Comparative analysis and analysis of 15 pre-trained models

The following question list is the main thread of this article.

Question List

Q1: Compare pre-trained language models from different dimensions?

Q2: What are the NLP feature extraction mechanisms based on deep learning? What are the advantages and disadvantages of each?

Q3: What are the advantages and disadvantages of autoregressive and self-encoding language models?

Q4: What is the kernel mechanism of the one-way model? What are the disadvantages?

Q5: In-depth understanding of Transformer's internal mechanisms:

Why scale dot products instead of dot product models?

What advantages does a dot product model have over an additive model?

Why is the long mechanism effective?

Q6-Q10: Exploration of BERT Core Mechanism

Why is BERT so effective?

What are the advantages and disadvantages of BERT?

What downstream NLP tasks does BERT excel at?

Is BERT based on "word input" or "word input"? (For Chinese tasks)

Why is BERT not suitable for natural language generation tasks (NLG)?

Q11-Q15: Given the shortcomings of the BERT native model, the subsequent BERT series models are:

How to improve the generate task to Bert?

How to introduce knowledge to Bert?

How to introduce multi-task learning mechanism to Bert?

How to improve the mask strategy?

How to do Fine Tuning?

Q16: What is the background proposed by XLNet?

Q17: Why is XLNet so effective:

Why can PLM model bidirectional contexts?

How to solve the problem of no target location information?

Q18: How does Transformer-XL model long text?

The following article will discuss the above issues one by one from the following aspects.

1. Comparison of pre-trained language models from different perspectives
2.The basis of pre-trained language models: feature extraction mechanism + classification of language models
3. Review of One-way Model + Exploration of Kernel Mechanism
4. BERT's kernel mechanism
5. Introduction of BERT series models
6.The exploration of the core mechanism of XLNET
7.The future of pre-trained language models

1. Comparison of pre-trained language models from different perspectives
Q1: Compare 4 pre-trained language models from feature extraction, pre-trained language model target, improvement direction of BERT series models, and feature representation:

Different feature extraction mechanisms:
RNNs: ELMO / ULMFiT / SiATL;
Transformer: GPT1.0 / GPT2.0 / BERT series models;
Transformer-XL: XLNet;

Different pre-trained language goals:
AutoEncode: BERT series models;
AutoRegression: one-way model (ELMO / ULMFiT / SiATL / GPT1.0 / GPT2.0) and XLNet;

Improvement of BERT series models:
Introduce common sense: ERNIE1.0 / ERNIE (THU) / ERNIE2.0 (referred to as

"ERNIE series");
Introduce multi-task learning: MTDNN / ERNIE2.0;
Improvements based on generation tasks: MASS / UNILM;
Different mask strategies: WWM / ERNIE series / SpanBERT;
Fine-tuning: ROBERTa;
Feature representation (whether it can represent context)
One-way feature representation: one-way model (ELMO / ULMFiT / SiATL / GPT1.0 / GPT2.0);
Two-way feature representation: BERT series model + XLNet;

2. The basis of pre-trained language models: feature extraction mechanism + classification of language models
Q2: What are the NLP feature extraction mechanisms based on deep learning? What are the advantages and disadvantages of each?

1) Can it handle long-distance dependence?
Long distance-dependent modeling capabilities: Transformer-XL> Transformer> RNNs> CNNs
MLP: It does not consider sequence (position) information and cannot process variable-length sequences, such as NNLM and word2vec;
CNN: Consider sequence (position) information, cannot handle long-distance dependencies, focus on n-gram extraction, pooling operation will cause sequence (position) information loss;
RNNs: Naturally suitable for processing sequence (position) information, but still unable to handle long-distance dependencies (such as the disappearance of gradients caused by BPTT), so they are also called "long short-term memory units (LSTM)"
Transformer / Transformer-XL: self-attention solves long-distance dependencies without position deviation;

2) Feedforward / cyclic network or serial/parallel computing
MLP / CNNs / Transformer: feedforward / parallel
RNNs / Transformer-XL: loop / serial:
Q3: Pros and cons of self-regression models and self-coding models

1) Self-regression model:

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

advantage:

Density estimation of the joint probability of text sequences, which is a traditional language model, which is naturally suitable for handling naturally generated tasks;

Disadvantages:

The joint probability is decomposed (sequentially disassembled) from left to right according to the text sequence, and the bidirectional feature cannot be characterized by context information;

Representative model: ELMO / GPT1.0 / GPT2.0;

Improvement: XLNet promotes the traditional autoregressive language model, changes sequential disassembly to random disassembly (permutation of language models), and generates context-dependent two-way feature representations;

2) Self-coding language model

Advantages: The essence is the self-encoding feature representation of noise reduction. The MLM is constructed by introducing noise [MASK] to obtain the context-dependent two-way feature representation;

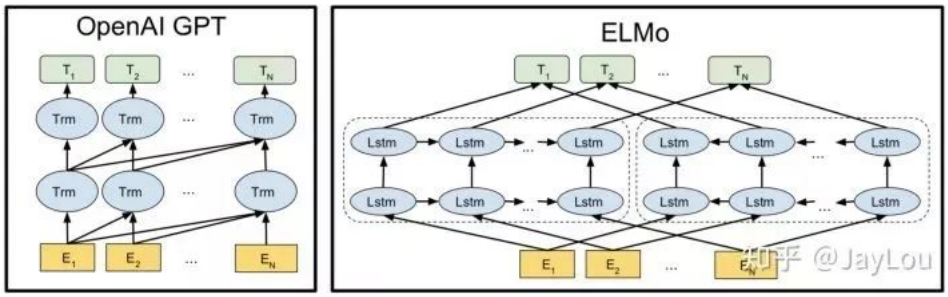
Disadvantages: Independence is introduced, which is a biased estimate of joint probability, without taking into account the correlation between predictions [MASK]

It is not suitable to directly deal with the generation task. The setting of the MLM pre-training target causes the pre-training process and the generation process to be inconsistent;

The noise, [MASK], during pre-training will not appear during the finetune phase, causing a two-phase mismatch problem;

Representative model: BERT series model;

3. one-way model review + exploration of the kernel mechanism



Q4: What is the kernel mechanism of the one-way model? What are the disadvantages?

1) ELMo (Allen Institute) [6]

Key points:

The introduction of a bidirectional language model is an integration of two unidirectional language models (forward and backward);

By saving the pre-trained 2-layer biLSTM, it can be applied to downstream tasks through feature integration or finetune;

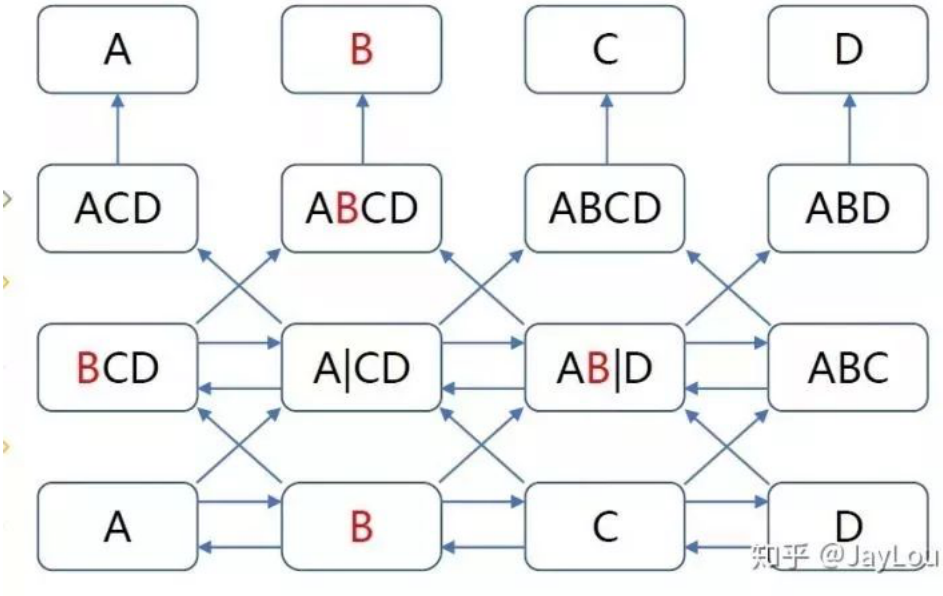
Disadvantages:

It is an autoregressive language model in nature. It can only obtain unidirectional feature representations and cannot simultaneously obtain context representations.

LSTM cannot resolve long-distance dependencies.

Why can't biLSTM build bidirectional language models?

Cannot take two layers of biLSTM to perform feature extraction at the same time to build a bidirectional language model, otherwise, the problem of label leakage will occur; therefore, the forward and backward LSTM parameters of ELMO are independent, sharing word vectors, and constructing language models independently;



2) ULMFiT (fast.ai) / SiATL

1) Key points of ULMFiT [7]:

Three-stage training: LM pre-training + fine-tuning specific tasks LM + fine-tuning specific classification tasks;

Feature extraction: 3-layer AWD-LSTM;

Fine-tune specific classification tasks: thawing layer by layer;

2) Key points of SiATL [8]:

Two-stage training: LM pre-training + fine-tuning of classification tasks for specific tasks (introducing LM as an auxiliary target, which is useful for small data, as opposed to GPT);

-Feature extraction: LSTM + self-attention;

Fine-tune specific classification tasks: thawing layer by layer;

Both use some techniques to solve the catastrophic forgetting problem in the finetune process: if the unsupervised data used for pre-training and task data are in different fields, the effect of layer-by-layer thawing is more obvious [9];

3) GPT1.0 / GPT2.0 (OpenAI)

GPT1.0 [10]

Key points:

The transformer is used for feature extraction, and Transformer is applied to pre-trained language models for the first time;

The finetune stage introduces language model auxiliary goals (assistance goals are useful for large data sets, but small data has declined, as opposed to SiATL), to solve the catastrophic forgetting in the finetune process;

Pre-training is consistent with finetuning, unifying the two-phase framework;

GPT2.0 [11]

Key points:

There is no fine-tuning process for specific models: GPT 2.0 believes that pre-training already contains a lot of information required for specific tasks.

Generate tasks with good results, using wider coverage and higher quality data;

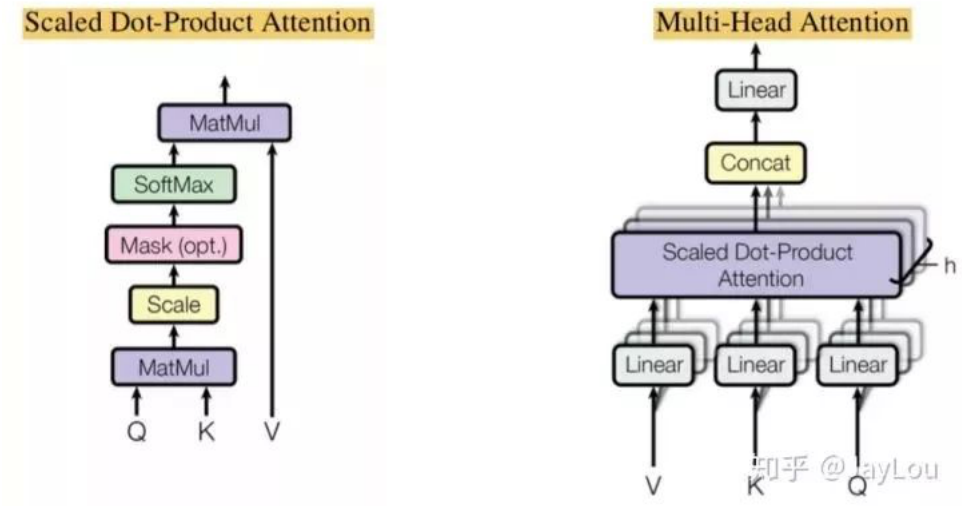
Disadvantages:

It is still a one-way autoregressive language model and cannot obtain context-sensitive feature representations;

Four, BERT kernel mechanism exploration

This section introduces the core mechanism of BERT. Before answering "Why is BERT so effective?", First, introduce the core mechanism of the Transformer.

Q5: In-depth understanding of Transformer [12] internal mechanism



1) Multi-Head Attention and Scaled Dot-Product Attention

The essence is that self-attention dynamically encodes a variable-length sequence through an attention mask to solve long-distance dependencies, no positional deviations, and parallel computing.

Why scale dot products instead of dot product models?
When the dimension d of the input information is relatively high, the value of the dot product model usually has a large variance, which causes the gradient of the softmax function to be smaller. Therefore, the scaled dot product model can better solve this problem.

Why is the bilinear dot product model (after linear transformation of Q and K)?
The bilinear dot product model introduces asymmetry and is more robust (the diagonal element value of the Attention mask is not necessarily large, which means that the current position may not necessarily have a higher attention score).

What advantages does a dot product model have over an additive model?
The common Attention mechanism is the additive model and the dot product model. In theory, the complexity of the additive model and the dot product model is similar, but the dot product model can make better use of the matrix product to achieve higher calculation efficiency (actually (As the dimension d increases, the additive model will be significantly better than the dot product model).

Why is the long mechanism effective?
Similar to feature selection in CNN through multi-channel mechanism;
In the Transformer, the scaled Dot-Product Attention is first performed through the split head, which can make the dimension d of the dot product calculation small (to prevent the gradient from disappearing) and reduce the attention mask matrix.

2) Position-wise Feed-Forward Networks
FFN maps the Multi-Head Attention results at each location to a feature space of a larger dimension, then uses ReLU to introduce non-linearity for filtering, and finally recovers back to the original dimension.
After the Transformer abandoned the LSTM structure, ReLU in FFN became a major

unit that provided nonlinear transformation.

3) Positional Encoding

Change the Positional Embedding to Positional Encoding. The main difference is that Positional Encoding is expressed in formulas and cannot be learned, while Positional Embedding is learnable (such as BERT). There is not much difference in training speed and model accuracy between the two solutions; the Positional Embedding position encoding range is fixed, while the Positional Encoding encoding range is unlimited.

Translation of Chinese sentences in the image:

1. Why introduce sin and cos to build model for positional encoding?
2. Introduce sin and cos is to let model learn the relative position. For any two positions, (pos) and (pos + k), their positional encoding is fixed linear transformation of k.
3. It can be proved that any two positions' encoding with difference of k in position are equal in the Euclidean space, which is only related to k.

- 为什么引入sin 和 cos 建模Positional Encoding ?

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i1)} = cos(pos/10000^{2i/d_{model}})$$

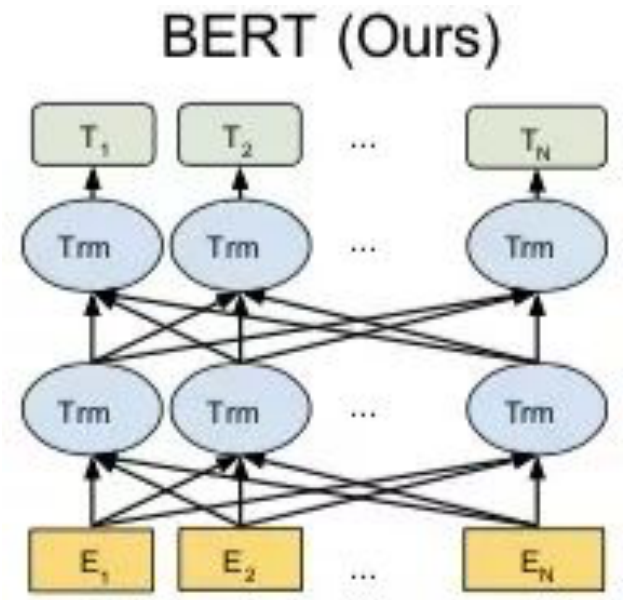
- 引入sin 和 cos 是为了使模型实现对相对位置的学习，两个位置 pos 和 pos+k 的位置编码是固定间距k的线性变化：

$$PE(posk, 2i) = PE(pos, 2i)PE(k, 2i1)PE(pos, 2i1)PE(k, 2i)$$
$$PE(posk, 2i1) = PE(pos, 2i1)PE(k, 2i1) - PE(pos, 2i)PE(k, 2i)$$

- 可以证明：间隔为k的任意两个位置编码的欧式空间距离是恒等的，只与k有关。

$$||[PE(posk, 2i), PE(posk, 2i1)] - [PE(pos, 2i), PE(pos, 2i1)]|^2$$
$$= |sin(posk, 2i) - sin(pos, 2i)|^2|cos(posk, 2i) - cos(pos, 2i)|^2$$
$$= 2 - 2cos(k, 2i)$$

Q6: Why is BERT [13] so effective?



It introduces Masked Language Model (MLM) pre-training target, which can obtain context-dependent bidirectional feature representation;

It introduces Next Sentence Prediction (NSP) pre-training target, good at processing sentence or paragraph matching tasks;

It introduces the powerful feature extraction mechanism Transformer (multiple mechanisms coexist):

Multi-Head self-attention: The multi-head mechanism is similar to the "multi-channel" feature extraction. Self-attention dynamically encodes a variable-length sequence through an attention mask to solve long-distance dependencies (no position deviation) and can be calculated in parallel;

Feed-forward: Calculate non-linear hierarchical features in the location dimension;

Layer Norm & Residuals: Speed up training to make "deep" networks more robust;

Introduce large-scale, high-quality text data;

Q7: What are the advantages and disadvantages of BERT?

Advantages: Ability to obtain context-sensitive bidirectional feature representations;

Disadvantages:

The poor performance of the generation task: the inconsistency between the pre-training process and the generation process leads to poor results on the generation task;

Take the independence assumption: the correlation between predictions [MASK] is not considered, and it is a biased estimate of the joint probability of the language model (not a density estimate)

Input noise [MASK], causing the difference between pre-training and fine-tuning;
Unable to document level NLP tasks, only suitable for sentence and paragraph level tasks;

Q8: What downstream NLP tasks are BERT good at [14]?
1. Suitable for sentence-level and paragraph-level tasks, not for document-level tasks;
2. It is suitable for processing high-level semantic information extraction tasks, and has little effect on improving shallow semantic information extraction tasks (such as some simple text classification tasks);
3. Suitable for sentence/paragraph matching tasks; therefore, in some tasks, auxiliary sentences (similar to matching tasks) can be constructed to improve results (such as relationship extraction/emotion mining tasks);
4. Not suitable for NLG tasks;

Q9: Is BERT based on "word input" or "word input"? (For Chinese tasks)
1. If it is based on "word input", it will exacerbate the OOV problem and increase the input space. It is necessary to use a much larger corpus to learn the function mapping of the input space to the label space.
2. With the ability of Transfomer feature extraction, word segmentation is no longer necessary, and feature learning at the word level can be incorporated into internal feature representation learning.

Q10: Why is BERT not suitable for natural language generation tasks (NLG)?
1. Because BERT itself is inconsistent in the pre-training process and the generation process, it does not have a corresponding mechanism for the generation task, resulting in poor results on the generation task and cannot be directly applied to the generation task.
2. If BERT or GPT is used for the natural language generation task of Seq2Seq, the encoder and decoder can be pre-trained separately, but the encoder-attention-decoder structure is not jointly trained, and BERT and GPT are in the conditional generation task Just sub-optimal.

5. Introduction to the progress of BERT series models

This section introduces some models, all of which are improvements to the BERT native model in some directions.

Q11: For the BERT native model, how do subsequent BERT series models improve the Generation Task?

1) MASS (Microsoft) [15]

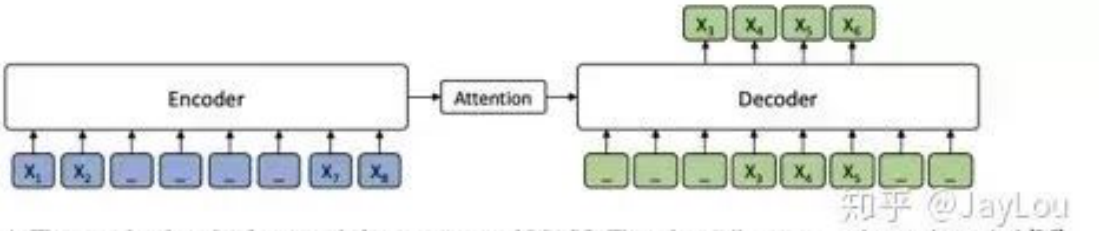


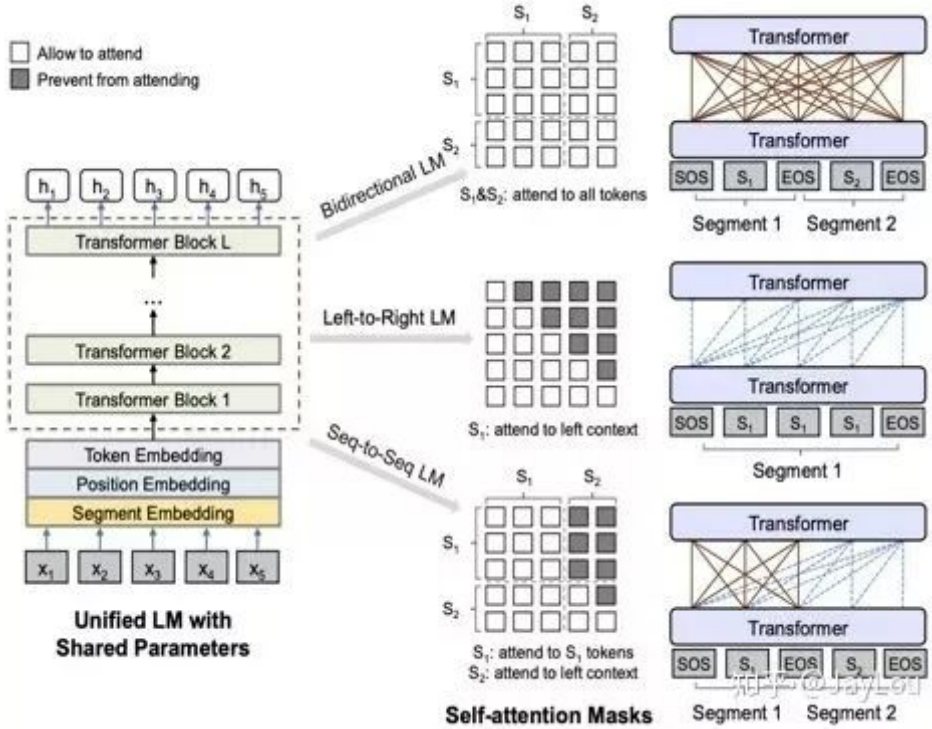
Figure 1. The encoder-decoder framework for our proposed MASS. The token "_" represents the mask symbol [M].

Unified pre-training framework: Through a similar Seq2Seq framework, the BERT and LM models are unified during the pre-training stage;

Encoder understands unmasked tokens; Decoder needs to predict continuous [MASK] tokens to get more language information; Decoder extracts more information from Encoder;

When k = 1 or n, the probabilistic form of MASS is consistent with the standard LM in BERT and GPT (k is the continuous segment length of the mask)

2) UNILM (Microsoft) [16]



Unified pre-training framework: unify BERT and LM directly from the perspective of the mask matrix;

3 Attention Mask matrices: LM, MLM, Seq2Seq LM;

Note: LM in UNILM is not a traditional LM model, it is still implemented by introducing [MASK];

Q12: For the BERT native model, how did the subsequent BERT series models introduce knowledge?

1) ERNIE 1.0 (Baidu) [17]

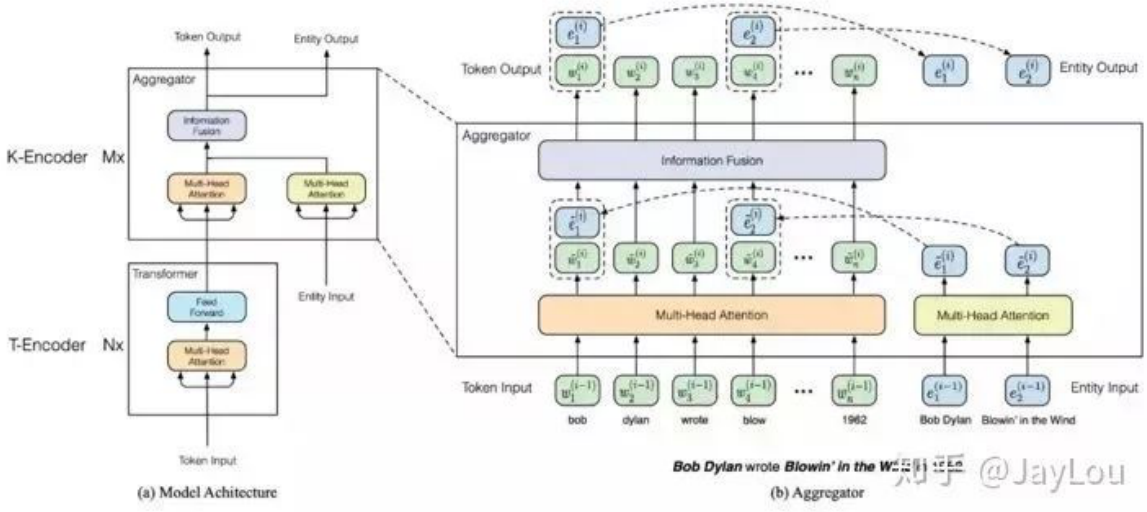
Introduce knowledge (actually pre-recognized entities) during the pre-training phase, and introduce 3 [MASK] strategy predictions:

Basic-Level Masking: Like BERT, masking subwords cannot obtain high-level semantics;

Phrase-Level Masking: continuous phrase mask;

Entity-Level Masking: mask entity;

2) ERNIE (THU) [18]



Based on the BERT pre-trained native model, the entities in the text are aligned to the external knowledge map, and the entity vector is obtained as the input of ERNIE through knowledge embedding;

Because the pre-training process of language representation and the knowledge representation process are very different, two independent vector spaces will be

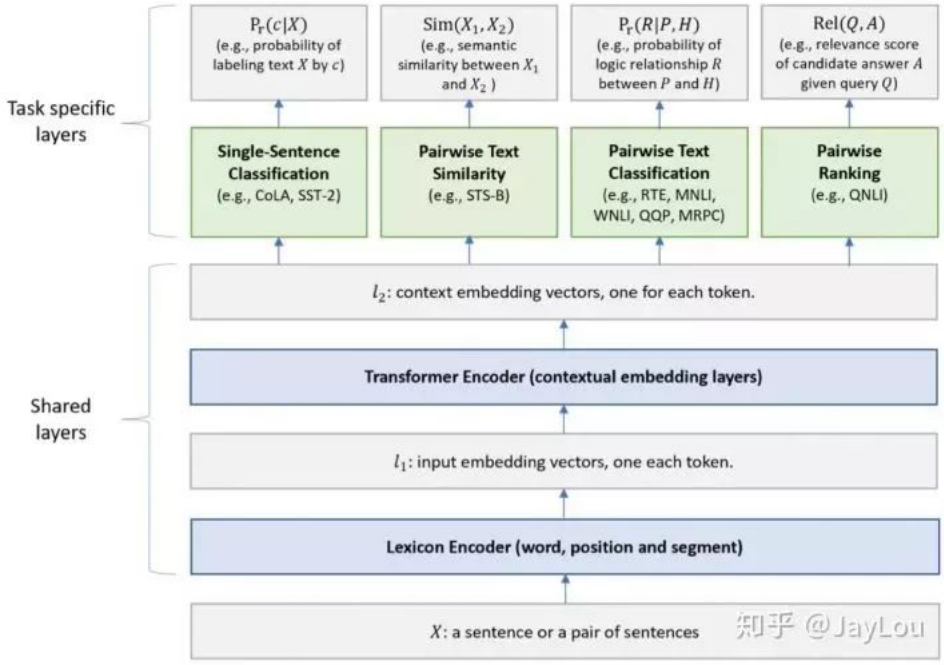
generated. To solve the above problem, where there is an entity input, the entity vector, and the text representation are fused by a non-linear transformation to fuse vocabulary, syntax and knowledge information;

Introduce improved pre-training target Denoising entity auto-encoder (DEA): Require the model to predict the corresponding entity based on a given entity sequence and text sequence;

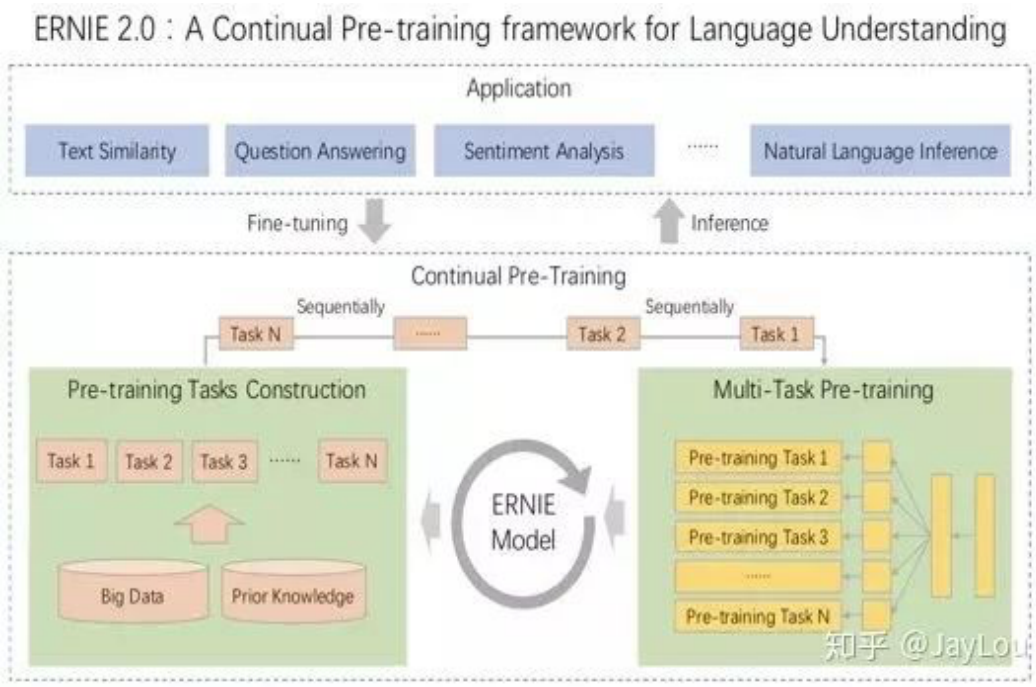
Q13: For the BERT native model, how did the subsequent BERT series models introduce multi-task learning mechanism?

Multi-task learning [19] refers to learning multiple related tasks at the same time, allowing these tasks to share knowledge during the learning process, and using the correlation between multiple tasks to improve the model's performance and Generalization. Multi-task learning can be considered as an inductive transfer learning, that is, to improve the generalization ability by using the information contained in related tasks as an inductive bias. The training mechanism of multi-task learning is divided into simultaneous training and alternating training.

1) MTDNN (Microsoft) [20]: Introducing the multi-task learning mechanism in downstream tasks



2) ERNIE 2.0 (Baidu) [21]: Introduce multi-task learning during pre-training



MTDNN introduces multi-task mechanism in downstream tasks, while ERNIE 2.0 introduces multi-task learning (interaction with prior knowledge base) in pre-training so that the model can learn more language knowledge from different tasks.

There are three main tasks:

word-aware task: capture vocabulary-level information;

structure-aware task: capture syntactic-level information;

semantic-aware task: capture semantic information;

The main way is to build incremental learning (more tasks can be continuously introduced later) model, and continuously update the pre-trained model through multi-task learning. This continuous and alternating learning paradigm will not make the model forget the language knowledge learned before.

Several sub-tasks of the three major types of tasks are used for training together. When new tasks are introduced, the previous tasks will continue to be introduced to prevent forgetting the knowledge that has been learned before, specifically a process of gradually increasing the number of tasks [22]:

(task1)-> (task1, task2)-> (task1, task2, task3)-> ...-> (task1, task2, ..., taskN),

Q14: For the BERT native model, how do subsequent BERT series models improve the mask strategy?

Native BERT model: mask according to the subword dimension, and then make

predictions;
BERT WWM (Google): Mask according to the whole word dimension, and then make predictions;

ERNIE and other series: introducing external knowledge, masking according to the entity dimension, and then making predictions;

SpanBert: instead of masking according to the boundary information of prior words/entities/phrases, it takes a random mask:
Use Span Masking: randomly select a length of space according to the geometric distribution, and then randomly choose the starting position based on the uniform distribution, and finally choose the length mask; by sampling, the average masked length is the length of 3.8 words;
Introduce Span Boundary Objective: The new pre-training goal is to enable the word vectors of the masked Span boundary to learn the masked part of Span; the new pre-training goal is used with MLM;
Note: BERT WWM, ERNIE and other series, SpanBERT aim to implicitly learn the relationship between the predicted words (the strong correlation of the mask part itself) [23], and in XLNet, it is displayed by PLM plus autoregression. Learning the relationship between predicted words

Q15: For the BERT native model, how do the subsequent BERT series models perform fine-tuning?
RoBERTa (FaceBook) [24]:
Discard NSP for better results;
Dynamically change the masking strategy, copy the data for 10 copies, and then uniformly perform random masking;
Adjust the peak learning rate and the number of warm-up update steps;
Train on longer sequences: do not truncate the sequence, use full-length sequences;
Exploring the core mechanism of XLNet
After the BERT series of models, Google ’s XLNet has significantly surpassed BERT in tasks such as question answering, text classification, and natural language understanding. Connection language modeling methods and pre-training methods.

Q16: What is the background proposed by XLNet [26]?

Pre-trained models such as ELMO and GPT are based on traditional language models (autoregressive language models AR). Autoregressive language models are naturally suitable for generating tasks, but they cannot characterize bidirectional contexts. Therefore, people have turned to the study of self-coding ideas (Such as BERT series models);

Although the self-encoded language model (AE) can be characterized in a two-way context, it:

The BERT series models introduce independence assumptions without considering the correlation between predictions [MASK];

The setting of MLM pre-training targets causes the pre-training process and the generation process to be inconsistent;

[MASK] noise during pre-training will not appear during the finetune phase, causing a two-phase mismatch problem;

Is there any way to build a model that has the advantages of both AR and AE without them?

Q17: Why is XLNet so effective: Analysis of the kernel mechanism

1) Permutation language model (Permutation LM, PLM)

If you measure the number of dependencies modeled in the sequence, the standard LM can reach the upper bound. Unlike MLM, LM does not rely on any independent assumptions. Drawing on the ideas of NADE [27], XLNet promotes standard LM to PLM.

Why can PLM model can build model with bidirectional contexts?

The essence of PLM is the embodiment of multiple decomposition mechanisms of LM joint probability;

Sequential disassembly of LM is generalized to random disassembly, but the original position information of each word needs to be retained (PLM is just a factorization/permutation of the language model modeling method, not a rearrangement of the position information of words!)

If you traverse the $T!$ Decomposition method and the model parameters are shared,

PLM will learn a variety of bidirectional contexts; in other words, when we consider all possible $T!$ Permutations, for all the predicted words The context can be learned! Since traversing the $T!$ The path is very computationally expensive (for a 10-word sentence, 10! = 3628800). Therefore, it is only possible to randomly sample the parts in $T!$ And ask for expectations;

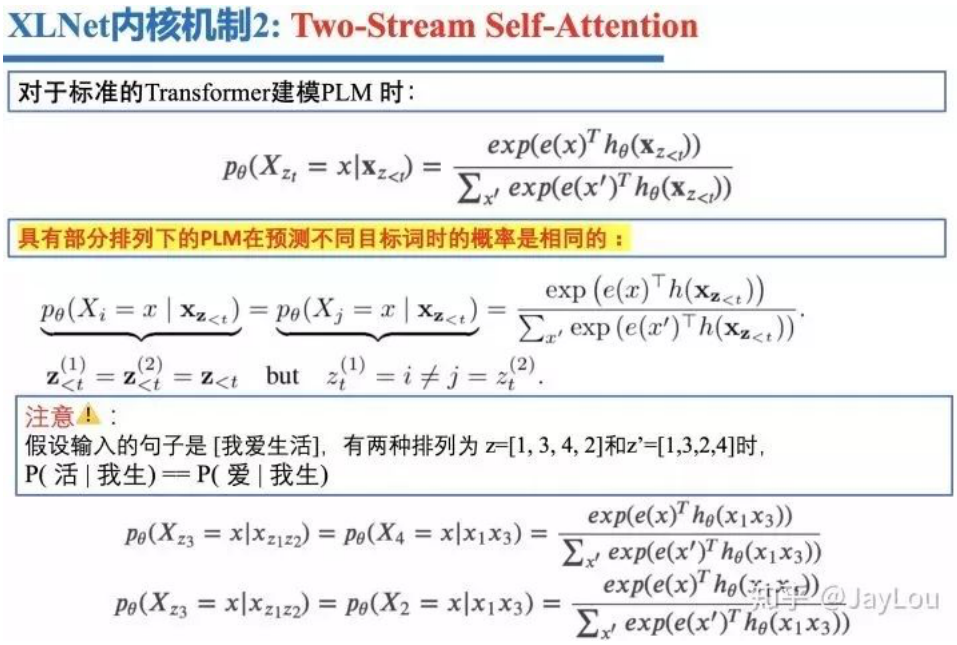
$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T}\log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}<t})\right].$$

2) Two-Stream Self-Attention

If a standard Transformer is used to model the PLM, there will be a problem that there is no target location information. The point of the problem is that the model does not know which word to predict, which leads to the probability that the PLM with partial permutations is the same when predicting different target words.

Picture Translation:

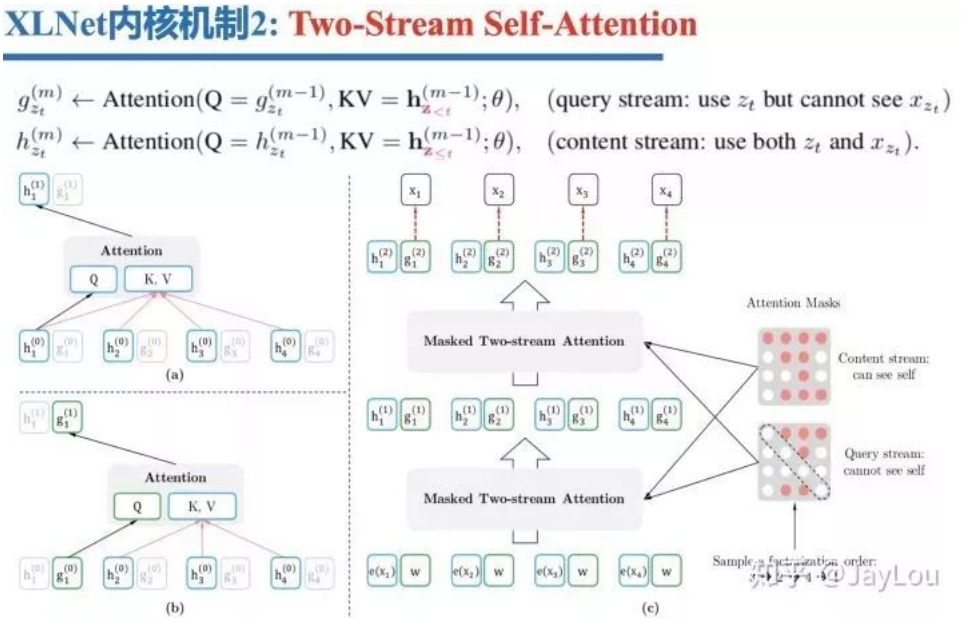
1. For standard Transformer, when it's building PLM:
2. PLM, which has part permutation, has same probabilities when prediction targets.
3. Attention: if the sentence is 'I love the life', when there are two permutation: z=[1,3,4,2] and z' = [1,3,2,4], P(love | I, the) = P(life | I, the)



How to solve the problem of no target location information?

For the problem of no target location information, XLNet introduced Two-Stream Self-Attention:

Translation: XLNet core mechanism 2



The Query stream is just to predict the current word, it only contains the location information, and does not contain the content information of the word;

The Content stream mainly provides content vectors for other words for the Query stream, including location information and content information;

3) Merging the advantages of Transformer-XL (see Q18 for details)

Q18: How does Transformer-XL [28] model long texts?

BERT (Transformer) has a large input length of 512, so how do you model document-level text?

The vanilla model performs segmentation, but there is a problem of context fragmentation (the semantic information of continuous documents cannot be modeled). At the same time, inference needs to be repeated for calculation, so the inference speed will be slow;

Transformer-XL improvements

Each segment should have a different position encoding, so Transformer-XL uses relative position encoding;

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

The representation calculated by the previous segment is repaired and cached to serve as an extension context resume when the model processes the next new segment;

It is possible that the length of the dependency relationship is increased by N times, where N represents the depth of the network;

Solve the problem of context fragmentation and provide the necessary context for the token in front of the new segment;

Transformer-XL is 1800+ times faster than vanilla Transformer during the evaluation of language modeling tasks because it does not require repeated calculations;

Introduce the recurrence mechanism (do not use BPTT to derive):

Introduce relative position encoding scheme:

The future of pre-trained language models

The above [pre-trained language model] is mainly introduced from two aspects: one is the general comparison; the other is the one-way language model, the BERT series model, and the XLNet model.

It can be seen that more exploration directions in the future [pre-trained language models] are mainly [25]:

Reviving the language model: further, improve the language model goals and constantly break the upper bound of the model;

Big data and big computing power: push big data and big computing power to the extreme;

Faster inference: Is it possible for lightweight models to achieve SOTA?

Introduce richer knowledge information, finer tuning parameters, and more valuable MASK strategies;

Unified condition generation task framework, such as unified encoding and decoding tasks based on XLNet, while considering faster decoding methods;

references

[1] NLP will usher in a golden decade https://www.msra.cn/zh-cn/news/executivebylines/tech-bylines-nlp

[2] a review of the recent history of NLP

[3] AIS: ACL2019 Progress Report

[4] ACL Chairman Zhou Ming: Embrace the bright future of ACL and NLP

[5] Language model pre-training method in natural language processing https://www.jiqizhixin.com/articles/2018-10-22-3

[6] ELMO: Deep contextualized word representations

[7] ULMFiT: Universal Language Model Fine-tuning)

[8] SiATL: An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models

[9] NLP in BERT and Post-era https://zhuanlan.zhihu.com/p/66676144

[10] GPT: Improving Language Understanding by Generative Pre-Training

[11] GPT2.0: Language Models are Unsupervised Multitask Learners

[12] Transformer: Attention is all you need

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

[14] Innovation in the era of Bert (Application): Bert's application progress in various fields of NLP https://zhuanlan.zhihu.com/p/68446772

[15] MASS: Masked Sequence to Sequence Pre-training for Language Generation

[16] UNILM: Unified Language Model Pre-training for Natural Language Understanding and Generation

[17] ERNIE: Enhanced Representation through Knowledge Integration

[18] ERNIE: Enhanced Language Representation with Information Entities

[19] nndl: Neural Networks and Deep Learning

[20] MT-DNN: Multi-Task Deep Neural Net for NLU

[21] ERNIE 2.0: A CONTINUAL PRE-TRAINING FRAMEWORK FOR LANGUAGE UNDERSTANDING

[22] Chen Kai:
https://www.zhihu.com/question/337827682/answer/768908184

[23] SpanBert: A deep exploration of Bert pre-training

[24] RoBERTa: A Robustly Optimized BERT Pretraining Approach

[25] ab They Created XLNet that Sweeped NLP: Interview with Dr. Yang Zhilin of

CMU

[26] XLnet: Generalized Autoregressive Pretraining for Language Understanding

[27] Neural autoregressive distribution estimation

[28] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context