TOWARD UNDERSTANDING TRANSFORMER BASED SELF SUPERVISED MODEL

Yizhou Zhang
zhangyiz@usc.edu

October 1, 2019

1 Introduction

Recently, transformer based large scale self supervised models such as BERT have drawn attention of researchers from different areas of NLP. As these models are deep neural networks with a large quantity of parameters, understanding their behavior in downstream tasks is challenging and attracts great research interest. A straightforward method of studying these neural network based models is to analyse their layer-wise representation by some algorithms like unsupervised clustering and compare the result with human's manner in solving some downstream tasks like question answering [1]. Furthermore, as attention mechanism plays an important role in transformer, it is beneficial to analyse the behaviour of different attention heads in BERT [2]. Apart from the above intuitive methods, some new aspects have been proposed proposed recently to study the identifiability of attention weights[3].

2 Layer-Wise analysis of Transformer representation in QA task

Layer-Wise analysis of Transformer representation can help people understand the function of each layer in a downstream task like question answering. In this section, we will introduce a paper of analysis of fine-tuned BERT[4] on Question-Answering datasets. In this task, BERT is trained to predict the answer of a question given the context containing some supporting facts. Figure 1 shows two samples from datasets. This paper aims at answering the following four questions:

- Do Transformers answer questions step by step in a similar manner to humans?
- Do specific layers in a fine-tuned BERT solve different tasks?
- What influence does fine-tuning have on the network's hidden state?
- Can a layer-wise evaluation help find out why and how a network gives a wrong prediction on a specific sample?

The layer-wise analysis is based on two methods: K-means clustering and edge probing task. The K-means clustering help us understand the model's behavior on a specific data sample. In each layer, clustering algorithm is performed on the token embeddings from a data sample. The result helps us understand how the network matches the tokens to close locations in the embedding space. And the edge probing tasks give us a general analysis of each layer over all data samples. The probing tasks translate the core NLP task to several sub classification tasks. In this paper, QA is translated to Named Entity Labeling (NEL), Coreference Resolution (COREF), Relation Classification (REL), Question Type Classification (QUES) and Supporting Fact Identification (SUP). To understand the function of different layers, each token's representation in each layer is extracted to train a MLP model for the probing task, as shown in Figure 2. The accuracy of a layer in a probing task indicates how much information associated with the task is carried by the corresponding layer.

2.1 Steps of BERT when handling QA task

To visualize the result of clustering, the authors first reduce the dimension of representation vectors to 2 via PCA and then run K-means algorithm on the 2D vectors. The value of K is selected based on the observation of the 2D

	SQuAD	bAbI
Question	What is a common punishment in the UK and Ireland?	What is Emily afraid of?
Answer	**detention**	**cats**
Context	**Currently detention is one of the most common punishments in schools in the United States, the UK, Ireland, Singapore and other countries.** It requires the pupil to remain in school at a given time in the school day (such as lunch, recess or after school); or even to attend school on a non-school day, e.g. "Saturday detention" held at some schools. During detention, students normally have to sit in a classroom and do work, write lines or a punishment essay, or sit quietly.	**Wolves are afraid of cats.** Sheep are afraid of wolves. Mice are afraid of sheep. Gertrude is a mouse. Jessica is a mouse. **Emily is a wolf.** Cats are afraid of sheep. Winona is a wolf.

Figure 1: Two samples of SQuAD (left) and bAbI (right) datasets. Supporting facts are printed in bold. The SQuAD sample can be solved via word matching and entity resolution while bAbI sample requires basic logic reasoning.

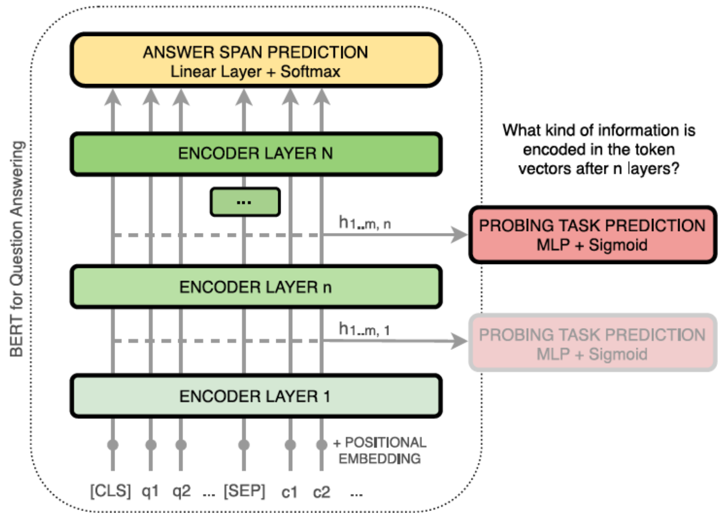
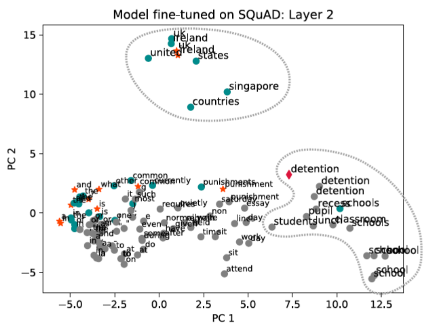


Figure 2: A illustration of Edge Probing

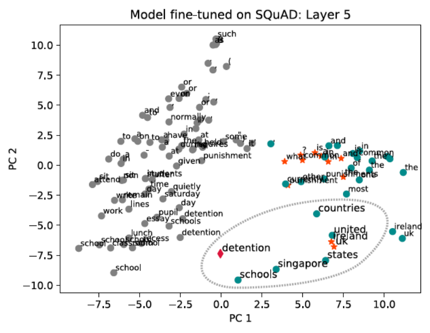
vectors. The results of two selected samples are shown in Figure 3 and 4. The observation of the clusters shows that the meanings of clustering centers vary in different layers, indicating the model handle the task through different steps. Four major steps are:

- Semantic Clustering: early layers in the model behave very similarly as word2vec and other word embedding methods. The words with similar topic like all animals form clusters as shown in the Figure 3(a) and 4(a).
- Connecting Entities with Mentions and Attributes: in the middle layers, entities and their attributes are connected based on some relations in the input as shown in the Figure 3(b) and 4(b). For example, the "detention", "schools" and several countries form a cluster as "detention" is applied in "schools" of these countries.
- Matching Questions with Supporting Facts: in this step, tokens from supporting facts form a cluster. Some samples show such behavior in earlier layers, but the accuracy scores of probing tasks show that in general the higher layers have the strongest ability.
- Answer Extraction: in the last layer, the question is extracted from all tokens.

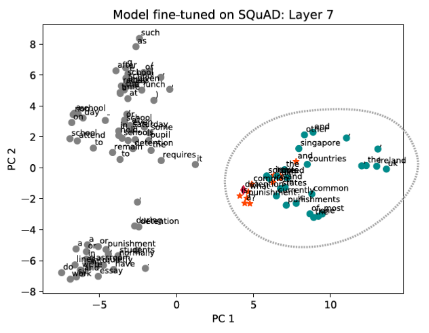
Above results are verified by the results of probing tasks on all data samples, as shown in Figure 5. The model performs in a same pattern on three datasets except when handling Question Type Classification. The Named Entity Labeling task is first solved while Coreference Resolution and Relation Classification are solved then. This phenomenon indicates that the entity associated tasks are handled around the middle layers. In the later layers, Supporting Fact Identification task is solved. Question Type Classification is an exception. A possible explanation is the difference of statistic structures in different datasets.



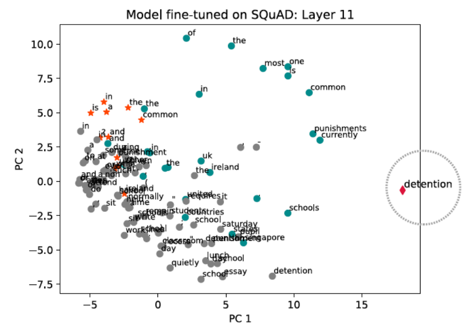
(a) SQuAD Phase 1: Semantic Clustering. We observe a topical cluster with 'school'-related and another with 'country'-related tokens.



(b) SQuAD Phase 2: Entity Matching. The marked cluster contains matched tokens 'detention', 'schools' and the countries that are applying this practice.

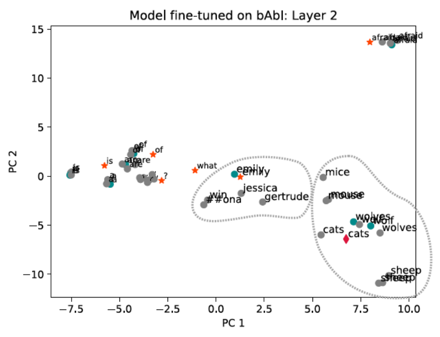


(c) SQuAD Phase 3: Question-Fact Matching. The question tokens form a cluster with the Supporting Fact tokens.

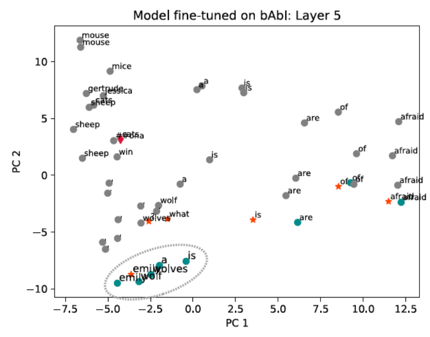


(d) SQuAD Phase 4: Answer Extraction. The answer token 'detention' is separated from other tokens.

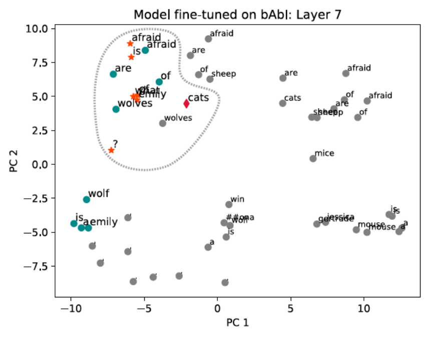
Figure 3: A sample from SQuAD dataset. The diamond means correct answer. The stars mean question tokens. Other points are context tokens.



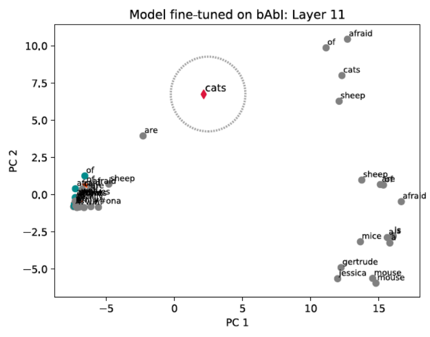
(a) bAbI Phase 1: Semantic Clustering. Names and animals are clustered.



(b) bAbI Phase 2: Entity Matching. The determining relation between the entities 'Emily' and 'Wolf' is resolved in a cluster.



(c) bAbI Phase 3: Question-Fact Matching. In this case the question tokens match with a subset of Supporting Facts ('Wolves are afraid of cats'). The subset is decisive of the answer.



(d) bAbI Phase 4: Answer Extraction. The answer token 'cats' is separated from other tokens.

Figure 4: A sample from bAbI dataset. The mark meaning is as same as 3

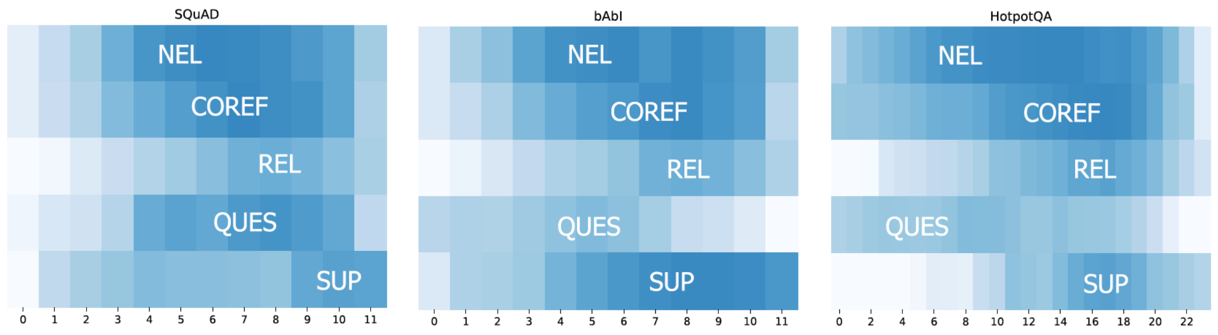


Figure 5: The results of probing tasks. Higher saturation means higher accuracy.

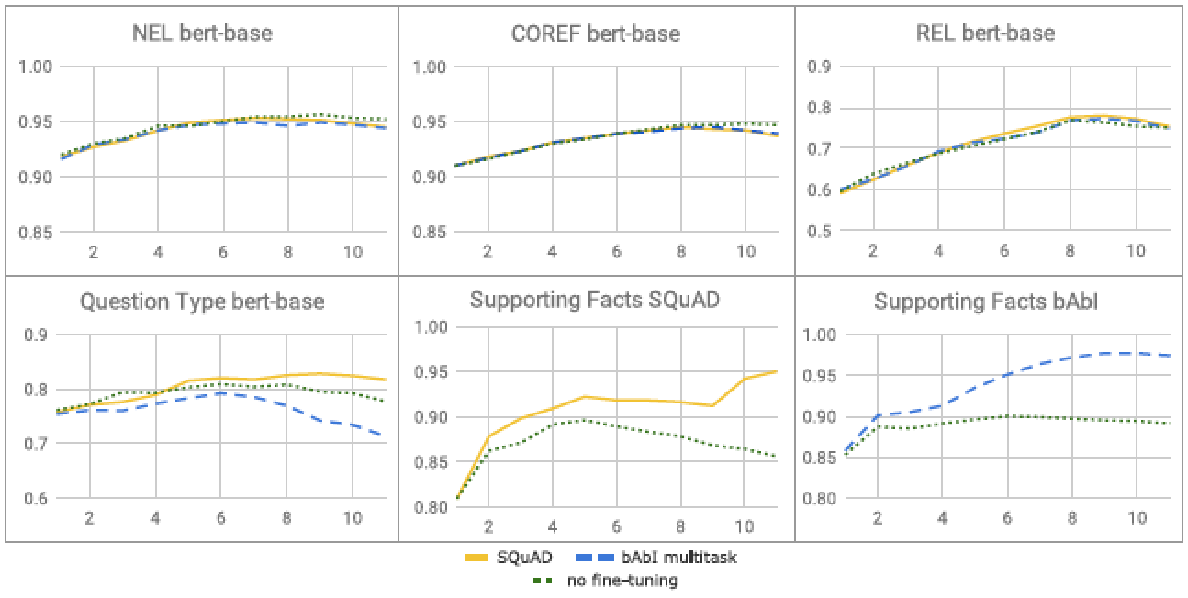


Figure 6: The comparison of models with/without fine-tuning on SQuAD and bAbI datasets.

2.2 The impact of fine-tuning

The impact of fine-tuning is shown in Figure 6 and 7. As we can see, in the first four probing tasks, the fine-tuning does not make models perform obviously better (in fact it performs worse on the HotpotQA dataset). Only in the Supporting Facts Identification tasks, which is mainly handled by the last several layers fine-tuning shows meaningful improvement. Such phenomenon indicates that the pre-trained model has already obtained enough ability to capture the information the QA task requires. The fine-tuning improves the performance by making the model only preserve the information associated with the specific task in the final layers.

2.3 Some observation from wrong cases

The authors also analyze some wrong cases. Basically, the wrong prediction can be divided into two classes. The first class is that the model gives the prediction after finishing the four steps discussed in previous subsections. However, in one or some of the steps, the model made mistakes such as wrong Supporting Fact selection. The second class is that the models failed in finishing the four steps. In some cases, the token embeddings from all the layers form topical clusters instead of associated entity cluster or supporting fact cluster, indicating all the layers remain in the Semantic Clustering step and failed in extracting other information.

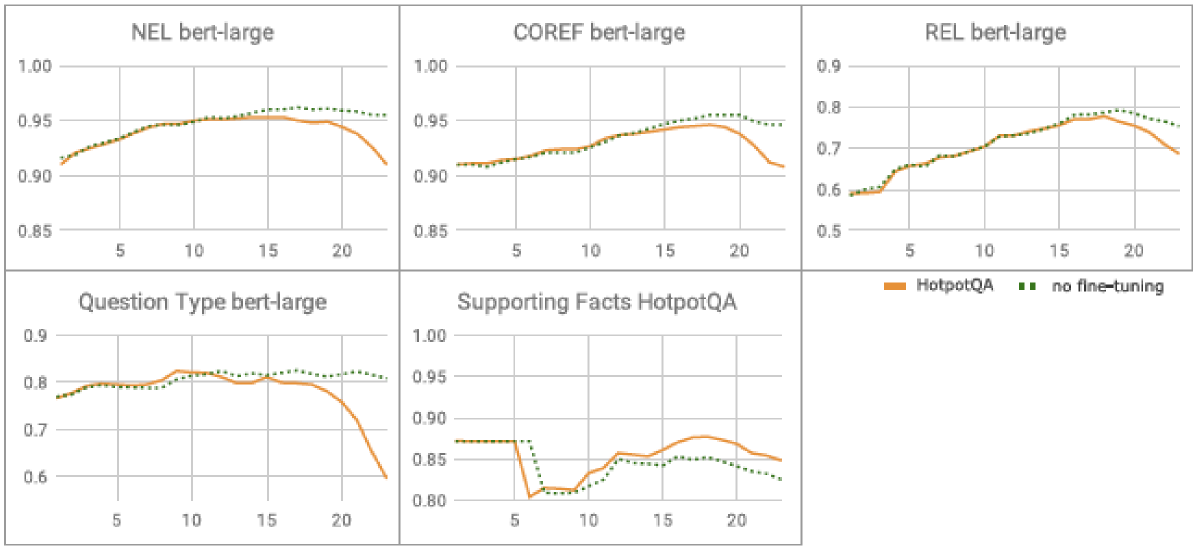


Figure 7: The comparison of models with/without fine-tuning on HotpotQA datasets.

3 Analysis of attention weights

Another important method of analyzing Transformer based pre-trained models is attention weight analysis. In this way, we are able to analyze what kind of language information or tokens an attention head mostly concentrates on. This section will introduce a paper analyzing BERT's attention mechanism [2]. Different from [1], this paper focuses on each attention head instead of each layer. The main issues discussed in this paper includes:

- The surface-level pattern of different attention heads: do they focus broadly on the text or only some specific tokens?
- The linguistic function of attention heads: the function of individual head (dependency syntax) and the function of multiple heads' collaboration (dependency parsing).

3.1 Surface-level Pattern of Attention Heads

3.1.1 Relative Position

The authors compute how often attention heads attend to the current token, the previous token and the next token. Most heads pay little attention to the current token. However, there are some heads that pay heavy attention (typically larger then 50%) to the next and previous tokens.

3.1.2 Attending to Separator Token

In BERT, input text is separated by Separator Tokens [CLS] and [SEP]. Intuitively, they are only some separator markers and have no semantic information. However, the authors found that a substantial amount of BERT's attention focuses on these BERT's separator tokens and language separator tokens (',' and '.'). The result is shown in Figure 8. As we can see, over half of the attention in layers 6-10 is paid to the [SEP] separator. One possible explanation is that separator markers may aggregate sentence level information. However, if this hypothesis holds, these [SEP] tokens should pay broad attention to all tokens in a sentence. But as shown in the bottom sub-figure, most attention of [SEP] tokens is paid to their own representation in the previous layer. Furthermore, quantitative samples shows that heads with special functions (like linking direct objects to their verbs) will attend to [SEP] if they are not applicable for the current token (e.g. the current token is a adjective). Based on this phenomenon, the authors speculate that attending to [SEP] actually means 'no-op' when the attention head is not applicable.

To verify their hypothesis, the authors calculate the importance of a token's embedding x by calculating the norm of $\frac{\partial \mathcal{L}}{\partial x}$, where $\mathcal{L}$ is the loss function in BERT's masked language modeling task. The result is shown in Figure 9. As we can see, in the later layers where the average attention to [SEP] is very heavy, the average importance of [SEP] is much

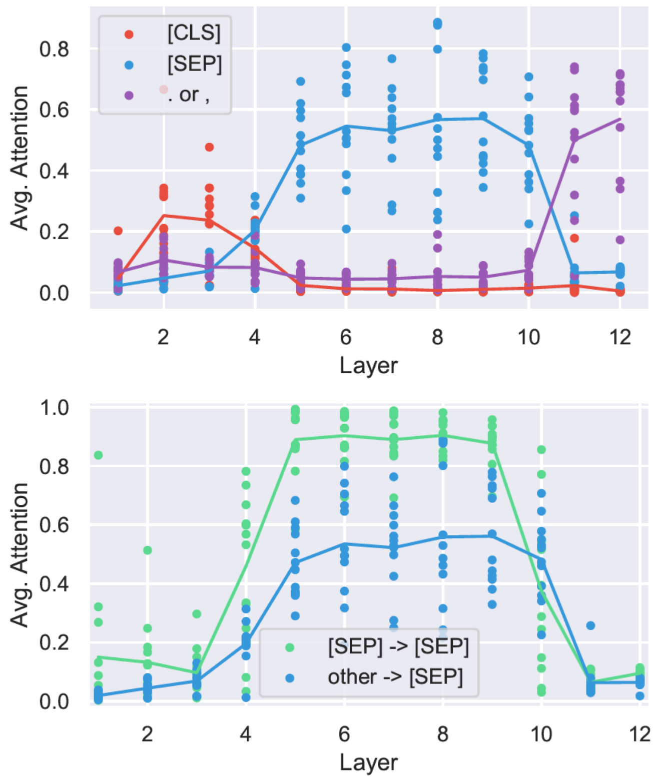


Figure 8: The top sub-figure is the average attention weigth to BERT's separator tokens and language separator tokens. While the bottom sub-figure shows the average attention paid by [SEP] token to its own embedding and other words' embedding in previous layer.

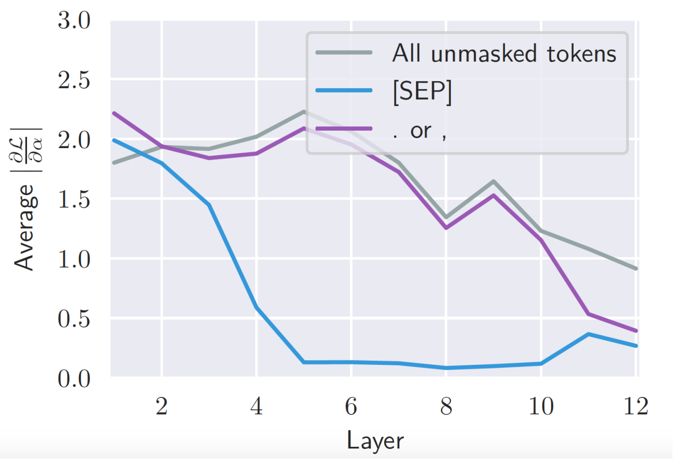


Figure 9: The gradient based importance of different tokens in different layers.

lower than all unmasked tokens and language separator tokens, indicating they actually do not carry very important information.

3.1.3 Broad Attention and Focused Attention

To measure whether an attention head tends to broadly attend to all tokens or not, the authors calculate the entropy score of all attention heads, as shown in Figure 10. The result shows that some heads have very broad attention weight. The outputs of these heads are basically the bag-of-words vectors or embedding vectors of sentences. A typical example is

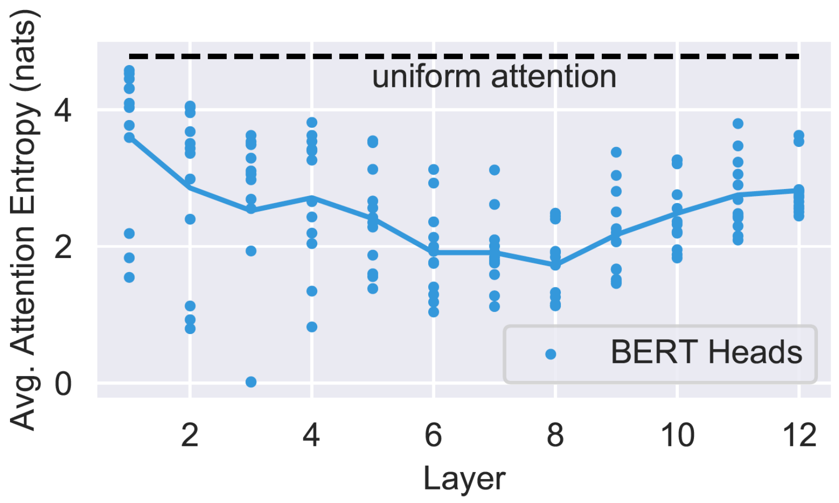


Figure 10: Caption

Dependency Relation	Attention Head	Accuracy of Attention	Baseline
All	7-6	34.5	26.3(1)
Prep	7-4	66.7	61.8(-1)
pobj	9-6	76.3	34.6(-2)
det	8-11	94.3	51.7(1)
nn	4-10	70.4	70.2(1)
nsubj	8-2	58.5	45.5(1)
amod	4-10	75.6	68.3(1)
poss	7-6	80.5	47.7(1)
auxpass	4-10	82.5	40.5(1)

Table 1: This table shows the accuracy of the best attention head and the baseline with best offset for each dependency relation. The definition of relation is from [5]. Each attention head is denoted in the form of "Layer-index", e.g. the "7-6" mean the sixth head in the seventh layer. The numbers in the brackets is the offset, e.g. the (1) mean the baseline select the first token at the right of the dependency.

[CLS], which basically aggregates the information of all the tokens in the previous sentence. This finding makes sense because [CLS] token is applied as the input of "next sentence prediction".

3.2 Linguistic Function of Attention Heads

3.2.1 Function of an Individual Head

The authors apply two probing tasks to analyze the model. Here, we only introduce the first one (dependency syntax) as the second one (correference resolution) is not as insightful as the first one. The authors evaluate the ability of each attention head to link the dependency to its head token. The result is shown in Table 1. The baseline is a rule based method which predicts the token with fixed offsets to the dependency as the head token. The result shows that the none individual attention can finish the probing task with high accuracy. However, each head may be good at capturing a specific type of dependency syntax.

3.2.2 Function of Collaborating Heads

As each individual head only handles a specific syntax, the model's overall "knowledge" about syntax should be distributed across multiple heads. So the authors design a family of probing classifiers taking attention weights of token pairs as input and train the classifiers to build the dependency parsing tree of a sentence. They design two kinds of classifiers: Attention-Only probe and Attention-and-Words probe. Both of them are linear model with a softmax normalization. The difference is that Attention-Only probe only takes attention weights as input while Attention-and-Words probe uses the inner product of parameter matrices and the GloVe embedding of tokens which can provide the

Model	UAS Head
Structual Probe	80
Right-branching	26
Distance + GloVe	58
Random Init Attn + GloVe	30
Attn only	61
Attn + GloVe	77

Table 2: This table shows the parsing accuracy of the different methods.

semantic information of the tokens to produce a word-sensitive weight for the particular attention head. The authors compare this probing classifier with three baselines:

- Right-branching baseline: always predict the head is to the dependent's right
- Distance + GloVe: use the GloVe embedding and the distance between dependency and candidate head word as input feature.
- Random Init Attn + GloVe: use the Attention-and-Words probe with random attention weights.
- Structual Probe: a SOTA method based on BERT.

The result shows that the probing classifiers have comparable performance to the SOTA method (not exactly comparable because the settings are not exactly same), indicating BERT learns some aspects syntax purely as a by-product of self-supervised training.

4 On Identifiability in Transformers' Attention Mechanism

A recent paper submitted to ICLR 2020 [3] develops a new technique to enhance the attention analyzing method's ability to identify the component of attention matrix that really influences the embedding.

The authors introduce a new concept: "effective attention weights". To understand this concept, we need some mathematical analysis to attention mechanism. A typical attention mechanism transforms the hidden representation with dimension d into a query vector Q with dimension d_q, a key vector K with dimension d_k and a value vector V with dimension d_v, defined as:

$$Attention(Q,K,V) = A \cdot V \quad with \quad A = \text{softmax}(\frac{QK^T}{\sqrt{d_q}}) \tag{1}$$

In BERT, to remain same dimension number in each layer after concatenating vectors from all attention heads together, each attention head uses a matrix $H \in \mathbb{R}^{d_v \times d}$ reduce the dimension to $d_v = \frac{d}{h}$ where h is the number of attention head in each layer. So, each attention head's output is defined as:

$$Attention(Q,K,V)H = AEW^V H = AT \tag{2}$$

where $E \in \mathbb{R}^{d_s \times d}$ is the input embeddings, d_s is the length of input sequence and W^V is the transforming matrix of value vector. $EW^V H$ is defined as T.

4.1 Null space of T

The null space describes all vectors that are mapped to 0 by T:

$$\text{null}(T) = \{\hat{x}^T \in \mathbb{R}^{1 \times d_s} | \hat{x}^T T = 0\} \tag{3}$$

This space has a special property that for $\hat{A} = [\hat{x}_1^T, \hat{x}_2^T, \cdots, \hat{x}_{d_s}^T]$ where $\hat{x}_i^T \in \text{null}(T)$,

$$(A + \hat{A})T = AT \tag{4}$$

Therefore, for any T with a null space whose dimension is not zero, there are a infinite number of $\hat{A}$. And from Rank Nullity theorem, we know the dimension of null(T) is:

$$\text{null}(T) = d_s - rank(T) \tag{5}$$

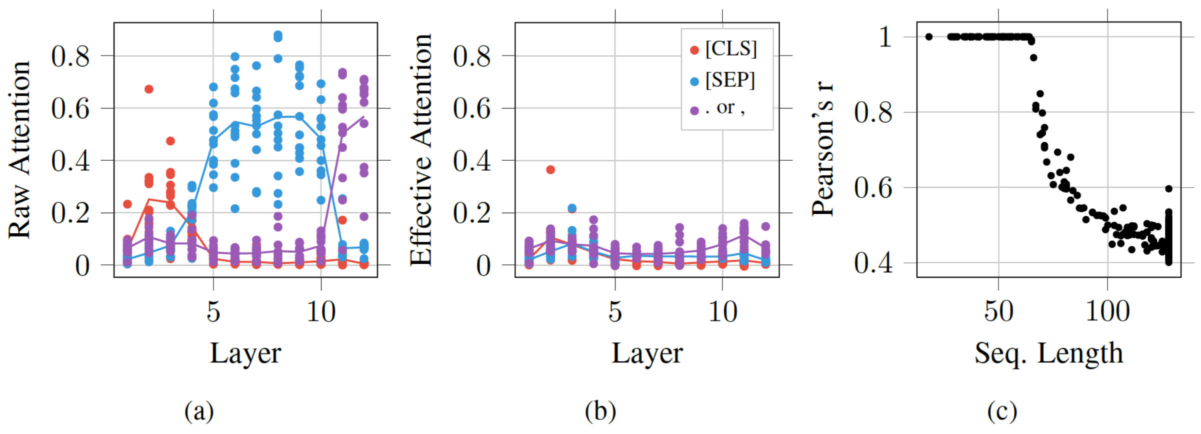


Figure 11: (a) is the result using raw attention. (b) is the result using effective attention. (c) is the Pearson correlation coefficient of effective attention and raw attention as a function of sequence length

while the upper bound of $rank(T)$ is:

$$rank(T) \leq min(rank(E), rank(W^V), rank(H)) \leq min(d_s, d, d_v) = min(d_s, d_v) \tag{6}$$

Then we have:

$$\text{null}(T) = d_s - rank(T) \geq d_s - min(d_s, d_v) \tag{7}$$

Therefore, as long as the sequence length is larger than d_v, the dimension of null(T) will not be zero.

4.2 Effective Attention

Although the null space always exists when the sequence length is large enough, one can decompose A in to the component orthogonal to the null space $A^{\perp}$ and the component in the null space $A^{||}$. Hence, the authors propose effective attention, defined as:

$$A^{\perp} = A - \text{Projection}_{\text{null}(T)} A \tag{8}$$

which is the part of the attention weights that really influence the final result. The experiment comparing raw attention and effective attention is shown in Figure 11. The authors repeat the experiments about attention weights to separator tokens in [2]. After applying effective attention, the attention weights to these tokens reduce heavily, verifying the suggestion in [2] that these attentions correspond to 'no-op'. Furthermore, the authors also run an experiment to observe the influence of sequence length to the correlation coefficient of effective attention and raw attention. As shown in Figure 11, when the length becomes larger, the dimension of null space increases.

References

[1] Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A Gers. How does bert answer questions? a layer-wise analysis of transformer representations. 2019.

[2] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. 2019.

[3] Anonymous. On identifiability in transformers. In *Submitted to International Conference on Learning Representations*, 2020. under review.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Marie-Catherine De Marneffe and Christopher D Manning. Stanford typed dependencies manual. Technical report, 2008.