The Disaster Tweets - Text Classification

CAI Shizhan, SONG Wenxin

A Kaggle Competition

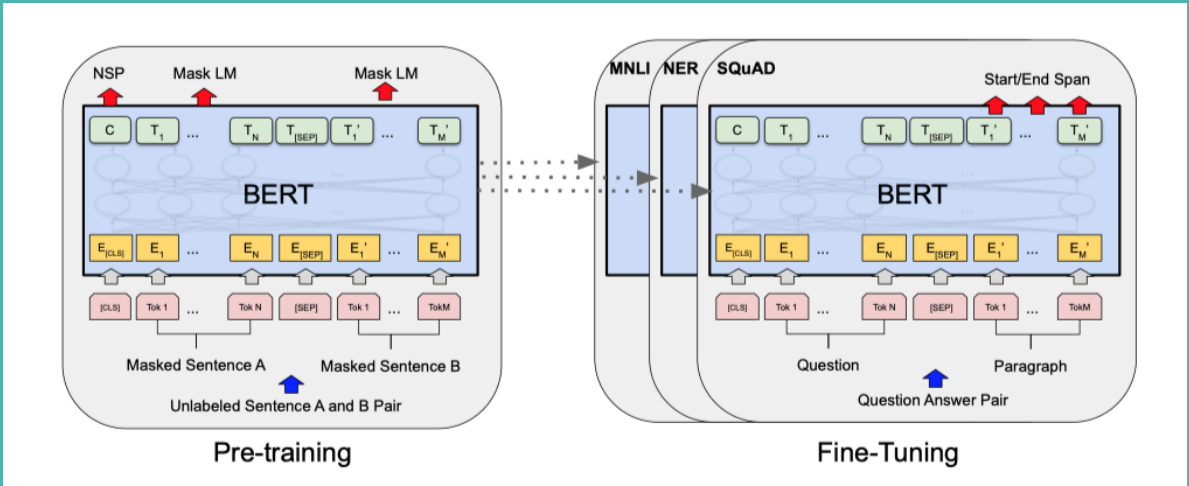
Natural Language Processing with Disaster Tweets
Predict which Tweets are about real disasters and which ones are not

Dataset

id	keyword	location	text	target
1			Our Deeds are the Reason of this #earthquake May ALLAH Forgive u	1
4			Forest fire near La Ronge Sask. Canada	1
5			All residents asked to 'shelter in place' are being notified by officers.	1
6			13,000 people receive #wildfires evacuation orders in California	1
7			Just got sent this photo from Ruby #Alaska as smoke from #wildfires	1

1. BERT: Bidirectional Encoder Representation from Transformer

A cutting-edge model proposed in 2019



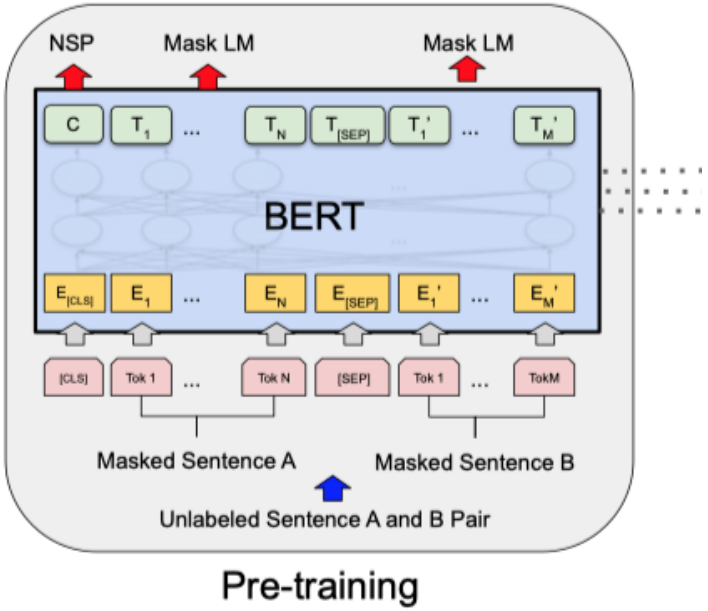
1.1 Pre-training

Task #1: Masked LM

- Randomly mask 15% of the tokens and calculate the loss on the masked tokens.

Task #2: Next Sentence Prediction (NSP)

- Choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A

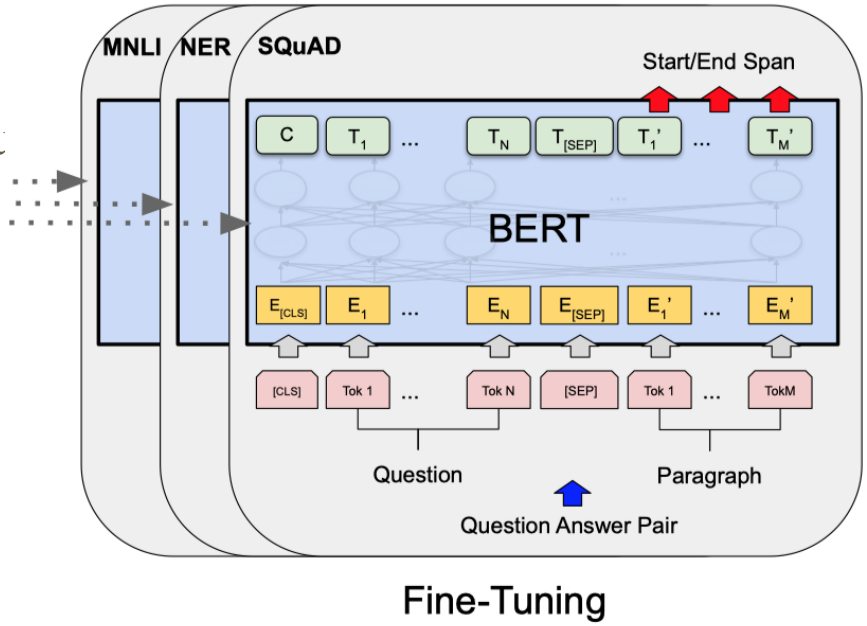


Pre-training

1.2 Fine-tuning

Tune hyperparamters for each task:

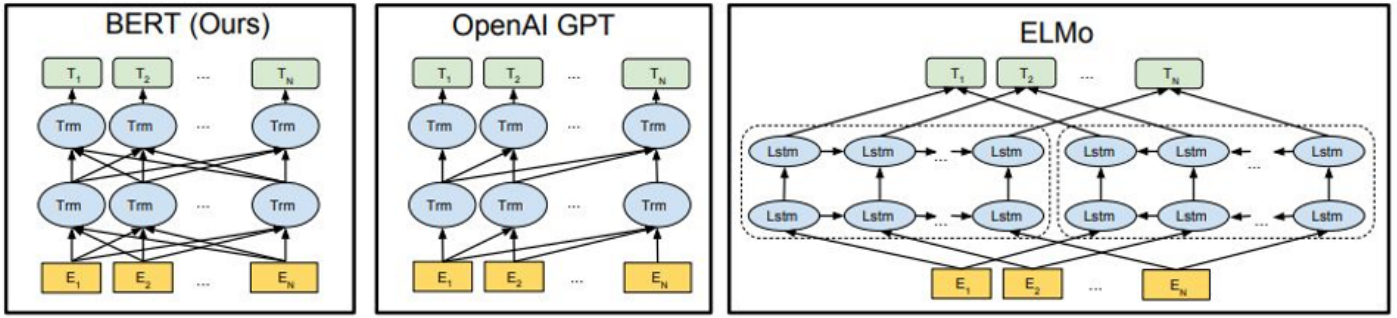
Batch size, Learning rate (Adam), Nu
of epochs...



Fine-Tuning

1.3 Comaprison

Outperformed many other popular models at that time:



OpenAI GPT: single directional

ELMo: Uses different objective functions.

ELMo: Uses $P(w_i|w_1, \dots w_{i-1})$ and $P(w_i|w_{i+1}, \dots w_n)$ as objective functions, train them separately

BERT: Uses $P(w_i|w_1, \dots, w_{i-1}, w_{i+1}, \dots w_n)$

2. RoBERTa: A Robustly Optimized BERT Pretraining Approach

Key differences:

(1) dynamically changing the masking pattern;

(2) removing the next sentence prediction objective;

(3) training the model longer, with bigger batches, over more data and longer sequence;

2.1 Static vs. Dynamic Masking

BERT: performed masking once during data preprocessing, resulting in a single static mask.

RoBERTa: Uses a method that mimics dynamic masking. Avoid using the same mask for every training.

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

2.2 Model Input Format and Next Sentence Prediction

SEGMENT- PAIR + NSP: Original input format used in BERT

SENTENCE - PAIR + NSP: Each input contains a pair of natural sentences, either sampled from a contiguous portion of one document or from separate documents.

FULL - SENTENCES: Each input is packed with full sentences sampled contiguously from one or more documents, such that the total length is at most 512 tokens.

DOC - SENTENCES: Inputs are constructed similarly to FULL - SENTENCES, except that they may not cross document boundaries.

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

2.3 Text Encoding

BERT: Uses a **character-level** BPE vocabulary of size 30K

RoBERTa: Uses a larger **byte-level** BPE vocabulary containing 50K subword units

Reason:

‘*Using bytes makes it possible to learn a subword vocabulary of a modest size (50K units) that can still encode any input text without introducing any “unknown” tokens.’*

2.4 Summary

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

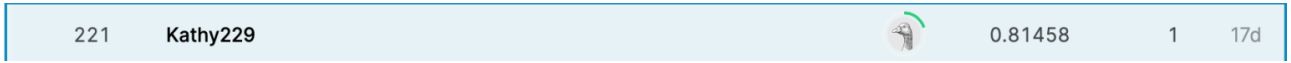
2.5 RoBERTa result on the dataset

- Accuracy on validation set: 0.80

- Result on testing set at Kaggle:

  F1 score: 0.81458,

  Ranking: 221



3. XLNet

One Flew Over the ~~Cuckoo's Nest~~ Sesame Street

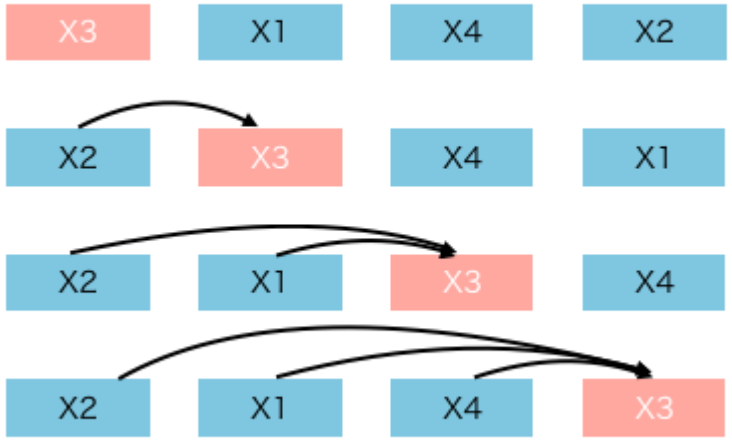
ELMo

BERT

XLNet

3.1 Autoregressive(AR)

The process of generating a sentence is as follows: generate the first word according to the probability distribution, then generate the second word according to the first word, and then generate the third word according to the first two words, ... until the entire sentence is generated



$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t}) = \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x_t)\right)}{\sum_{x'} \exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x')\right)}, \qquad (1)$$

3.2 Denoising Autoencoders(DAE)

Autoencoders are Neural Networks which are commonly used for feature selection and extraction. Denoising Autoencoders corrupts the data on purpose by randomly turning some of the input values to zero

$$\max_{\theta} \quad \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}}) = \sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x_t)\right)}{\sum_{x'} \exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x')\right)}, \qquad (2)$$

$$H_\theta(\mathbf{x}) = [H_\theta(\mathbf{x})_1, H_\theta(\mathbf{x})_2, \cdots, H_\theta(\mathbf{x})_T]$$

3.3 Pros and Cons of the two pretraining objectives

- **Independence Assumption**: As emphasized by the ≈ sign in Eq. (2), BERT factorizes the joint conditional probability p(x_bar|x_hat) based on an independence assumption that all masked tokens x_bar are separately reconstructed. In comparison, the AR language modeling objective (1) factorizes p_θ(x) using the product rule that holds universally without such an independence assumption.

  For example, “New”, ”York” are independent assumed by BERT model, but we know they are highly corelated.

3.3 Pros and Cons of the two pretraining objectives

- **Input noise**: The input to BERT contains artificial symbols like [MASK] that never occur in downstream tasks, which creates a pretrain-finetune discrepancy. Replacing [MASK] with original tokens as in BERT paper

  (In the paper, “masked” token with (1) 80% the [MASK] token (2) 10% a random token (3) 10% the unchanged token)

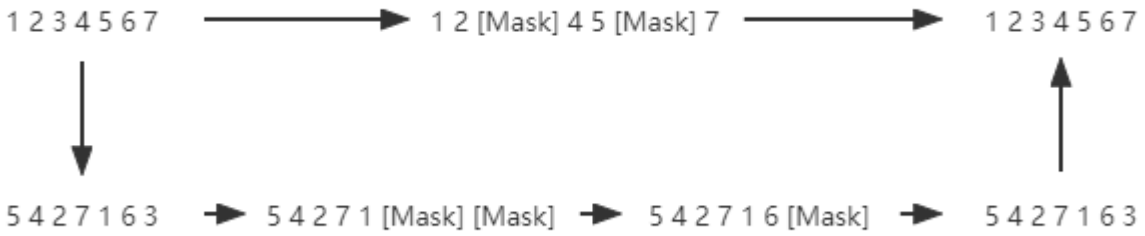
  does not solve the problem because original tokens can be only used with a small probability — otherwise Eq. (2) will be trivial to optimize. In comparison, AR language modeling does not rely on any input corruption and does not suffer from this issue.

3.3 Pros and Cons of the two pretraining objectives

- **Context dependency**: The AR representation $h\theta(x1:t-1)$ is only conditioned on the tokens up to position t (i.e. tokens to the left), while the BERT representation $H\theta(x)t$ has access to the contextual information on both sides. As a result, the BERT objective allows the model to be pretrained to better capture bidirectional context.

3.4 Permutation Language Modeling(PLM)

Randomly select one kind of sentence permutation, and then "mask" a certain amount of words at the end, and finally use AR to predict the words that are "masked" according to this arrangement.



Remark: The proposed objective only permutes the factorization order, not the sequence order. In other words, we keep the original sequence order, use the positional encodings corresponding to the original sequence, and rely on a proper attention mask in Transformers to achieve permutation of the factorization order.

3.4 Implement of PLM

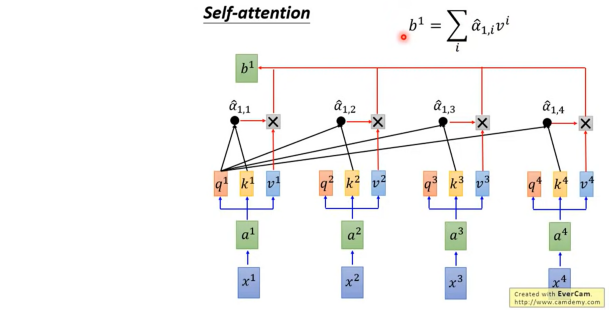
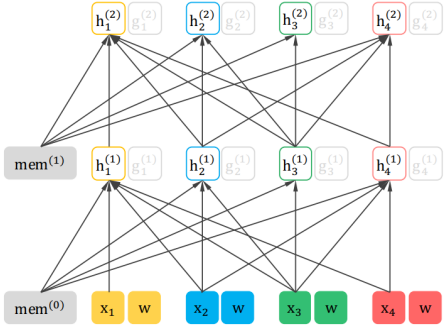
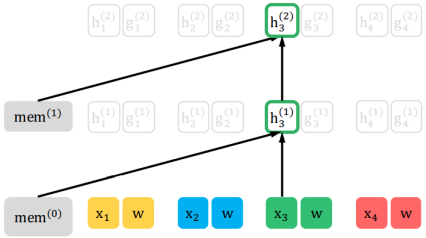


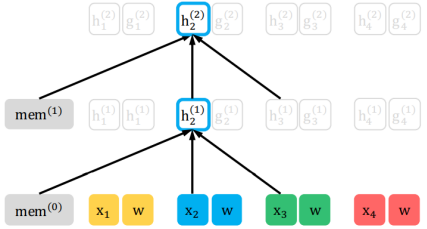
Figure by Hung-yi Lee



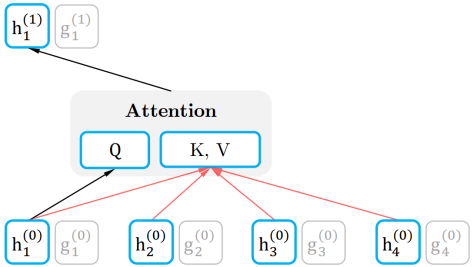
Joint View of the Content Stream
(Factorization order: 3 → 2 → 4 → 1)



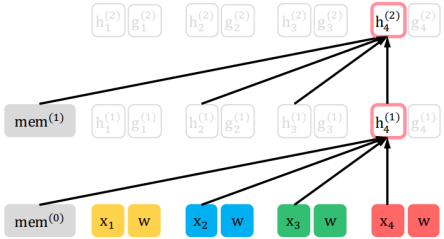
Position-3 View



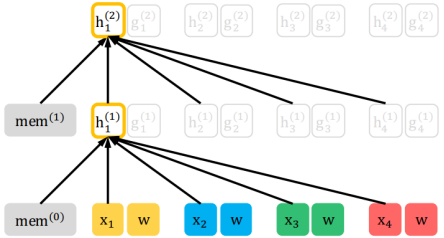
Position-2 View



(a)



Position-4 View



Position-1 View

3.5 Problem for simple attention layer

Consider z=[1 3 2 4], z_3 = 2

$$p_\theta(X_{z_3} = x | x_{z_1 z_2}) = p_\theta(X_2 = x | x_1 x_3) = \frac{exp(e(x)^T h_\theta(x_1 x_3))}{\sum_{x'} exp(e(x')^T h_\theta(x_1 x_3))}$$

Consider z'=[1 3 4 2], z_3 = 4

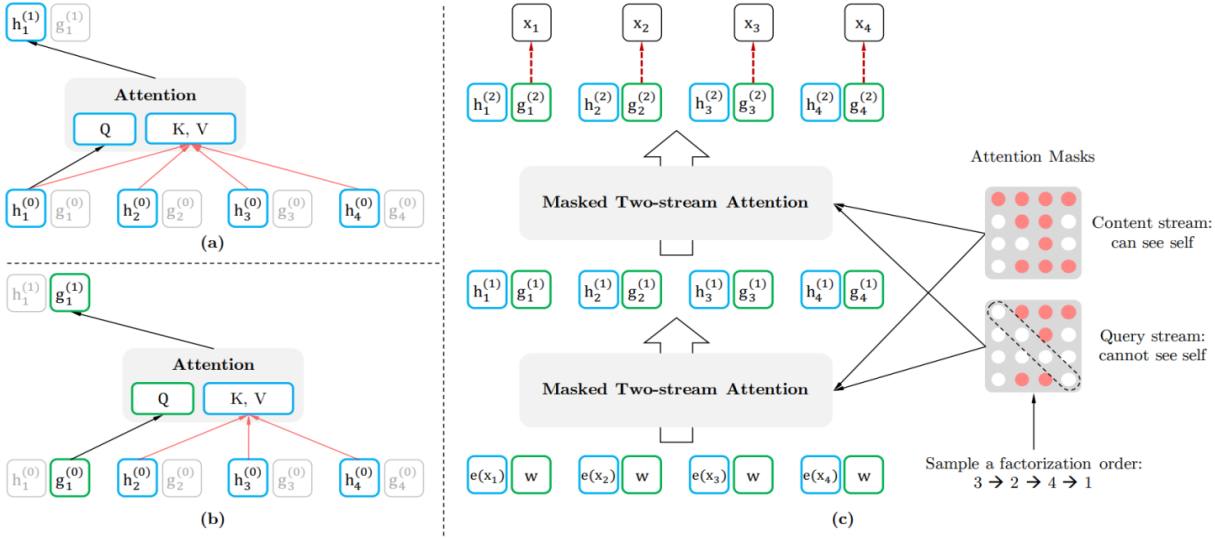
$$p_\theta(X_{z_3} = x | x_{z_1 z_2}) = p_\theta(X_4 = x | x_1 x_3) = \frac{exp(e(x)^T h_\theta(x_1 x_3))}{\sum_{x'} exp(e(x')^T h_\theta(x_1 x_3))}$$

For example, "I love Hong Kong".
In this distribution fuction,
p(X_2 = "Kong"|"I","Hong")=
p(X_4 = "Kong"|"I","Hong").
Apparently, "I Hong Kong" will make sence and its appearance likelihood is higher than "I Kong Hong".
This distribution function is not will defined and is lack of word position's information.

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{z_{<t}}) = \frac{\exp\left(e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}{\sum_{x'} \exp\left(e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}, \quad (4)$$

where gθ(xz<t , zt) denotes a new type of representations which additionally take the target position zt as input

Two-Stream Self-Attention for Target-Aware Representations



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

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

- The content representation hθ(xz≤t), or abbreviated as hzt , which serves a similar role to the standard hidden states in Transformer. This representation encodes both the context and xzt itself.
- The query representation gθ(xz<t , zt), or abbreviated as gzt , which only has access to the contextual information xz<t and the position zt, but not the content xzt

3.6 Partial Prediction

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T}\log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}})\right]. \tag{3}$$

While the permutation language modeling objective (3) has several benefits, it is a much more challenging optimization problem due to the permutation and causes slow convergence in preliminary experiments. To reduce the optimization difficulty, we choose to only predict the last tokens in a factorization order. Formally, we split z into a non-target subsequence z≤c and a target subsequence z>c, where c is the cutting point.

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\log p_\theta(\mathbf{x}_{\mathbf{z}_{>c}} \mid \mathbf{x}_{\mathbf{z}_{\le c}})\right] = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=c+1}^{|\mathbf{z}|}\log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}})\right]. \tag{5}$$

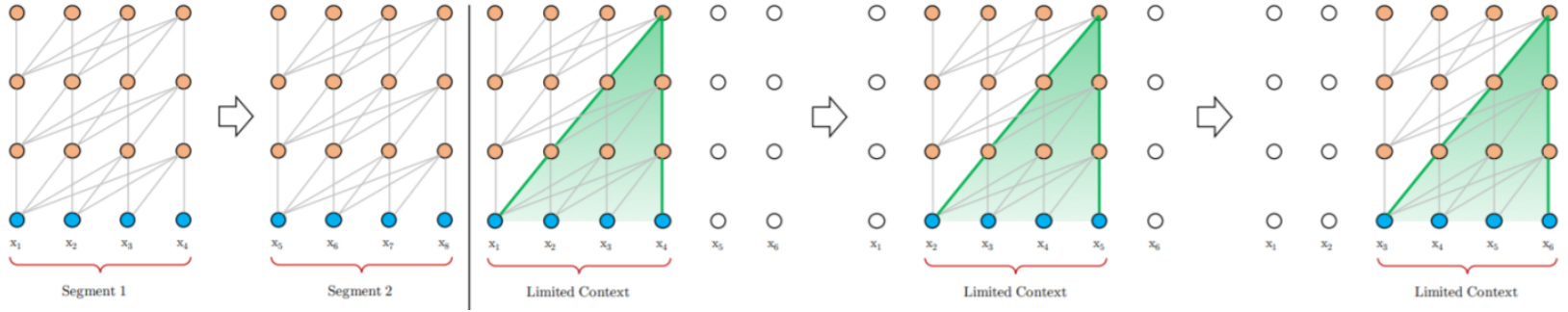
$$\frac{|z|-c}{|z|}=\frac{1}{K}$$

The best K value is between 6 and 7 (better). In fact, if we take the reciprocal of K and convert it to a percentage, we will find that the best ratio is between 14.3% and 16.7%.The percentage of BERT that hides the token as "[MASK]" is 15%

3.7 Transformer-XL

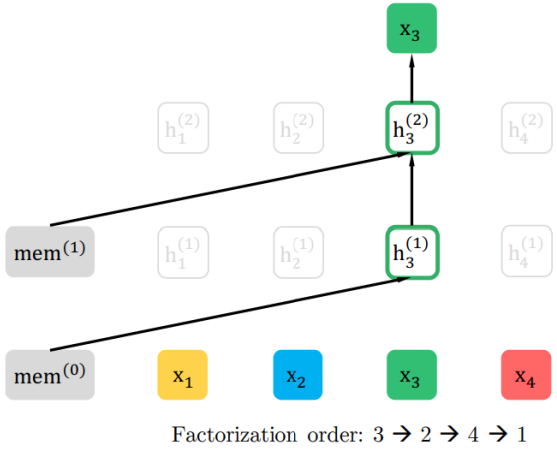
Transformer requires that the input is a fixed-length word sequence.

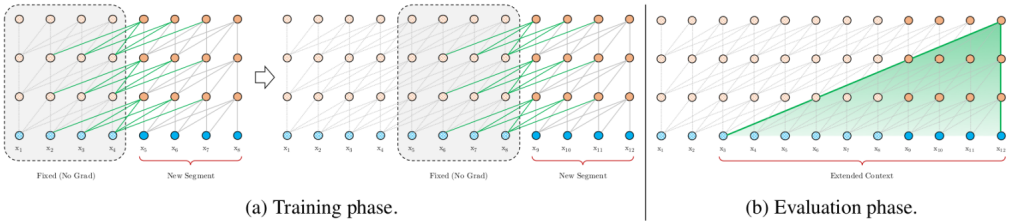
As a consequence of the fixed context length, the model cannot capture any longer-term dependency beyond the predefined context length. In addition, the fixed-length segments are created by selecting a consecutive chunk of symbols without respecting the sentence or any other semantic boundary. Hence, the model lacks necessary contextual information needed to well predict the first few symbols, leading to inefficient optimization and inferior performance. We refer to this problem as **context fragmentation**.



3.7 Segment recurrence mechanism

In fact, the idea is very simple, because generally when training Transformer, the text will be processed into a segment according to a certain length. For example, when BERT is pre-processed, it will first be processed into 512-length samples, even the text before processing may be longer. In this case, there is some longer context information that the model cannot learn





(a) Training phase. (b) Evaluation phase.

So what Segment Recurrence Mechanism wants to do is. After the previous segment is calculated, all the hidden states (hidden states) calculated by it are saved and put into a Memory, and then in the current segment calculation, the previously saved hidden states and the hidden states of the current segment are saved Put together K and V as the Attention mechanism to obtain longer context information.

3.7 Relative Positional&Segment Encodings

$$\mathbf{A}_{i,j}^{\text{abs}} = q_i^\top k_j = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}$$

$$\mathbf{A}_{i,j}^{\text{rel}} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} + \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}$$

We store [x1 x2 x3 x4] in memory. However, when we want to use it in next sentence [x5 x6 x7 x8], the positional enconding in the sentence is changed. We have to use relative positional encoding.

Same method for segments.

3.8 Pretraining & SOTA in text classification

$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Model	**IMDB**	**Yelp-2**	**Yelp-5**	**DBpedia**	**AG**	**Amazon-2**	**Amazon-5**
CNN [15]	-	2.90	32.39	0.84	6.57	3.79	36.24
DPCNN [15]	-	2.64	30.58	0.88	6.87	3.32	34.81
Mixed VAT [31, 23]	4.32	-	-	0.70	4.95	-	-
ULMFiT [14]	4.6	2.16	29.98	0.80	5.01	-	-
BERT [35]	4.51	1.89	29.32	0.64	-	2.63	34.17
XLNet	**3.20**	**1.37**	**27.05**	**0.60**	**4.45**	**2.11**	**31.67**

Table 4: Comparison with state-of-the-art error rates on the test sets of several text classification datasets. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large).

3.9 Comparison with BERT & Own opinions

Both BERT and XLNet perform partial prediction, i.e., only predicting a subset of tokens in the sequence. This is a necessary choice for BERT because if all tokens are masked, it is impossible to make any meaningful predictions. In addition, for both BERT and XLNet, partial prediction plays a role of reducing optimization difficulty by only predicting tokens with sufficient context. However, the independence assumption discussed in Section 2.1 disables BERT to model dependency between targets.

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

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

In my perspective, in essene, these two process are equivalent.
Increase the data size used in the pre-training step and utilize transform-XL.
Memory assumption is high. we used Tesla P100 16G and it only supported XLNet-Base.
New idea but the model is complicated

188	scaiag		0.82071	7	15d