only 15% of tokens are predicted in each batch, which suggests that more pre-training steps may be required for the model to converge.

Pre-training Task1: Masked LM

- The [MASK] token is never seen during fine-tuning.
- To mitigate this, Rather than always replacing the chosen words with [MASK], the data generator will do the following:
  - 80% of the time: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]
  - 10% of the time: Replace the word with a random word, e.g., my dog is hairy → my dog is apple
  - 10% of the time: Keep the word unchanged, e.g., my dog is hairy → my dog is hairy.

- If we used [MASK] 100% of the time the model wouldn't necessarily produce good token representations for non-masked words. The non-masked tokens were still used for context, but the model was optimized for predicting masked words.
- If we used [MASK] 90% of the time and random words 10% of the time, this would teach the model that the observed word is *never* correct.
- If we used [MASK] 90% of the time and kept the same word 10% of the time, then the model could just trivially copy the non-contextual embedding.

Pre-training Task2: Next Sentence Prediction

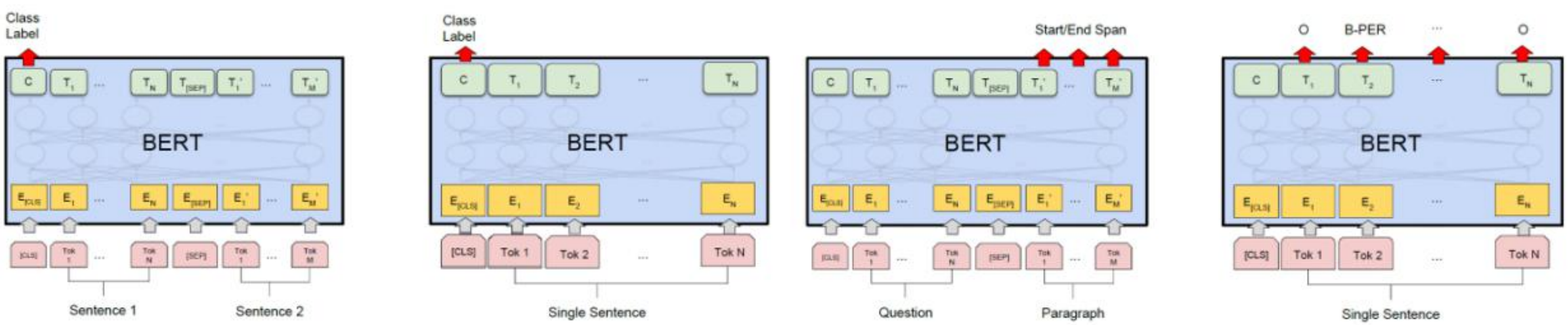
- In order to train a model that understands sentence relationships, they pre-train a binarized next sentence prediction task.
- When choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A, and 50% of the time it is a random sentence from the corpus.

Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

Fine-tuning Procedure

- All of the parameters are fine-tuned jointly to maximize the log-probability of the correct label.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

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

General Language Understanding Evaluation (GLUE) Test results

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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

- It is interesting to observe that $\text{BERT}_{\text{LARGE}}$ significantly outperforms $\text{BERT}_{\text{BASE}}$ across all tasks, even those with very little training data.

Feature-based Approach with BERT

Layers	Dev F1
Finetune All	96.4
First Layer (Embeddings)	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Sum All 12 Layers	95.5

Table 7: Ablation using BERT with a feature-based approach on CoNLL-2003 NER. The activations from the specified layers are combined and fed into a two-layer BiLSTM, without backpropagation to BERT.

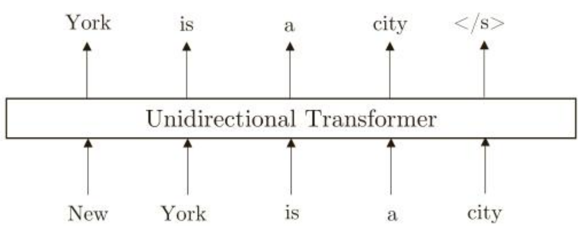
- This demonstrates that BERT is effective for both the fine-tuning and feature-based approaches

XLNET: Generalized Autoregressive Pretraining for Language Understanding

AR、AE language modeling

Two Objectives for Pretraining

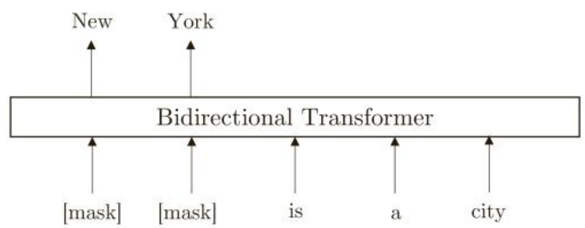
Auto-regressive (AR) language modeling



$$\log p(\mathbf{x}) = \sum_{t=1}^{T} \log p(x_t | \mathbf{x}_{<t})$$

Not able to model bidirectional context. ☹

(Denoising) Auto-encoding (AE)



$$\log p(\bar{\mathbf{x}} | \hat{\mathbf{x}}) = \sum_{t=1}^{T} \text{mask}_t \log p(x_t | \hat{\mathbf{x}})$$

Predicted tokens are independent of each other. ☹
[mask] is not used during finetuning. ☹

New Objective: Permutation Language Modeling

- Sample a factorization order
- Determine the attention masks based on the order
- Optimize a standard language modeling objective

$$\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T}\log p(x_{z_t}|\mathbf{x}_{\mathbf{z}_{<t}})\right]$$

- Benefits:
  - Autoregressive, avoiding disadvantages of AE
  - Able to model bidirectional context

Examples

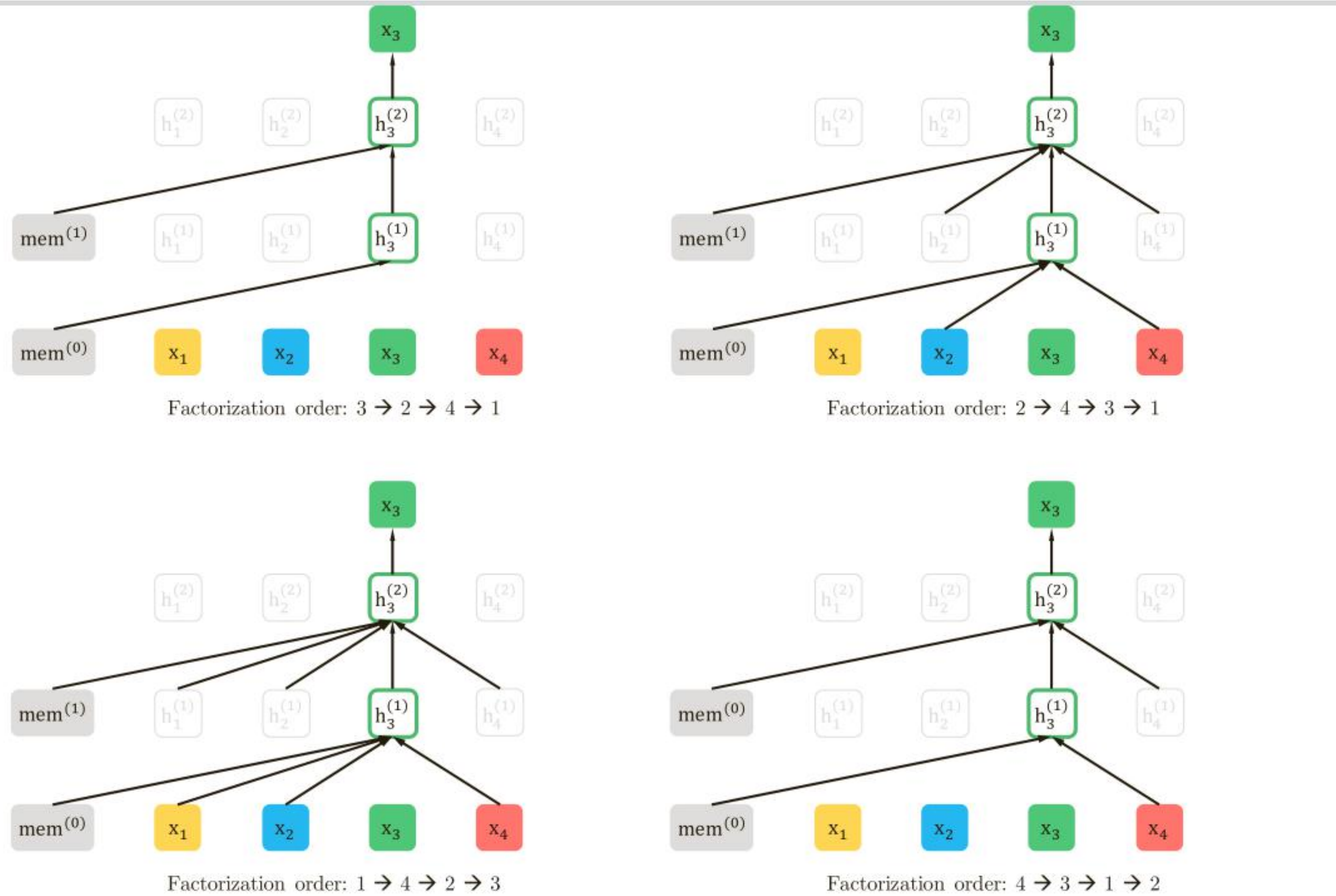
Factorization order: New York is a city

  P(New York is a city)
= P(New) * P(York | New) * P(is | New York) * P(a | New York is) * P(city | New York is a)

Factorization order: city a is New York

  P(New York is a city)
= P(city) * P(a | city) * P(is | city a) * P(New | city a is) * P(York | city a is New)

Sequence order is not shuffled.
Attention masks are changed to reflect factorization order.



x3
h1(2) h2(2) h3(2) h4(2)
mem(1)
h1(1) h2(1) h3(1) h4(1)
mem(0)
x1 x2 x3 x4
Factorization order: 3 → 2 → 4 → 1
Factorization order: 2 → 4 → 3 → 1
Factorization order: 1 → 4 → 2 → 3
Factorization order: 4 → 3 → 1 → 2

Comparing XLNet and BERT Objectives

BERT objective (auto-encoding)

$$\mathcal{J}_{\text{BERT}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{is a city})$$

New and York are independent. ☹

depend on
how to sample

XLNet objective (auto-regressive)

$$\mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \boxed{\text{New},} \text{ is a city})$$

$$\text{or } \mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \boxed{\text{York},} \text{ is a city}) + \log p(\text{York} \mid \text{is a city})$$

Able to model dependency between New and York. ☺
Able to model bidirectional context. ☺
Factorize the joint probability using a product rule that holds universally.

Reparameterization

Standard Parameterization

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{\mathbf{z}_{<t}}) = \frac{e(x)^\top h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})}{\sum_{x'} e(x')^\top h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})}$$

h does not contain the position of the target.

$$p(X_3 = \text{is} \mid \text{New York}) = p(X_4 = \text{is} \mid \text{New York}) = p(X_5 = \text{is} \mid \text{New York})$$

Reduced to predicting a bag of words.

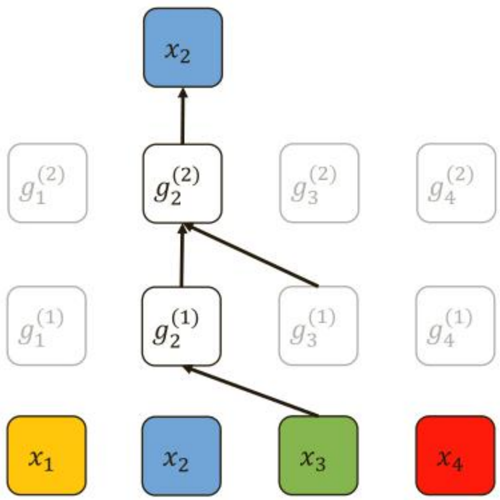
Solution: condition the distribution on the position.

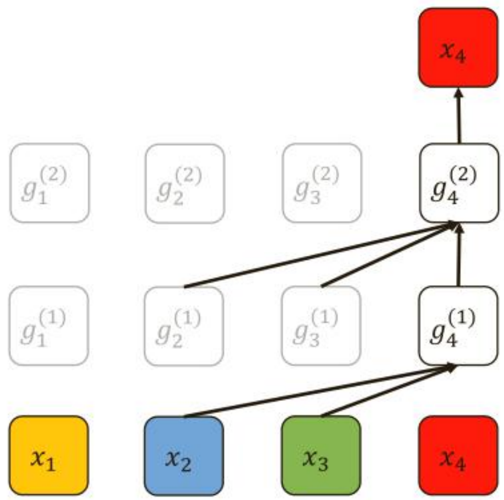
$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{\mathbf{z}_{<t}}) = \frac{e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, \boxed{z_t})}{\sum_{x'} e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, \boxed{z_t})}$$

"Stand at" z_t and predict self

How to formulate features g

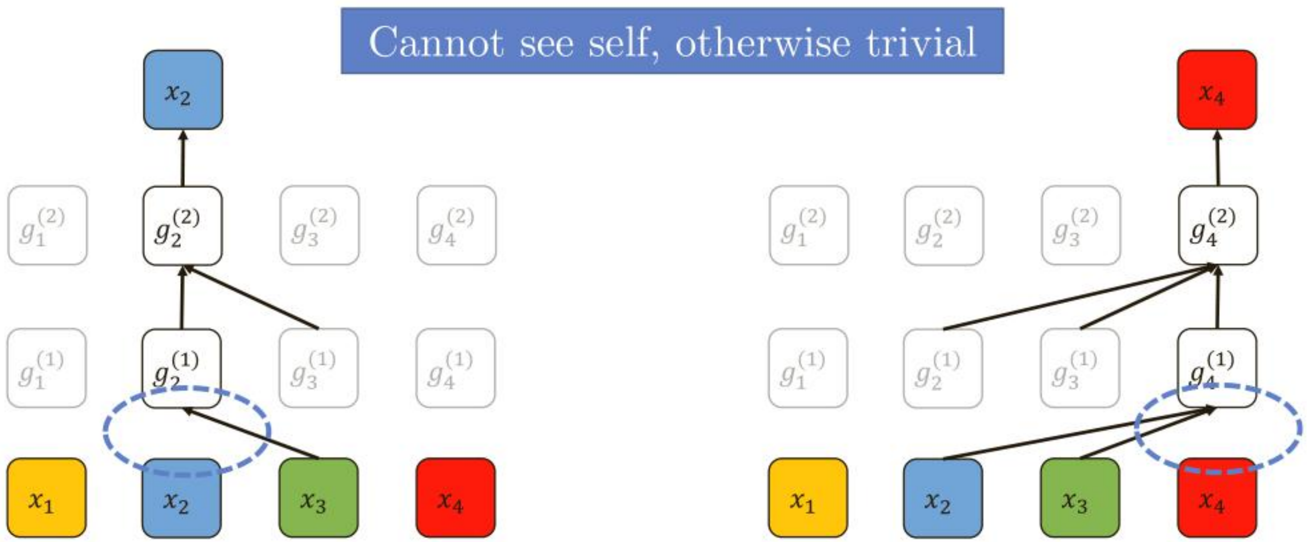
Let $g_i^{(l)}$ denote the feature of the i-th token on layer l
Suppose the factorization order is 3 2 4 1





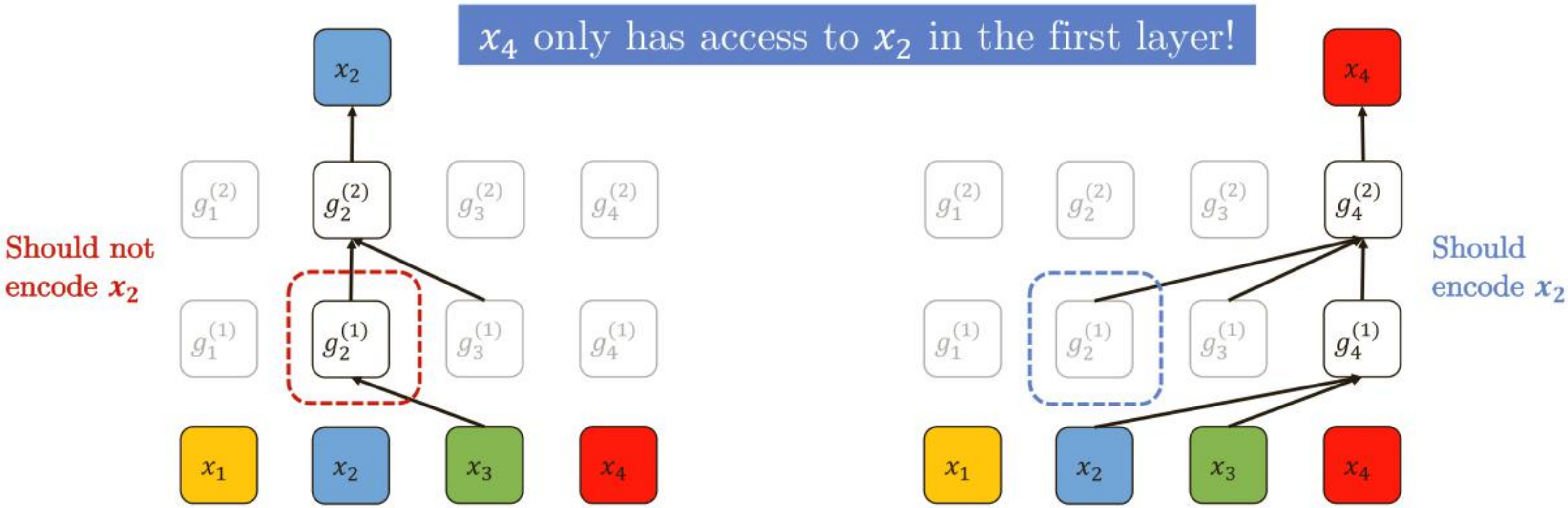
How to formulate features g

Let $g_i^{(l)}$ denote the feature of the i-th token on layer l
Suppose the factorization order is 3 2 4 1



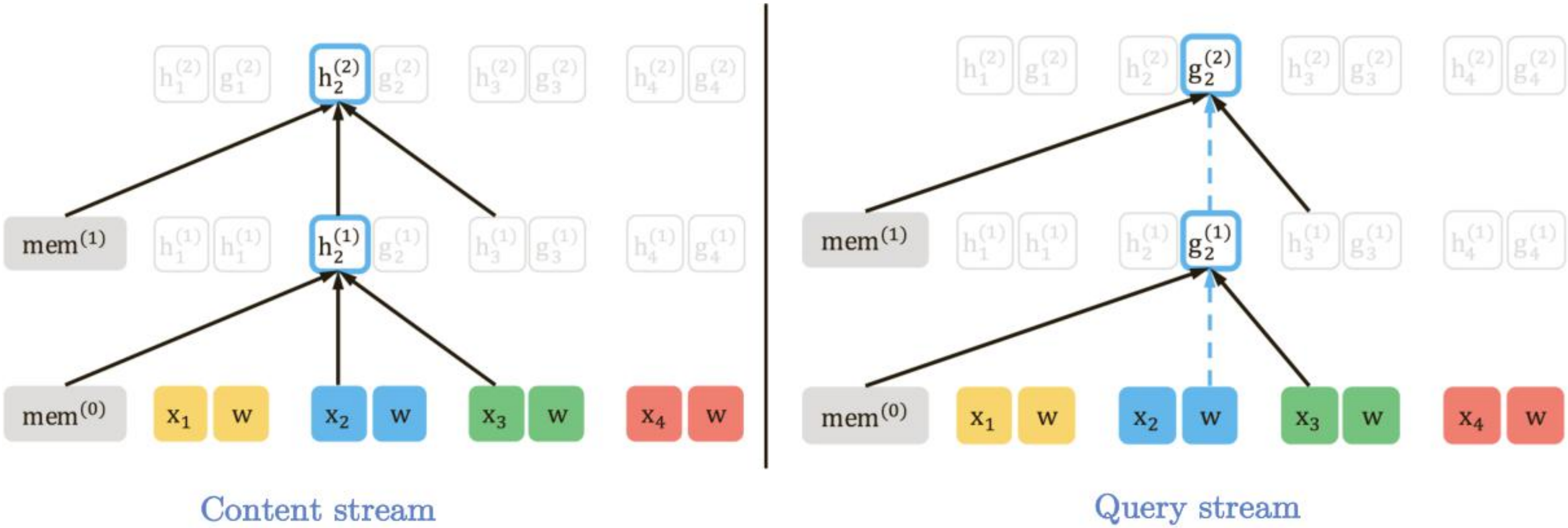
How to formulate features g

Let $g_i^{(l)}$ denote the feature of the i-th token on layer l
Suppose the factorization order is 3 2 4 1



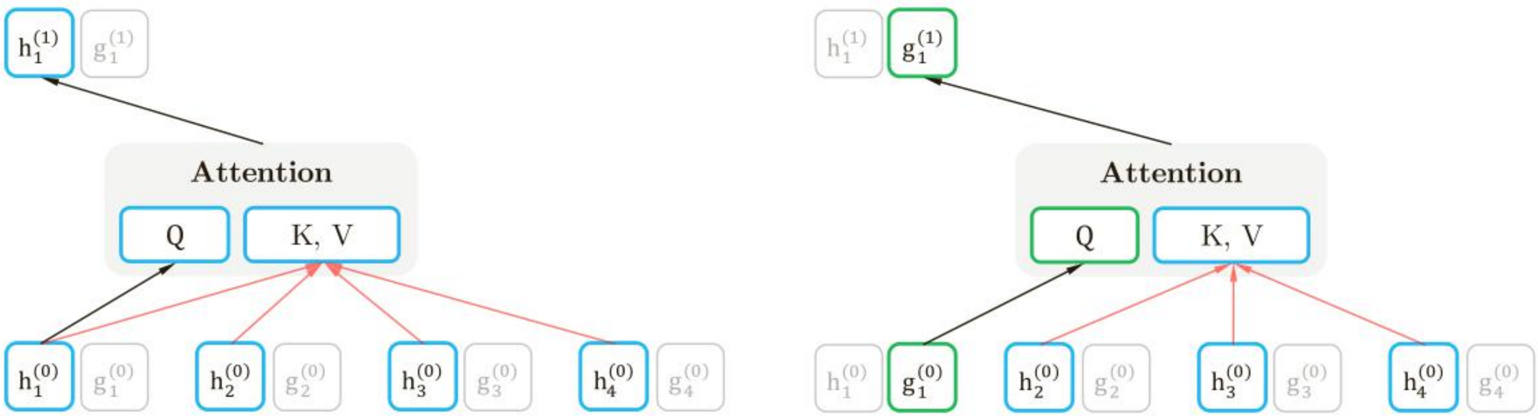
Two-Stream Attention

- Factorization order: 3, 2, 4, 1



encoder-----Content stream
decoder-----Query strteam

Two-Stream Attention



$$g_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = g_{z_t}^{(m-1)}, \textbf{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta), \quad \text{(query stream: use } z_t \text{ but cannot see } x_{z_t}\text{)}$$

$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \textbf{KV} = \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}\text{).}$$

At first layer, h is the word embeddings, and g is a trainable parameter.
Only h is used during finetuning. The last g is used for optimizing the LM loss.

Use g to predict every position content.
h only use in pre-train.

Summarizing XLNet

Challenges

Independence assumption in BERT

Standard parameterization is reduced to bag-of-words

Contradiction for predicting both self and others

Solutions

Permutation language modeling

Reparameterization with positions

Two-stream attention

参考文献

- [1] Peters, M. E. et al. Deep contextualized word representations. naacl (2018).
- [2] Radford, A. & Salimans, T. Improving Language Understanding by Generative Pre-Training.
- (2018).
- [3] Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional
- Transformers for Language Understanding. (2018).
- [4] Peters, M. E., Ammar, W., Bhagavatula, C. & Power, R. Semi-supervised sequence tagging
- with bidirectional language models. Acl (2017).

- https://www.cnblogs.com/robert-dlut/p/9824346.html
- https://github.com/AIProCon/AIProCon/blob/master/PPT/%E8%87%AA%E7%84%B6%E8%AF%AD%E8%A8%80%E5%A4%84%E7%90%86%E8%AE%BA%E5%9D%9B/%E3%80%8A%E8%87%AA%E7%84%B6%E8%AF%AD%E8%A8%80%E7%90%86%E8%A7%A3%E6%A8%A1%E5%9E%8BXLNet%E3%80%8B%E6%9D%A8%E6%A4%8D%E9%BA%9F.pdf