An introduction of Deep Learning Based Word Representation Applied to Natural Language Processing

Zihao FU
School of Information and Software Engineering
University of Electric Science and Technology of China
Chengdu, China
764336630@qq.com

***Abstract*—In the area of Natural Language Processing, high-quality word representations play key roles in neural language processing tasks. Recently, various model designs and methods have blossomed in the domain of word representation. In this paper, we will explain the theories of two major language models such as autoencoding (AE) and autoregressive (AR), and illustrate the architectures of several notable examples of AE and AR including ELMo, GPT, BERT and XLnet. By comparing the pros and cons of these models theoretically and the performances of these models in experiments, we will deepen the understanding of various learning methods and realize the trend in the development of language models.**

Keywords-Natural Language Processing; Word representation; Feature-extraction; Transformer; AttentionIntroduction

I. Introduction

NLP (Natural Language Processing) is a theory-motivated range of computational techniques for automatic analysis and representation of human language. The hardship and complexity of the representation of human language are large because it is even hard for different people to reach an agreement on the understanding of certain language.

Since sentences consist of words, the basic and significant work for the representation of human language is word representation and one of the most difficult challenges of the word representation is feature-extraction. Recently, we have several feature-extraction including GRU, LSTM[1] and Transformer[2]. GRU and LSTM, aiming at solving gradient problems, are variants of RNN (Recurrent Neural Network). LSTM is less mature than Transformer because this network could be greatly influenced by the distance. Transformer uses matrixes to record the relationships among the tokens and thus avoid the defects of LSTM.

Besides feature-extraction, language models play important roles in the word representation, there exits two major kinds of language model—AE and AR[3]. As to AR model, given a sentence $\mathbf{x} = \{x_1, x_2 \ldots x_T\}$, and the word for prediction is x_t, thus we perform the pre-training by maxing the likelihood:

$$\log p_\theta(x) = \sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}) \tag{1}$$

where $x_{<t}$ means the context in the forward position of x_t. As to AE model, the method to perform the pre-training is maxing the likelihood:

$$\log p_\theta(\bar{x} \mid x) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid x) \tag{2}$$

where $\bar{x}$ represents that the target token is masked and m_t is a hype-parameter representing that x_t is masked. Of course, there exist advantages and disadvantages in both models. As to AE model, it closes the bidirectional information gap in AR language model by the means of reconstructing the original data in the corrupted context, which contributes to better performance. Nevertheless, during the finetune time, the artificial symbols like [MASK] will be absent. This is the reason why there exists pretrain-finetune[4] discrepancy.

The emerging models adopt different feature-extractions and language models. ELMo[4] employs LSTM and AR, thus it has the inherent disadvantages of LSTM and AR. GPT adopts transformer in its architecture so it has a better performance in feature-extraction, but it is still an AR model. BERT uses the AE model, although it outperforms GPT[5] a little, it could not overcome the shortcomings of AE. XLnet[3] successfully integrates the advantages of both AE and AR, it is nearly the best-performance model recently.

We explain the theories of feature-extraction in section II and illustrate the language models in section III, then we also list the results of the experiments of these language models in section IV. And finally, we summarize the pros and cons of different models and explore the further development of the word representation.

978-1-7281-5094-9/19/$31.00 ©2019 IEEE
DOI 10.1109/MLBDBI48998.2019.00025

II. FEATURED-EXTRACTION

Featured-extraction plays a significant role in exploring the relationship among the tokens in the sentences. LSTM was proposed more earlier and but owned its internal problems. Attention mechanism, which could allow the modeling of dependencies without regard to the distance of tokens in input and output sentences, is integrated into overcome the disadvantages of LSTM. Transformer, which has encoder-decoder architecture and attention mechanism, is a more mature network compared with LSTM.

A. LSTM

Traditional RNN (Recurrent Neural Networks)[21] consists of networks with loops to input and output information.

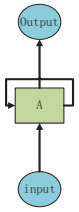


Figure 1. The input and output are connected by recurrent neural network 'A'

Fig. 1 illustrates the structure of this traditional RNN, in which an input is sent to A and then produce an output. In this kind of structure, 'A' loops allow the information to be passed to the next step. However, there exists a problem that if the connections between words in a sentence are not close enough, the output may not be appropriate, in other words, this traditional RNN has a bad performance in a 'long-term' situation.

In order to update RNN models to solve this kind of 'long-term' problems, we introduce the 'Long Short Term Memory networks' mechanism, usually called 'LSTM', which has a more complicated structure than the standard version of RNN. It can delete or add information automatically in order to retain the most useful information and enhance the efficiency of sentence transformation. The whole structure of 'LSTM' is displayed in Fig. 2:

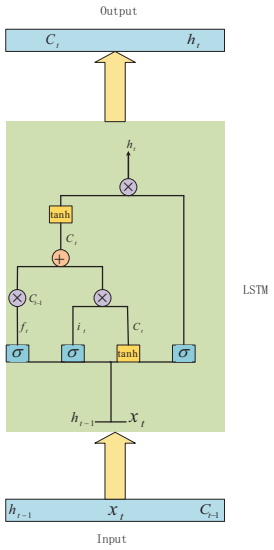


Figure 2. The LSTM component consists several gates

LSTM has its own inherent shortcomings. From the angle of the architecture of this network, the relationship of two words is greatly influenced by the distance. But in reality, distance is an inferior factor compared with the characteristics of word use and linguistic contexts. Therefore, a new feature-extraction is expected for the sake of reducing the influence of distance.

B. Encoder and Decoder

In this section, we will explain the theory of the encoder and decoder, which are basic components of the network architecture. Besides, the reason of the introduction of attention mechanism will also be given, as well as the improvement of the performance thanks to attention mechanism.

The function of the seq2seq [6]model is transferring the given sentence to another one, its key component is the encoder, decoder and the context which enables the encoder and decoder[6] to be connected. After receiving the context which was translated from the sentence in the encoder, the decoder will produce the according sentence.

In encoder, the input tokens are embedded and thus transformed into embedded vectors. Then these vectors will be conveyed into LSTM component together with the vector the Pre_hidden. The result of LSTM component will be sent into output and hidden component. Naturally, the hidden component will serve as the Pre_hidden for the next token.

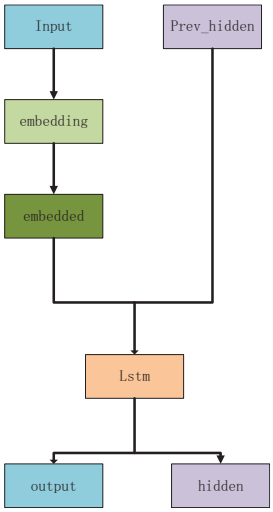


Figure 3. Illustration of specifics details of the encoder

In decoder, the input will be embedded and activated by *relu* function, and be sent together with vector from Prev_hidden layer to LSTM component. The result of LSTM component will be sent to the output component and hidden layer components. Additionally, before arriving at the output component, the result need to be processed by softmax function.

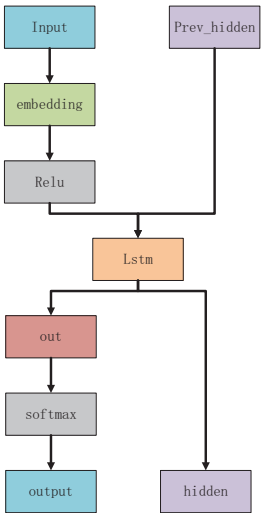


Figure 4. Illustration of details of the decoder

Fig. 5 illustrates the whole architecture of encoder and decoder. Given a sentence $\{x_1, x_2, x_3 ..., x_n\}$, each token will be input into the LSTM component. Then we gain the context, which will be sent together with x_n into the decoder. In decoder, the output of the previous step will influence the output of the next step. Finally, we gain the translation $\{y_1, y_2, ..., y_n\}$.

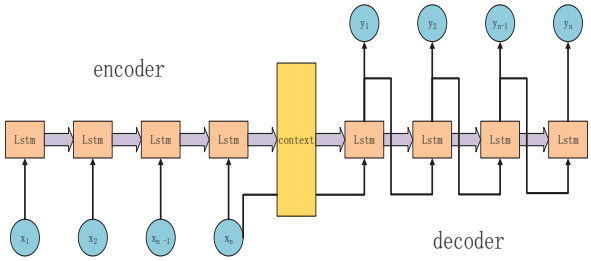


Figure 5. Application of LSTM in encoder and decoder

However, this original version of encoder-decoder is problematic because the context could not be updated by the decoder so that it will remain unchanged in the whole process. Therefore, we introduce attention mechanism to solve this problem. By giving weight to different inputs of the decoder, the context will be influenced and then changed.

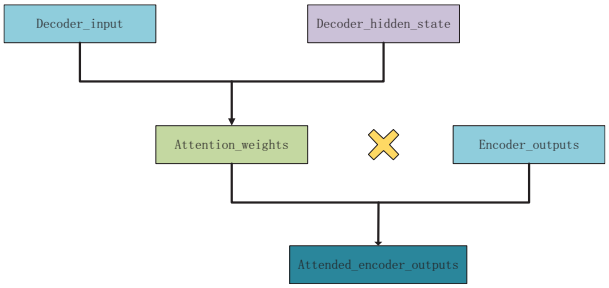


Figure 6. Details of the calculation of attention mechanism

Fig. 7 show details of the attention mechanism. At first, the decoder input is embedded and transformed into a vector. Then we could gain weight matrix with the help of this vector and Pre_hidden layer. After that, the result will be processed by Softmax[22] function which could be represented by the formula:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3}$$

So the Attn_weights matrix is produced. Finally, we need to multiply the Encoder output with the matrix we got in the previous step. Heretofore, we have achieved the attention mechanism by which we could see how much attention we will pay to each token.

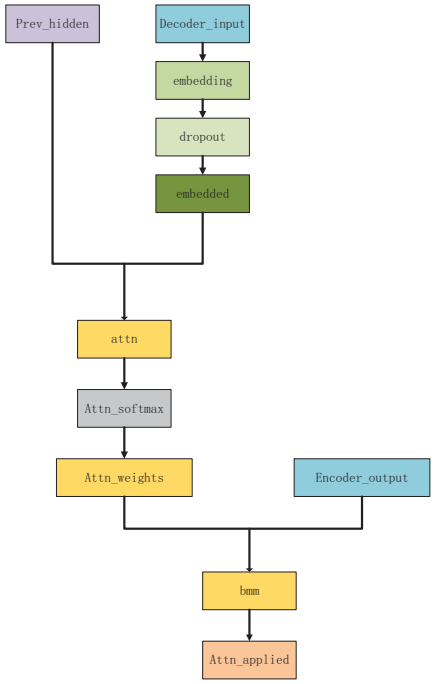


Figure 7. Illustration of details of attention mechanism

C. *Transformer*

Unlike traditional sequence transduction models which involved recurrent or convolutional neural network[7], the model called Transformer we are going to introduce entirely avoids these networks and make use of attention mechanism to improve performance in sequence transformation. The illustration of the whole structure is Fig. 8:

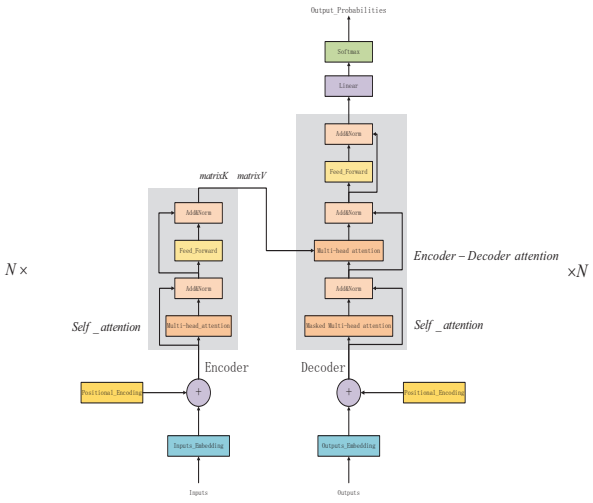


Figure 8. The architecture of Transformer-model

The whole structure consists of encoder and decoder. We input the sentence expected to be transformed at the 'Inputs' and the sentence transformed by human labor at the 'Outputs'. Encoder includes Self_attention[2] component and Feed_Forward layer, while decoder includes Self_attention mechanism, Encoder-Decoder Attention mechanism and Feed_Forward layer. Obviously, Self_attention mechanism has not been mentioned before but it is the most important part in the whole Attention mechanism, which can solve the problem about the reference of the pronouns, for example, in a sentence 'The animal didn't cross the street because it was too tired', we could figure out whether the 'it' refers to 'animal' or 'street', which is a simple task for the human but not as simple for machines. In the seq2seq model, we use addictive attention function, which is relatively simple. But in this Attention model, we use a more complex attention function called dot-product attention, which is applied in Multi-head attention component, this Multi-head attention component is also the most significant part in the Self attention mechanism.

First, we introduce some details about only 'one-head'. After the input being transformed into the embedding, we need to add a vector called 'Positional Encoding', which represents the position of the token in the sentence. Then we feed this sum to the Multi-head attention component. They are three kind of matrixes 'Q', 'K' and 'V' with the same dimension, in the 'Self attention' mechanism, which called 'Query' matrix, 'Key' matrix and 'Value' matrix. In order to gain these three kinds of matrixes, we multiply the input of 'Self attention' with matrix W^Q, W^K and W^V. Matrixes W^Q, W^K and W^V are randomly initialized and will be adjusted in the later work. After that, we gain matrix 'Z', which is also called 'attention head', through the formula:

$$\text{softmax}\left(\frac{Q\times K^{\cdot}}{\sqrt{d_k}}\right)V \tag{4}$$

where $\sqrt{d_k}$ is a scaling factor. Until now, we have finished introducing 'one-head'. Then we need to calculate different matrixes ' $Z_1, Z_2, ...Z_n$ ' through differently initialized matrixes W^Q, W^K and W^V. Then concentrate all these 'attention heads' and multiply this concentration with weight matrix W^O, then we send the product to the 'Feed forward' layer after normalized by the 'Add & Norm' layer. The illustration of the 'Multi-head attention' is shown in Fig. 9:

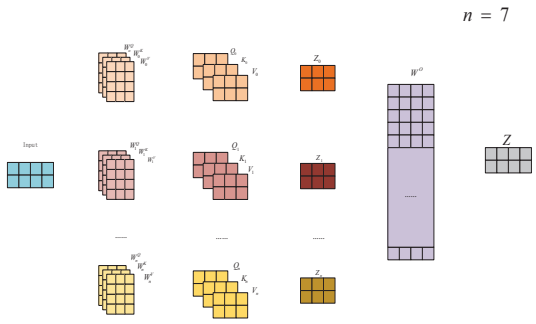


Figure 9. Illustration of calculation in multi-head

The process of the multi-head can also be expressed by the formula:

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O \quad (5)$$

where $head_i = Self\_attention(QW_i^Q, KW_i^K, VW_i^V)$

All the words in sentence will be transformed in this way in encoder. In decoder, the Self attention component and Encoder-Decoder attention component is similar to the Self attention in encoder. Of course, there exist several differences: First, the matrix 'K' and 'V' in Encoder-Decoder attention component is fed by the encoder, and matrix 'Q' comes from 'Self attention' component in the decoder. Second, the input of the Masked Multi-head attention component in decoder is from the output of the decoder. And as the word 'Masked' implies, when predicting the next words in this component, we only make use of the previous words and mask others.

Finally, we use the loss function to update the matrixes W^Q, W^K and W^V, we expected that once we input a sentence into the transformer, we can get the right output from the encoder. Compared with the LSTM, an inherent advantage of the Transformer that it is more parallelizable because it can solve sentence-level problems while LSTM could merely handle word-level problems.

The performance of the Transformer is better than the model based on convolutional and recurrent layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, this model achieves a new state of art. It is mostly the best model among the existing ensembles.

III. MODELS FOR WORD REPRESENTATION

We will illustrate the development of the models for word representation in which the feature-extracted could be applied. CBow and Skip-gram [8]model finds that there exist relationships among the tokens, unfortunately, it fails to figure out these relationships. ELMo applied the LSTM feature-extractionin its architecture, but it could not avoid the negative influence caused by the distance of the tokens. GPT integrates the transformer, which overcomes the shortcomings of LSTM, into its AR language model, while BERT combines the transformer in its AE language model. XLnet outperforms GPT and BERT by the means of taking both advantages of AE and AR and avoid the discrepancies caused by the strategies of AE language model between the pretrain and finetune.

A. CBow and Skip

A basic but significant problem in NLP is that we need to express the meaning of a word appropriately. Admittedly, a vector is a good choice. However, here comes the problem that what kind of vectors can be better. We need a kind of vectors that not only be different from but also can show relationship and similarity with each other in a certain dictionary. Therefore, we introduce a method called 'Word2vec'[8] to achieve this goal.

There are two sketchy kind of models in 'Word2vec', one is 'CBOW' (Continuous Bag of words), which is used to predict the center word with the help of context words, the other is Skip-Gram, an exact opposite version of 'CBOW'[1]. Here are illustrations of CBOW and Skip-gram models, in which there is a sentence composed of 5 words: w_{t-2}, w_{t-1}, w, w_{t+1}, w_{t+2}.

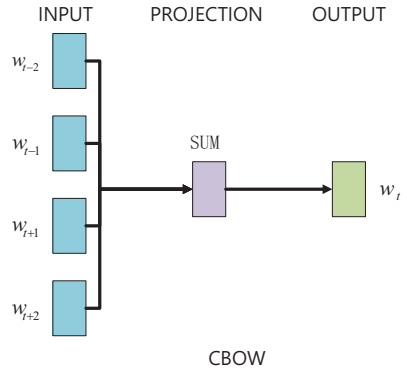


Figure 10. Illustration of the calculation of CBOW model

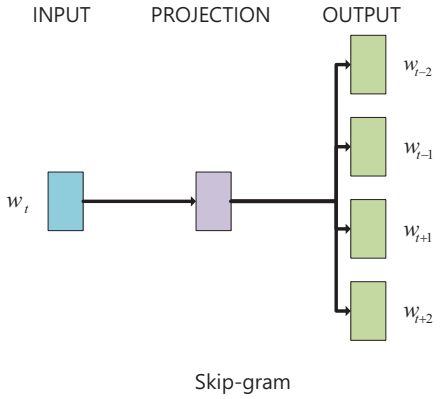


Figure 11. Illustration of the calculation of Skip-gram model

Now we will explain more details about CBOW model, as shown in Fig. 12, we suggest that the sum of the context words in a sentence is C, and the dimension of the one-hot vectors is V, and the weight matrixes are initialized into V rows and N columns. Firstly, in input layer, the words are transformed into one-hot[23] vectors, thus we gain C V-dimensional one-hot vectors. Then we multiply all of these vectors with matrix W, producing C N-dimensional vectors. Next, calculate the average of these N-dimensional vectors and then calculate the product of the average and matrix W', finally, we can gain a V-dimensional vector in the output layer, which will be compared with the vector of the center word. We will update the weight matrix W and W' with the help of gradient descent algorithm and the loss function which will calculate the difference of vector of the center word and the output vector. Eventually, the matrixes W and W' will be more mature so that we can predict center word more accurately.

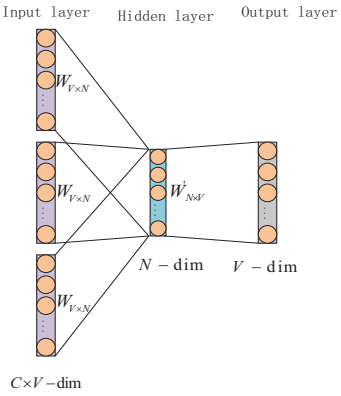


Figure 12. Model of CBOW

Also, we can display the above mentioned steps with several brief formulas:

$$Output = \frac{\sum N-\dim}{C} \times W'_{N\times V} \tag{6}$$

where $\sum N-\dim = \sum_{x=1}^{C} V-\dim_x \times W_{V\times N}$

Until now, we have finished the introduction of the CBOW model, the principle of the Skip-gram, although in the opposite direction, is similar to the CBOW. We will input the center word in the input layer and gain vectors to compare with vectors of context words, and then, likewise, we will update weight matrix W and W' in order to make this learning model more mature.

B. ELMo

In the CBOW and Skip-gram models, the methods are rudimental and the performance of art is not outstanding, therefore, we should explore more methods to improve the performance of those models. In this section, we introduce a method called ELMo (EMBEDDINGS FROM LANGUAGE MODELS) which can be added into the existing NLP structures in order to improve the performance of word representation by the means of pretraining the model with unlabeled data and then being fine tuned to other specific tasks. This model is able to take care of complex characteristic of the words and the usage of these words in various context. The sketch of this model is Fig. 13 (the sentence is: Stand up please):

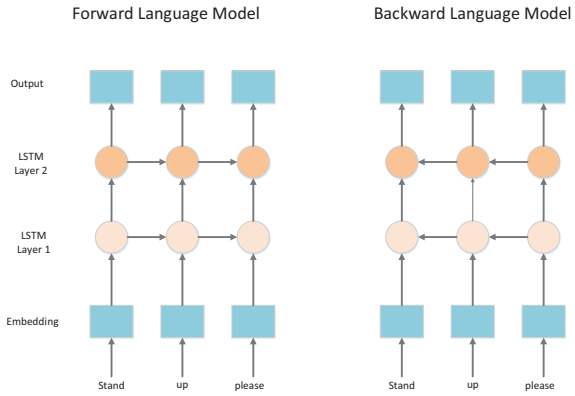


Figure 13. Illustration of LSTM applied in forward and backward language model

The structure of ELMo consists bidirectional LSTM layers, which means each words can influence both its former and latter words. Take an example of a sentence composed of N tokens $(t_1, t_2, ...t_N)$, on the one hand, each word is decided by the words in front of it, so the possibility of this sentence can be expressed by the formula:

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, ..., t_{k-1}) \tag{7}$$

likewise, each word can also be decided by the words behind it, the according formula is

$$p(t_1, t_2, ..., t_N) = \prod_{K=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, ..., t_N). \tag{8}$$

Thus, for the sake of gaining more appropriate parameters, we need to maximize the likelihood of this log function:

$$\sum_{k=1}^{N} \left(\log p\left(t_k \mid t_1, ..., t_{k-1}; \Theta_x, \vec{\Theta}_{LSTM}, \Theta_S\right) + \log p\left(t_k \mid t_{k+1}, ..., t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_S\right) \right) \tag{9}$$

where Θ_x represents token representation, $\vec{\Theta}_{LSTM}$ and $\overleftarrow{\Theta}_{LSTM}$ represents forward and backward directions of LSTM layers respectively, Θ_S represents Softmax layer.

In ELMo models, the final representation is a combination of the intermediate layer representations. We suggest the original token representation is x_k^{LM}, where k represents the specific position in the sentence. Then it will be transformed into $\overrightarrow{h}_k^{LM,j}$ where $j = 1, 2, ..., L$ after passing through the forward LSTM layer. Also, when passing through the backward LSTM layer, we can gain $\overleftarrow{h}_k^{LM,j}$. These two representations will be sent to next LSTM component to

predict t_{k+1}. Thus, the $2L+1$ representation in ELMo models according to each token t_k is:

$$R_k = \left\{ x_k^{LM}, \overrightarrow{h}_k^{LM,j}, \overleftarrow{h}_k^{LM,j} \mid j = 1,...,L \right\} = \left\{ h_k^{LM,j} \mid j = 0,...,L \right\} \quad (10)$$

where $h_k^{LM,0}$ is the token layer. We suggest the output representation in ELMo models for each token t_k is $ELMo_k = E\left(R_k; \theta_e\right)$. When it comes the simplest case, the representation merely comes from the top layer[17][18]: $E\left(R_k\right) = h_k^{LM,L}$, while in more complex cases, we can introduce softmax-normalized learned scalar weights $s = \text{Softmax}\left(w\right)$:

$$E\left(R_k; w, \gamma\right) = \gamma \sum_{j=0}^{L} s_j h_k^{LM,j} \quad (11)$$

where γ is a scalar parameter to scale the entire ELMo vector. In this way, it is clear that this model could make prediction with the help of the bidirectional tokens.

C. GPT

Compared with ELMo, which uses LSTM component in its architecture, GPT (Generative Pre-Training) achieves a more spectacular result with the help of Transformer. This model uses large corpus of unlabeled texts in pre-training and then fine-tunes in specific tasks and outperforms other previous models in 9 out of 12 tasks studied. The whole architecture of GPT is Fig. 14:

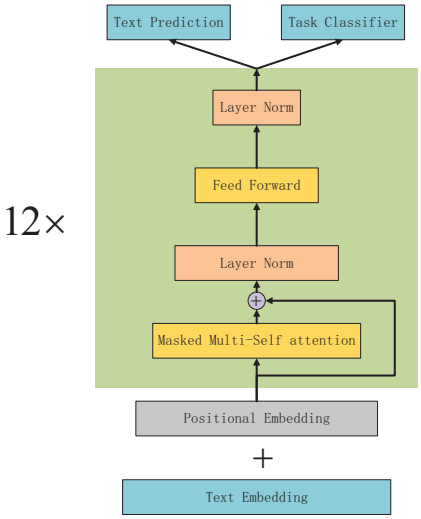


Figure 14. Architecture of GPT model

This architecture is a variant of the decoder[19] of Self-Attention, and it does not possess an Encoder-Decoder attention component. The Masked multi-head attention enables the GPT to only use tokens in the previous position of the prediction, which contributes to the application of unsupervised pre-training. Now we will specifically introduce the process of pre-training and fine-tuning.

As to pre-training, for a sentence $X = \left\{ x_1, x_2, ..., x_n \right\}$ from a unlabeled corpus, x_t is the token for prediction, k is the size of the context window, Θ is the parameter. What we need to do is to maximize the possibility:

$$L_1\left(X\right) = \sum_i \log P\left(x_i \mid x_{i-k}, ..., x_{i-1}; \Theta\right) \quad (12)$$

The process of the pre-training could be briefly expressed as three formulas:

$$h_0 = XW_e + W_p \quad (13)$$

$$h_l = transformer\_block\left(h_{l-1}\right) \forall i \in \left[1, n\right] \quad (14)$$

$$P\left(x\right) = \text{softmax}\left(h_n W_e^{\bullet}\right) \quad (15)$$

In these formulas, h_i represents the processing result of the i layer, $X = \left(x_{k-1}, ...x_{i-1}\right)$ represents the context vector of tokens, W_e is the token embedding matrix, and W_p represents the position embedding matrix.

As to fine-tuning process, we use labeled dataset. Assuming a labeled dataset C, we take a sentence $\left\{ x^1, x^2, ..., x^m \right\}$ for example, and the label of it is y. After passing through the pre-training process, The final transformer's block's activation is h_l^m, and the parameters of the added linear output layer is W_y. Thus, we could gain the formula:

$$P\left(y \mid x^1, , ..., x^m\right) = \text{softmax}\left(h_l^m W_y\right) \quad (16)$$

and in order to train this model in the whole dataset, we need to max the likelihood:

$$L_2\left(C\right) = \sum_{(x,y)} \log P\left(y \mid x^1, ..., x^m\right). \quad (17)$$

Additionally, integrating the language modeling to the fine-tuning could contributes to the generalization of the supervised model and convergence. Thus we could optimize the function:

$$L_3\left(C\right) = L_2\left(C\right) + \lambda * L_1\left(C\right) \quad (18)$$

where λ is a weight parameter.

In other specific tasks such as question answering or textual entailment, we need adapt the different situation to the pre-

trained model with the help of some kinds of descriptions so that we could input serval sentences into the model. Now we list different methods for different tasks respectively:

Textual entailment:

For textual entailment tasks, we add a delimiter token $(\$)$ between the premise and the hypothesis.

Similarity:

For similarity tasks, we input the two sentences in two possible orders with a delimiter between them, and then we gain two sentence representations h_l^m ,which will be fed to output layer after adding element-wise later.

Question/Answer:

For Question/Answer tasks, assuming a document z ,a question q ,and a set of possible answers $\{a_k\}$. And what we need to input is $[z;q;\$;a_k]$,and then they will be processed in the fine-tuning process.

There are many pros and cons in this model. Admittedly, it makes use of the Transformer, an efficient component better than LSTM. Also, it uses a large corpus for pre-training, which improves the performance of GPT in 9 of 12 datasets we study. And the pre-trained model could be fine-tuned to different tasks with texts. However, there exists a major disadvantage in GPT model. It only uses the unidirectional contexts which means that it could not learn relationship between the prediction and the tokens on the right side of it. For better performance, the modification of this model is expected.

D. BERT

Since the architectures applied in ELMo and GPT restrain the performance in fine-tuning approaches because ELMo is a feature-based approach but not a fine-tuning approache, and though GPT uses a feature-based approach, its architecture is left-to-right, which means we can merely rely on the previous tokens when making predictions. Fortunately, Bert (Bidirectional Encoder Representations from Transformers) is designed to overcome abovementioned shortcomings as it uses a fine-based approach and could take account of both left and right contexts in each layer. Thus, it can also have better performance in a wide range of tasks including question answering and language inference. The sketch of the architecture of Bert is Fig. 15:

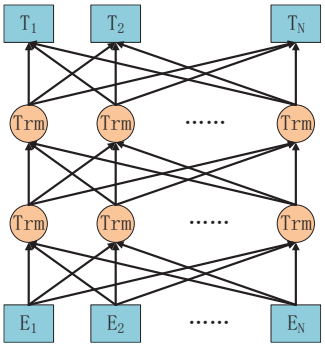


Figure 15. Architecture of Bert model. E_N represents the embedding of the Nth token. T_N represents the Nth token for prediction. 'Trm' represents Transformer component.

Compared with ELMo, both ELMo and Bert make use of the left and right contexts, but the differences are that, in ELMo, each LSTM component could merely be fed with one token, while in Bert, each transformer could be fed with all tokens. Obviously, the performance of Bert will be better. And compared with GPT, each transformer in GPT could only be fed with tokens in previous context, which means that it will undoubtedly have a worse performance than the Bert, which uses both left-to-right and right-to-left Transformer.

There are two significant parts in Bert, one is pre-training, the other is fine-tuning. As to Pre-training, it consists of two important tasks: Masked LM and Next Sentence Prediction (NSP). In the Task Masked LM, we randomly choose 15% of the tokens for prediction. The training strategy is to mask 80% of and change 10% of all tokens, and finally leave 10% of these tokens unchanged. In this way, the Bert model is able to be trained efficiently. In this way, it successfully takes advantage of the idea of CBOW which makes it take account of bidirectional contexts to achieve an unsupervised pre-training. However, it is hard to understand the reason of the specific strategy as masking 80%, changing 10% and leaving 10% of selected tokens unchanged. We have no idea about the origin of this strategy and it seems like a whim.

The other task is called NSP. The function of this task is to predict whether a sentence is following the certain sentence. This task is the advantage of the Bert because we could input one or two sentences into the model for training. When we input two sentences into the model, we add necessary information called Segment Embeddings, which could distinct these two sentences from each other, to the embeddings of the token in sentences. The Segment Embeddings of each sentence are same. Also, we add a symbol of [SEP], which means the separation, between these two sentences. The Segment Embeddings of [SEP] is as same as the ones of the first sentence, so is the 'CLS', which means the classification. So the input of the Transformer is the sum of Token Embeddings, Segment Embeddings and Position Embeddings. The illustration is Fig. 16:(A1, A2, A3 are tokens in sentence A while B1, B2, B3 are tokens in sentence B)

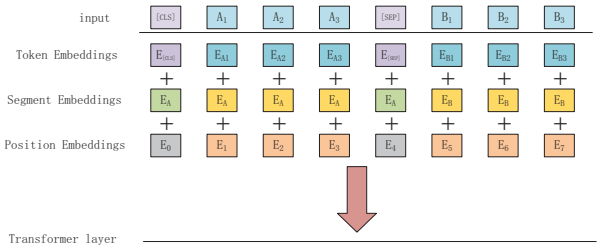


Figure 16. BERT input representation. The input embeddings are the sum of the token embeddings, segment embeddings and position embeddings.

Additionally, when choosing A and B from corpus, there is 50% of the time when B is actually following A, while in the rest of time we merely randomly choose sentence from the corpus.

As to fine-tuning part in Bert, this model can deal with 4 kinds of tasks including:

sentence pairs in paraphrasing

In this task, the input are two sentences and the output is a class label indicating whether these two sentences could be paired.

single sentence classification

In this task, the input is a single sentence and the output is a class label indicating the name of the classification of this sentence.

question-passage pairs in question answering

In this task, the inputs are a question and a paragraph where the answer could be found, and the output is the answer.

sequence tagging.

In this task, the input is a single sentence and the outputs are tags of the tokens.

The experiments of Bert are spectacular because it involves rich datasets including GLUE[9], SQuAD v1.1[10], SQuAD v2.0[11] and SWAG[12], in the meantime, the training is expensive, Training of BERTBASE costed 16 TPU in total and training of BERTLARGE costed 64 TPU chips in total. However, the effect of the rich dataset is double-edged. On the one hand, the extending of dataset is as important as the modification of the architecture and they are both efficient methods for models to make progress. On the other hand, given the high demand for hardware, it is really hard and nearly impossible to recurrent the result of the experiment, so we have to depend on the published data to evaluate the BERT, which sometimes may not seem like an advantage for us.

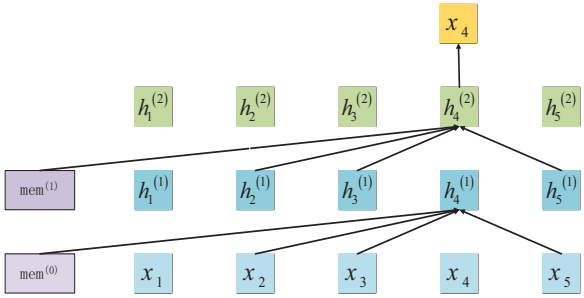
E. XLnet

The model BERT achieves a relatively outstanding performance among existing models, however, there still exist serval problems including the pretrain – finetune discrepancy caused by the masked tokens we randomly selected and the failure of learning the relationship between masked tokens. To overcome these disadvantages and retain the advantages of BERT, we introduce XLnet, a model not only providing a new method of learning bidirectional context with the help of permutations of factorization order but also integrating the idea of auto-regression (AR) model such as Transformer-XL [13]into an auto-encoding (AE) model such as BERT. As a result, XLnet achieves better performance than BERT.

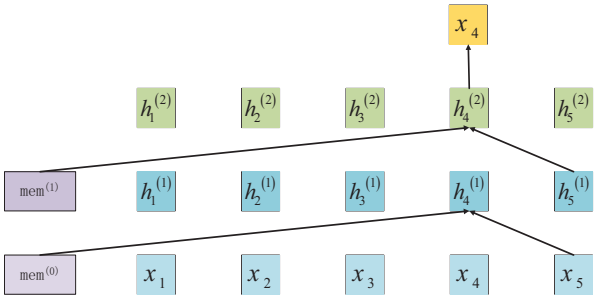
In the model XLnet, the central point that combines the advantages and overcomes the shortcomings of model AR and AE is permutations of factorization order, which we will illustrate in Fig. 17.

Suggest that the sentence consists of $\{x_1, x_2, x_3, x_4, x_5\}$,and our prediction is x_4 . h means the hidden representation.

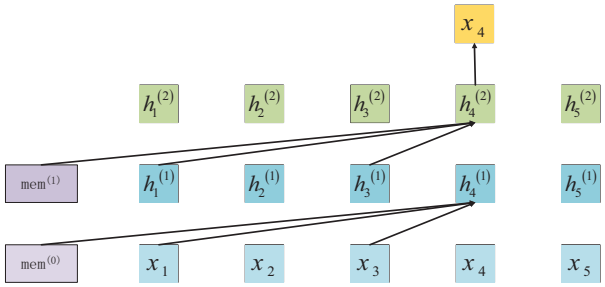
If the factorization order is: 3->5->2->4->1



If the factorization order is: 5->4->2->1->3



If the factorization order is: 1->3->4->2->5



If the factorization order is: 2->1->5->3->4

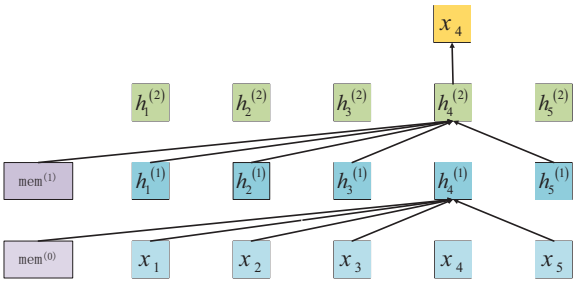


Figure 17. Illustration of the permutation language modeling objective for predicting x_4 given the same sentence but with different factorization orders.

As these diagrams imply, in each factorization order, we perform the prediction in a AR method, and we add the AE method thanks to the permutations. The advantages are that it not only has the opportunity to learn the relationship between the assumption and the tokens in both left and right positions, but also avoids the discrepancy caused by the randomly masked tokens and thus benefits the pretrain-finetune process. In order to express this process in a brief formula, we take a sentence $\left[x_1, x_2, x_3 \ldots, x_T\right]$, and we suggest $\mathbf{Z}_T$ represents the set of all possible permutations of this sentence, and let x_{zt} be the prediction and $x_{z<t}$ be the context in the previous position. Therefore, our target is maxing the possibility:

$$E_{z\sim\mathbf{z}_t}\left[\sum_{t=1}^{T}\log p_\theta\left(x_{zt}\mid x_{z<t}\right)\right]. \tag{19}$$

In the design of the architecture of XLnet, it involves 'Two-Stream Self-Attention for Target-Aware Representations'. 'Two-Stream' consists of Content stream and Query stream. Content stream is as the same as the standard self-attention, in which the token for prediction could see itself, while the Query stream could not, and this is most significant difference between the Content stream and Query stream. According to this, we introduce h, the content representation, and g, the query representation. h has access to both context and the prediction itself, while g could merely see the context but has access to the information about position of the prediction. The difference is illustrated in the diagrams Fig. 18 and Fig. 19, the token prediction is x_1 of the sentence $\left\{x_1, x_2, x_3\right\}$:

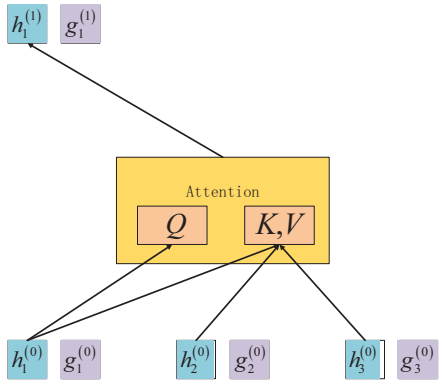


Figure 18. Content stream attention, which is the same as the standard version of self-attention

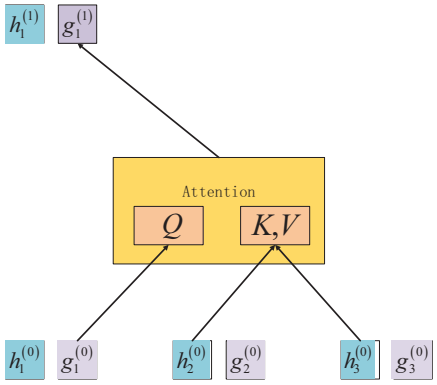


Figure 19. Query stream attention, which does not have access to the content.

The whole architecture is Fig. 20:

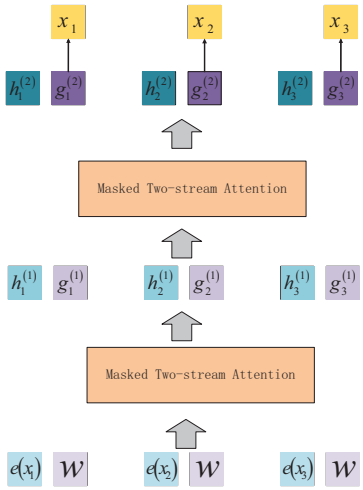


Figure 20. Overview of the permutation language modeling

In the above diagram, $e\left(x_i\right)$ means the embedding of the token in the i position of the sentence. w records the information about the position. Masked Two-stream Attention consists of abovementioned Content stream and Query stream. As to dealing with the parameter h, we need to max the likelihood:

$$p_\theta\left(X_{zt}=x\mid x_{z<t}\right)=\frac{\exp\left(e\left(x\right)^{\cdot}\ h_\theta\left(x_{z<t}\right)\right)}{\sum_{x'}\exp\left(e\left(x'\right)^{\cdot}\ h_\theta\left(x_{z<t}\right)\right)} \quad (20)$$

where $h_\theta\left(x_{z<t}\right)$ represents the production of the hidden layer. Since this parameter does not involve the information about the information about the position, we introduce g_θ to solve this problem. We need to max the likelihood:

$$p_\theta\left(X_{zt}=x\mid x_{z<t}\right)=\frac{\exp\left(e\left(x\right)^{\cdot}\ g_\theta\left(x_{z<t},z_t\right)\right)}{\sum_{x'}\exp\left(e\left(x'\right)^{\cdot}\ g_\theta\left(x_{z<t},z_t\right)\right)} \quad (21)$$

where $g_\theta\left(x_{z<t},z_t\right)$ contains the information about position.

When it comes to computation, since matrixes Q,K,V, which denote the query, key, and value in attention operation[20],were in the Masked Two-stream Attention component, for query stream where the cannot see the information about content, the formula is :

$$Attention\left(Q=g_{zt}^{(m-1)},KV=h_{z<t}^{(m-1)};\theta\right)\rightarrow g_{zt}^{(m)} \quad (22)$$

and for content stream where it has no idea about the information about position, the formula is:

$$Attention\left(Q=h_{zt}^{(m-1)},KV=h_{z\le t}^{(m-1)};\theta\right)\rightarrow h_{zt}^{(m)} \quad (23)$$

More specifically, we could illustrate the content stream and query stream in Fig. 21, we suggest the factorization order is 2->4->3->1:

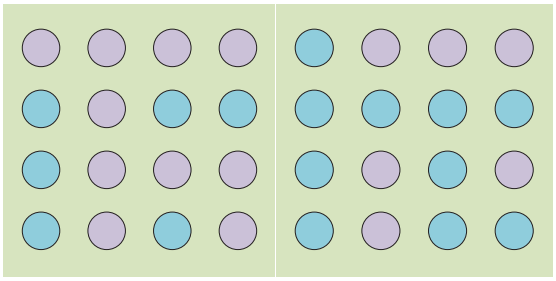


Figure 21. Illustration of the Content Stream and Query Stream in the factorization of 2->4->3->1

Obviously, in content stream, the token for prediction could see itself while it could not in query stream.

However, when faced with long texts, it is sometimes unfeasible to predict each token in sentence because it is too expensive, thus we generally introduce a hyper-parameter K, which means we only choose $\frac{1}{K}$ for prediction. In this way, the speed of training is raised and the memory will be saved.

This abovementioned architecture integrates the techniques of Transformer XL called relative positional encoding scheme and segment recurrence mechanism. Specially, it uses both the previous segments and the permutations setting. In this way, it can avoid the shortcomings caused by the [MASK] strategy in BERT, because compared with BERT, it could build relationship among tokens for prediction. This method is especially efficient in long texts, since the longer the text is, the more the masked tokens will be, and thus the more relationships among masked tokens we might fail to gain. XLnet is a more complicated and efficient model based on the transformer component and Attention mechanism and integrating advantages AR and AE methods. It is a momentous breakthrough in word representation areas.

IV. EXPERIMENTS

We summarize the performance of different models in various recently standard datasets. We will contrast these datasets and the state-of-art results as follows:

a) SQuAD2.0

SQuAD (Stanford Question Answering Dataset) is a reading comprehension dataset. As the name applies, the goal is to find the corresponding answer to the questions. The answer is contained in the texts from Wikipedia articles. The whole dataset includes two tasks called SQuAD1.1[24] and SQuAD2.0[25]. SQuAD2.0 is released as a successor of SQuAD1.1 because it introduces the situation when the answer to the question is not contained in the given texts. We will list the results of the leaderboard selected from the community of SQuAD2.0.

SG-Net [14]proposed to integrate the syntactic clues into multi-head attention mechanism so that the model could fuse both the global and attended representations. XLNet [3], as we mentioned above, used factorization order and autoregressive formulation to overcome the dependency caused by the masked positions. SpanBERT [15] proposes a method that has a little difference from the original version of BERT, instead of using token-level approach, this model masked the contiguous random spans and then trained the span boundary representations to predict the content of the masked spans. Deep contextualized word representations[4] propose a method that the word representations could be computed with the help of two-layer biLMs with character convolutions.

As we can see in TABLE I, the model integrating XLnet has the best performance compared with all other models, SpanBERT, although it has a different masking strategy from the standard version of BERT, could not outperform XLnet model. As to ELMo, since the architecture of Transformer is better than the LSTM component, its performance is relatively inferior.

TABLE I. RESULTS ON SQUAD2.0

Paper	Model	EM	F1
SG-Net: Syntax-Guided Machine Reading Comprehension	XLNet + SG-Net Verifier (ensemble)	88.174	90.702
XLNet: Generalized Autoregressive Pretraining for Language Understanding	XLNet	86.35	89.13
SpanBERT: Improving Pre-training by Representing and Predicting Spans	SpanBERT	85.7	88.7
Deep contextualized word representations	BiDAF + Self Attention + ELMo (single model)	63.372	66.251

b) GLUE

The GLUE (General Language Understanding Evaluation) is a collection of various language understanding tasks. It consists of a benchmark of nine sentence or sentence-pair language understanding tasks, a diagnostic dataset aiming at evaluating and analyzing model performance, a public leaderboard for tracking performance on the benchmark.

XLNet [3]jointly train an XLnet on four datasets including MNLI, SST-2, QNLI and QQP and then finetune the network on other tasks. Spanbert [15] followed BERT's single-task setting and then add a linear classifier on the top of the [CLS] token. BAM [16] introduces a novel method which gradually transitions the model from distillation to supervised training to enable the multi-task model to have better performance than its single-task teachers. GLUE[9] used a pair of two-layer neural language models trained on the Billion Word Benchmark[26]. The results of different models are shown in the table as bellow. The performance of XLnet model is the best, and it outperforms BERT and ELMo by a large margin. The performance of ELMo is the worst among these models.

TABLE II. RESULTS ON GLUE

Paper	Model	Score
XLNet: Generalized Autoregressive Pretraining for Language Understanding	XLnet-Large(ensemble)	88.4
spanbert: improvingpre-training by representing and predicting spans	SpanBERT(single-task training)	82.8
BAM!Born-AgainMulti-Task Networks for Natural Language Understanding	BERT+BAM	82.3
GLUE: A Multi-task Benchmark and Analysis Platform for Natural Language Understanding.	BiLSTM+ELMo+Attn	70.0

V. CONCLUSION

The update of learning methods and architectures has pushed the models to step forward. GPT outperforms ELMO because of the employment of the Transformer, but fails to compete with BERT because of the inherent disadvantages of AR language model. XLnet owns the advantages of the existing models and outperforms BERT by a large margin on 20 tasks and achieves state-of-art results on 18 tasks.

In the future, the XLnet is expected to be updated and modified. For example, XLnet records the relationships among the tokens in matrixes, which are actually graph structures. And in my opinion, it is possible to employ the graph structures in the feature-extraction of architectures of the language models.

Overall, the progress in the word representation is visual and constant. We hope to retain the advantages of existing models and achieve further breakthroughs in the area of word representation. From the perspective of the current trend, it seems that the improvement of word representation will bring convenience to downstream NLP tasks.

REFERENCES

[1] Hochreiter S , Schmidhuber, Jürgen. Long Short-Term Memory[J]. Neural Computation, 1997, 9(8):1735-1780.

[2] Vaswani A , Shazeer N , Parmar N , et al. Attention Is All You Need[J]. 2017.

[3] Yang Z, Dai Z, Yang Y, et al. XLNet: Generalized Autoregressive Pretraining for Language Understanding[J]. arXiv preprint arXiv:1906.08237, 2019.

[4] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[J]. arXiv preprint arXiv:1802.05365, 2018.

[5] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf, 2018.

[6] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014

[7] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111–3119.

[9] Wang A, Singh A, Michael J, et al. Glue: A multi-task benchmark and analysis platform for natural language understanding[J]. arXiv preprint arXiv:1804.07461, 2018.

[10] Rajpurkar P, Zhang J, Lopyrev K, et al. Squad: 100,000+ questions for machine comprehension of text[J]. arXiv preprint arXiv:1606.05250, 2016.

[11] Rajpurkar P, Jia R, Liang P. Know What You Don't Know: Unanswerable Questions for SQuAD[J]. arXiv preprint arXiv:1806.03822, 2018.

[12] Zellers R, Bisk Y, Schwartz R, et al. Swag: A large-scale adversarial dataset for grounded commonsense inference[J]. arXiv preprint arXiv:1808.05326, 2018.

[13] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860, 2019.

[14] Zhang Z, Wu Y, Zhou J, et al. SG-Net: Syntax-Guided Machine Reading Comprehension[J]. arXiv preprint arXiv:1908.05147, 2019.

[15] Joshi M, Chen D, Liu Y, et al. SpanBERT: Improving pre-training by representing and predicting spans[J]. arXiv preprint arXiv:1907.10529, 2019.

[16] Clark K, Luong M T, Khandelwal U, et al. BAM! Born-Again Multi-Task Networks for Natural Language Understanding[J]. arXiv preprint arXiv:1907.04829, 2019.

[17] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In ACL, 2017.

[18] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. CoRR, abs/1708.00107, 2017.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010, 2017.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ŁukaszKaiser,andIlliaPolosukhin. Attentionisallyouneed. InAdvancesinneuralinformation processing systems, pages 5998–6008, 2017.

[21] Graves A , Mohamed A R , Hinton G . Speech Recognition with Deep Recurrent Neural Networks[C]// 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013.

[22] Liu W, Wen Y, Yu Z, et al. Large-Margin Softmax Loss for Convolutional Neural Networks[C]// International Conference on International Conference on Machine Learning. 2016.

[23] Li L , Jiang Z , Huang D . A general instance representation architecture for protein-protein interaction extraction.[C]// IEEE International Conference on Bioinformatics & Biomedicine. IEEE, 2015.

[24] PranavRajpurkar,JianZhang,KonstantinLopyrev,andPercyLiang. Squad: 100,000+questions for machine comprehension of text. arXiv preprint arXiv:1606.05250, 2016.

[25] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. arXiv preprint arXiv:1806.03822, 2018.

[26] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. arXiv preprint 1312.3005, 2013.