Understanding “What Does BERT Look At? An Analysis of BERT’s Attention” in intent dataset

Intent Detection

Arnajak Tungchoksongchai st122458
Praewphan Tocharoenkul st122497
Nutapol Thungpao st122148

What Does BERT Look At?
An Analysis of BERT's Attention

Dataset : Wikipedia and the Penn Treebank

Model : Bert-base (12 layers 12 attention head each , Total 144 Head)

Methodology :

1) Surface-Level Patterns in Attention
   Dataset : Over 1000 Wikipedia segments (Unlabeled)
   Input sentences : [CLS]<Paragraph 1>[SEP]<Paragraph 2>[SEP]
2) Probing Individual Attention Head
   Dataset : Penn Treebank (Labeled : POS tag)
   Input sentences : [CLS]<Sentences>[SEP] , POS

Question

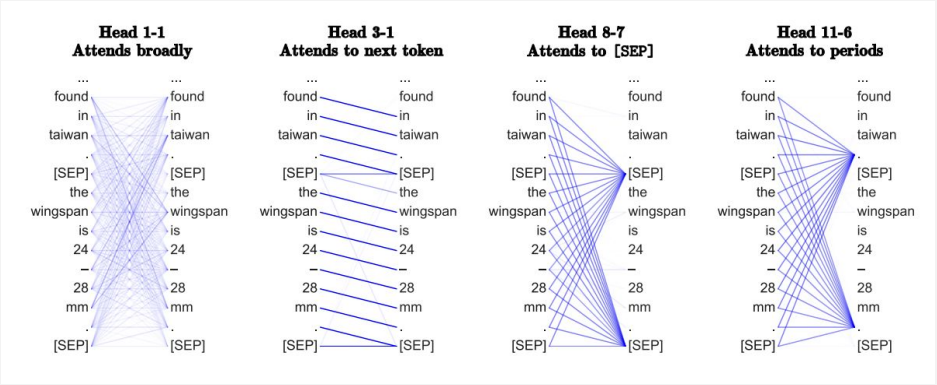
- **Instead of input with a paragraph(Wikipedia), we input it with a sentence(Hwu64). Are each head do the same role ?**

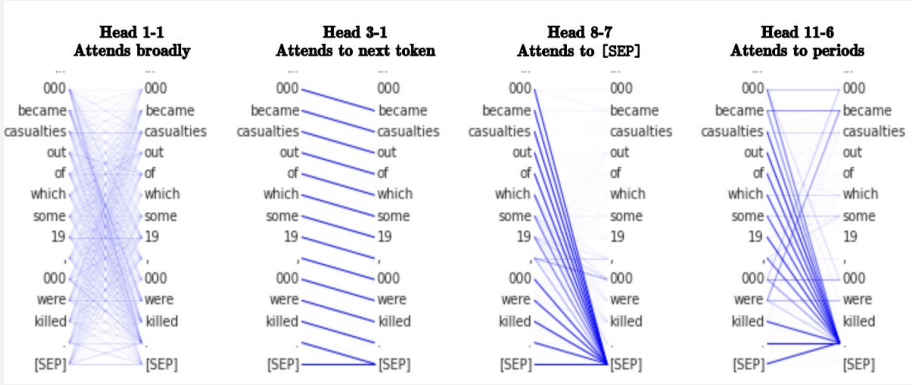
Ex. Wikipedia : he is featured in the video for faith no more \' s cover of " i started a joke " . in may 2009 , he starred in boy meets girl , a four - part drama that charts the progress of characters veronica and danny after an accident which causes them to swap bodies . [SEP] john watson in sherlock , the bbc contemporary adaptation of the sherlock holmes detective stories . the first episode of sherlock , " a study in pink " , was broadcast on 25 july 2010 to critical acclaim . for his performance in the role he won the bafta award for best supporting actor , 2011 and the primetime emmy award for outstanding supporting actor in a miniseries or

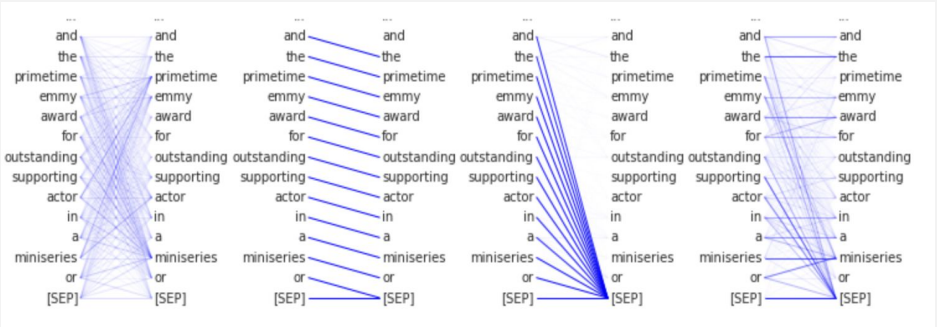
Hwu64 : please see see for me the alarms that you have set sunday morning

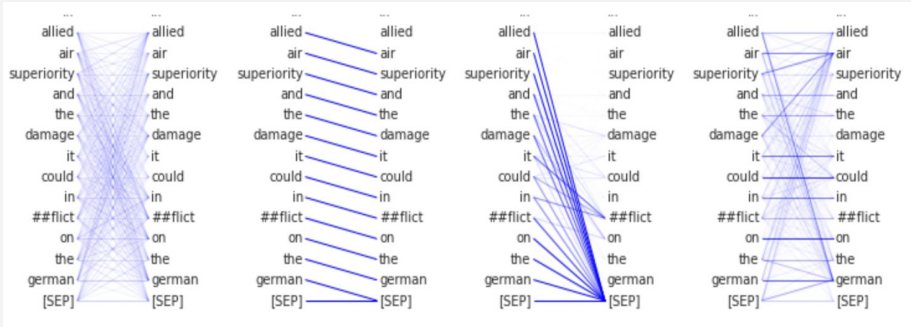
Examples of heads exhibiting the patterns :

Wikipedia vs Wikipedia (different paragraph)





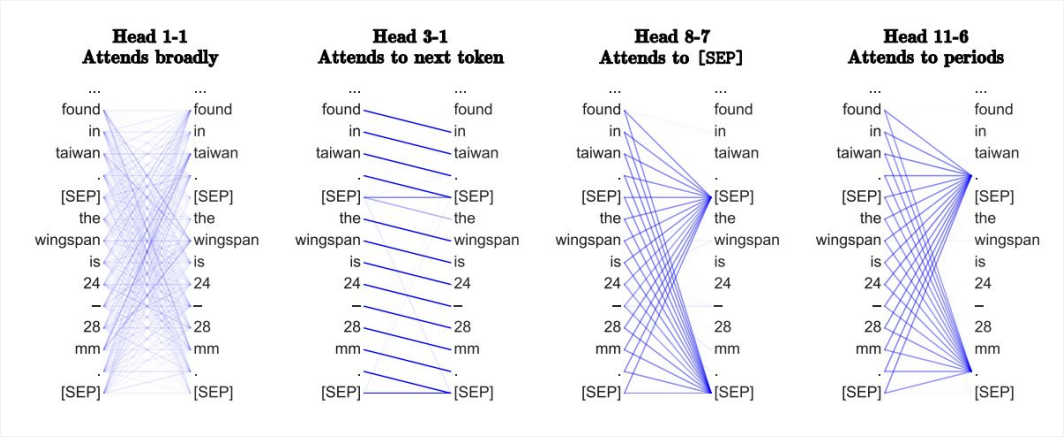




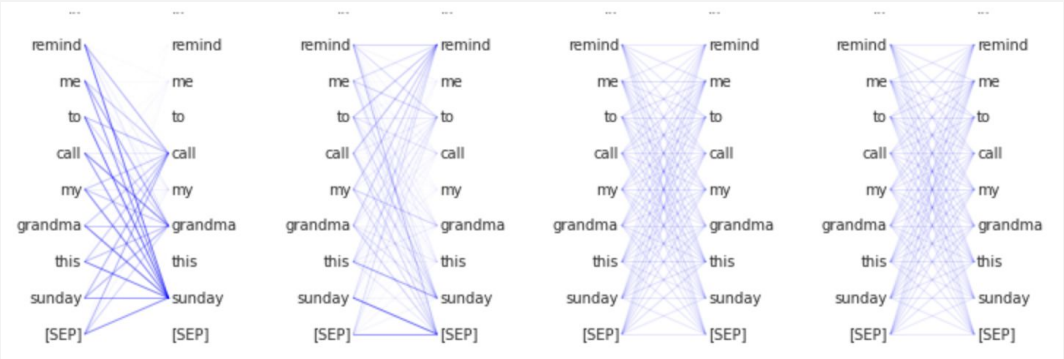
Result : Same Head have a same role

Examples of heads exhibiting the patterns :

Wikipedia vs Hwu64



Wikipedia



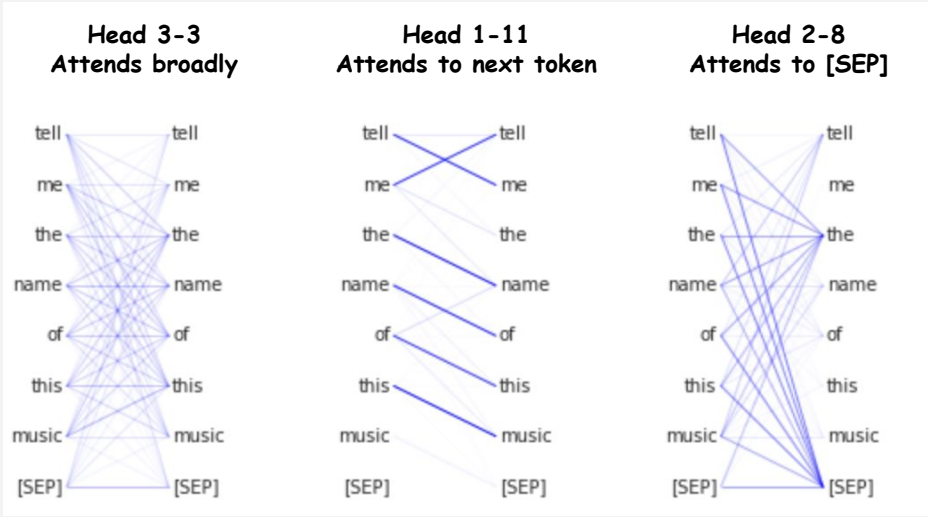
Hwu64

Result : same head do different role.

Examples of heads exhibiting the patterns :

Hwu64 vs Hwu64

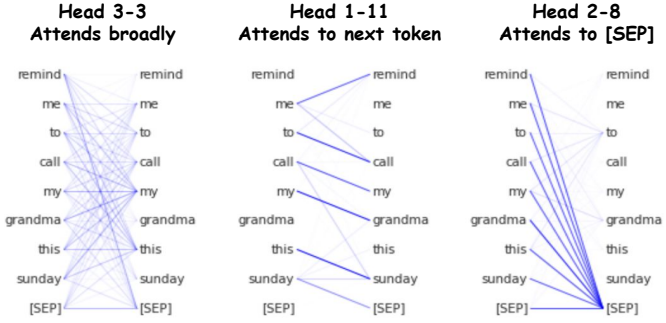
calendar_set



alarm_query

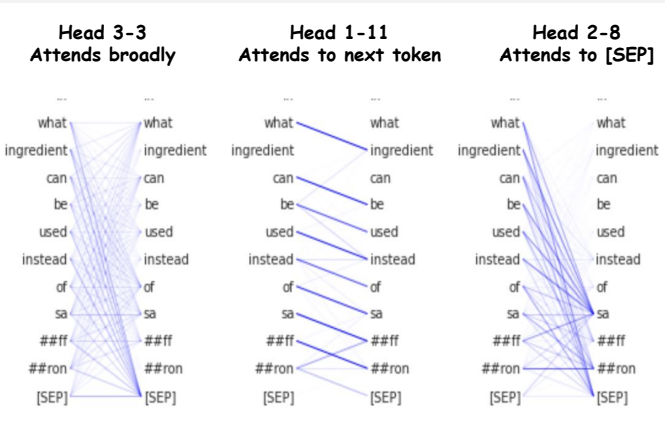
Examples of heads exhibiting the patterns : **Hwu64 vs Hwu64**

calendar_set

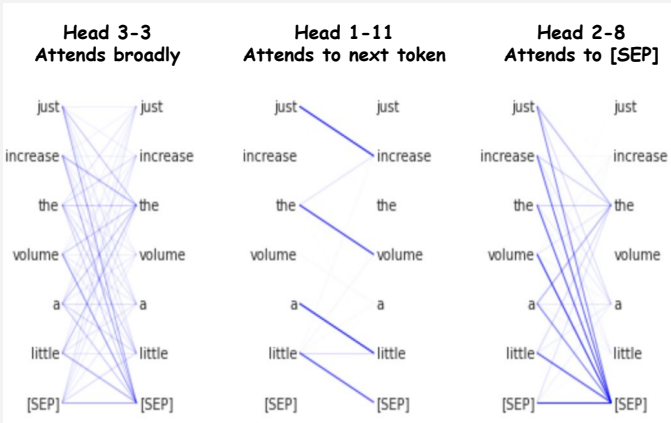


alarm_query

cooking_recipe



audio_volume_up



Compare with paper

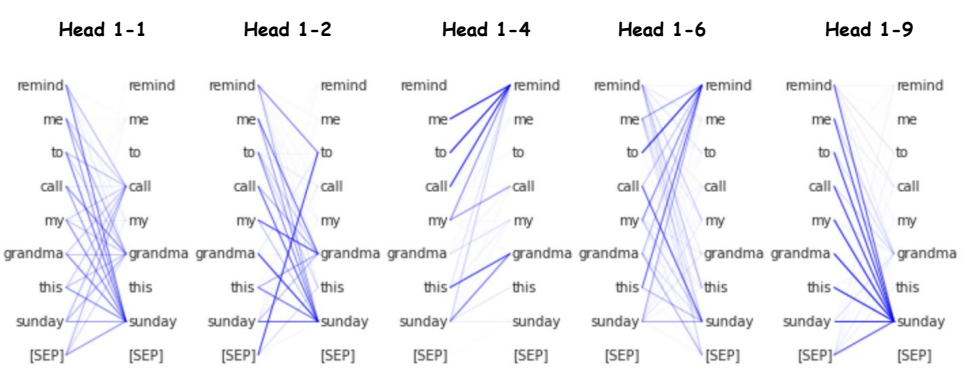
Broadly
Wikipedia : Head 1-1
Hwu64 : Head 3-3

To next token
Wikipedia : Head 3-1
Hwu64 : Head 1-11

To [SEP]
Wikipedia : Head 8-7
Hwu64 : Head 2-8

Interesting point

- **Mostly, head in layer 1 attend relating word of label**

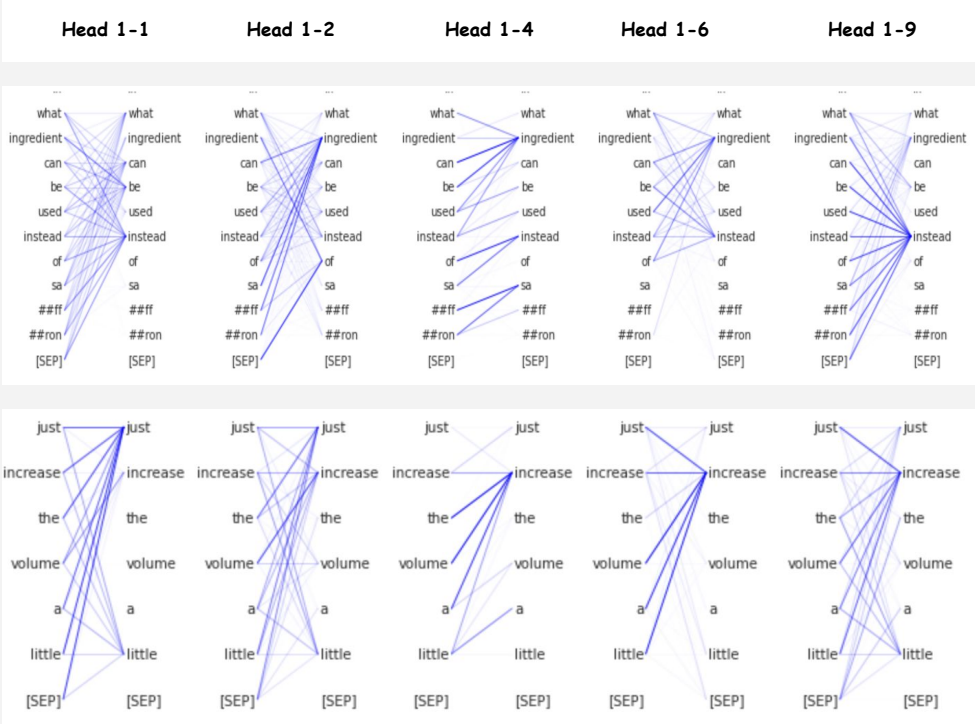


Calendar_set
Attend : remind
Grandma
sunday

Alarm_query
Attend : tell

Interesting point

- **Mostly, head in layer 1 attend to relating word of label**

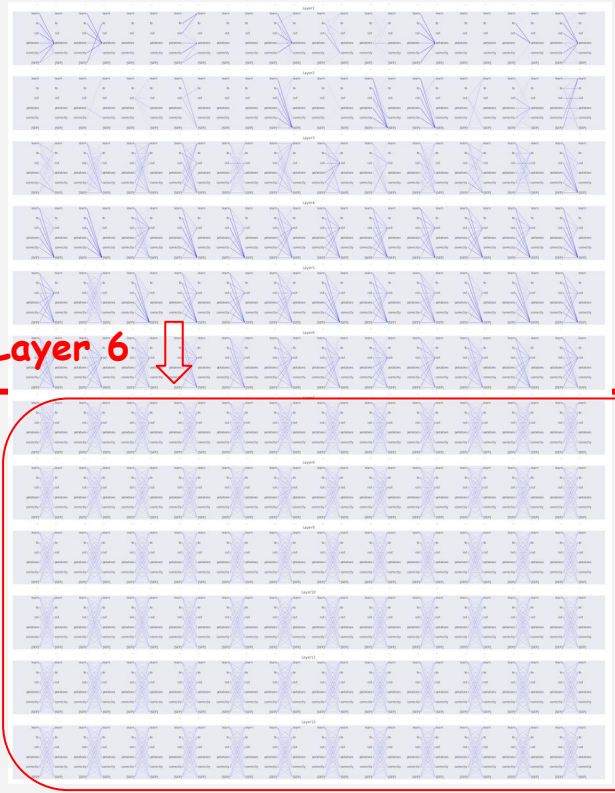


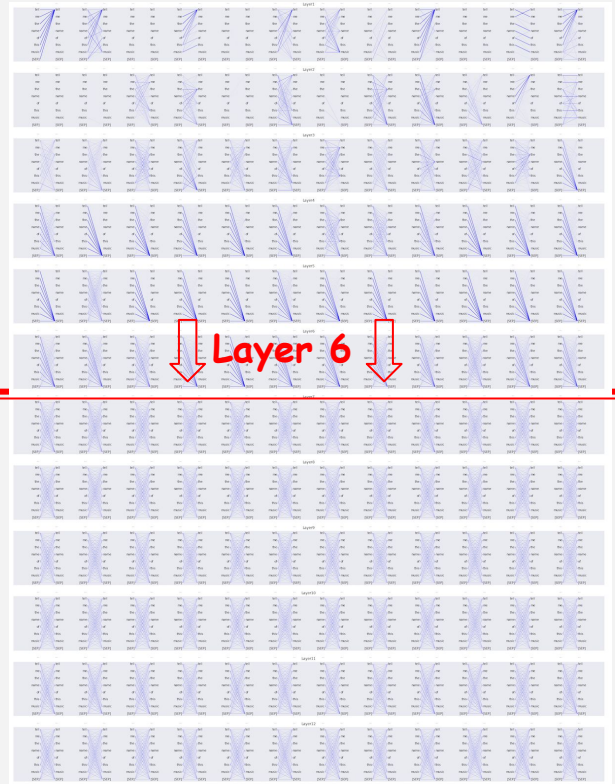
Cooking_recipe
Attend : ingredient
instead

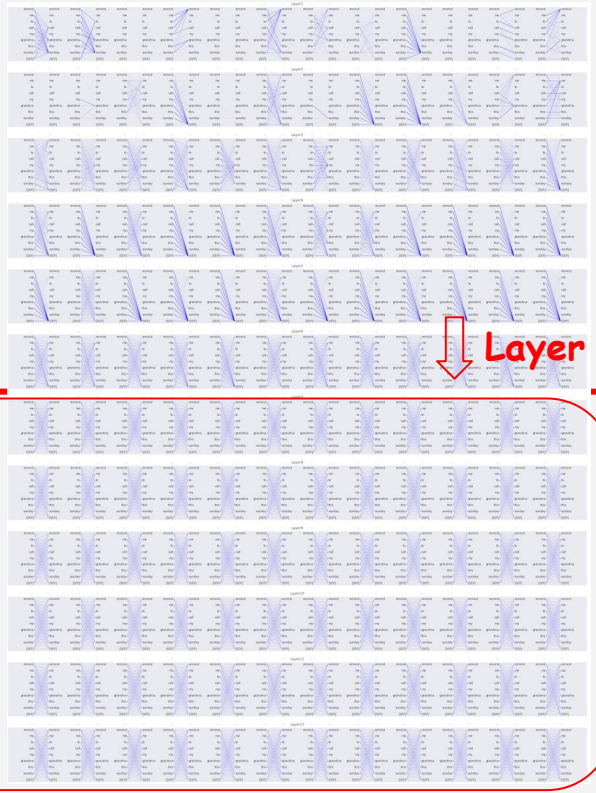
Audio_volume_up
Attend : just
increase

Interesting point

- **Mostly, head after layer 6 are broadly with low weight.**







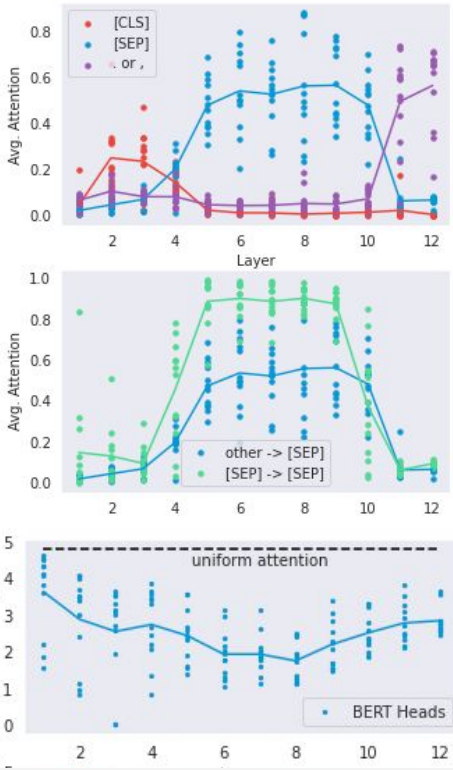
cooking_recipe

Alarm_query

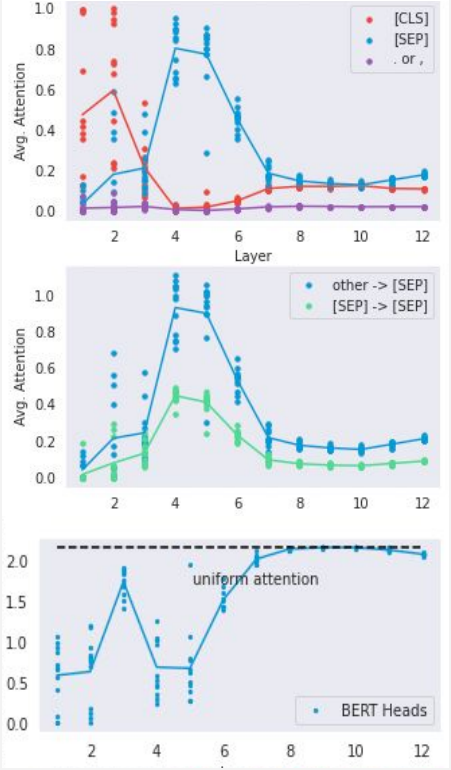
Calendar_set

Bert attention focus on a few token

Wikipedia



Hwu64

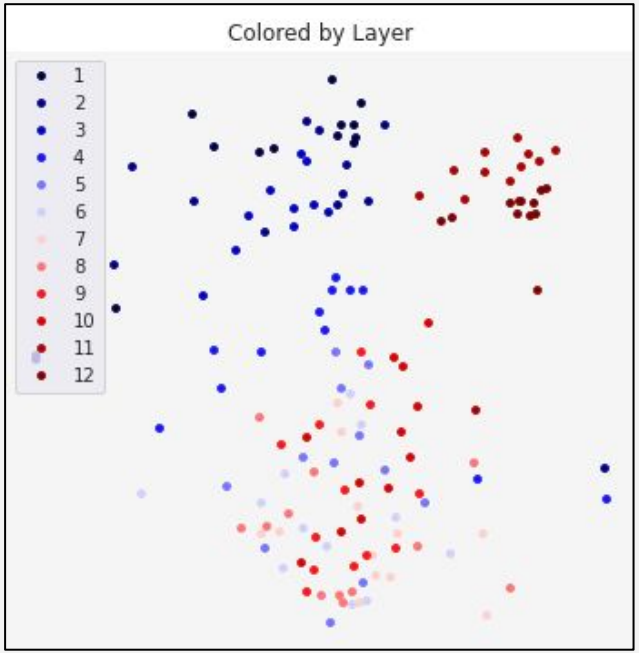


First figure :
Focus on [SEP] ->Wikipedia : layers 6 – 10
-> Hwu64 : layer 4–6
In Hwu64 layer 1–2 focuses on beginning sentence.

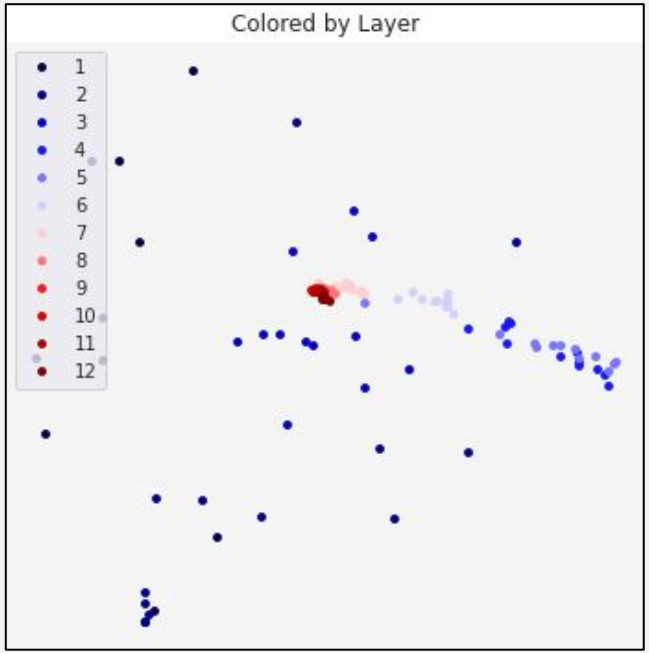
Second figure :
– Wikipedia : most head attempt to themselves.
– Hwu64 : most head attempt boardly

Last figure : compute avg. entropy
– Wikipedia : low layer have very broad attention
– Hwu64 : layer 3 and after layer 6 have broad attention

Clustering attention head



Wikipedia



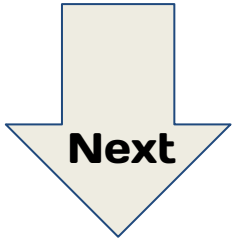
Hwu64

Are attention heads in the same layer similar to each other or different ?

- investigate these questions by computing the distances between all pairs of attention heads (Jensen-Shannon Divergence)
- Heads in the same layer tend to be close together
- In Hwu64 layer 7-12 also significantly close to each other

Following Question

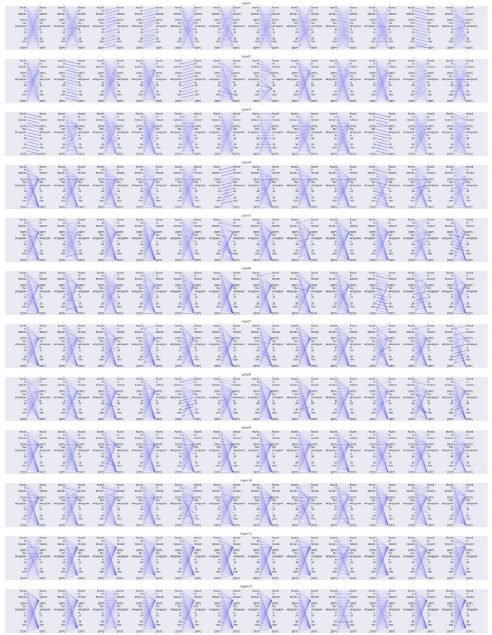
If model can attend some word of label before we classifier.
Are it increasing the accuracy ?

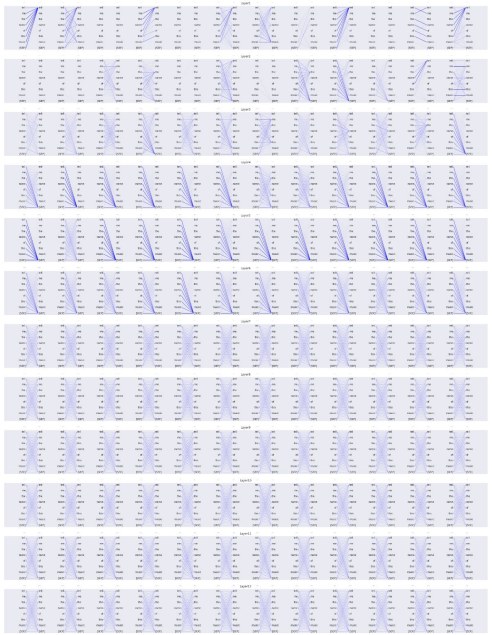


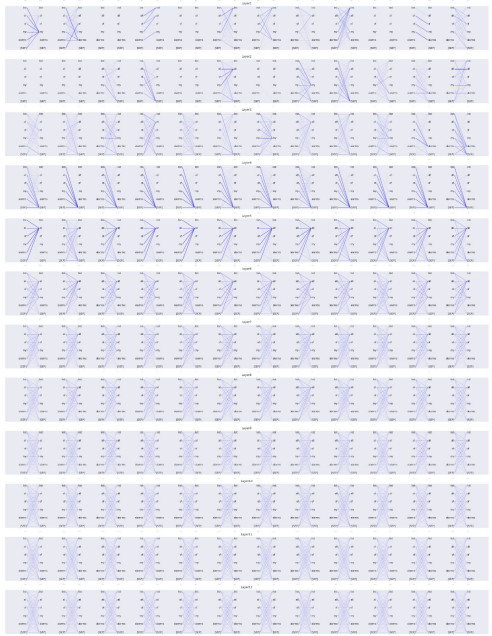
- Model use only knowledge base (pretrain weight only) -> Classifier
    - Model train in specific dataset -> Classifier

Which one is better ?

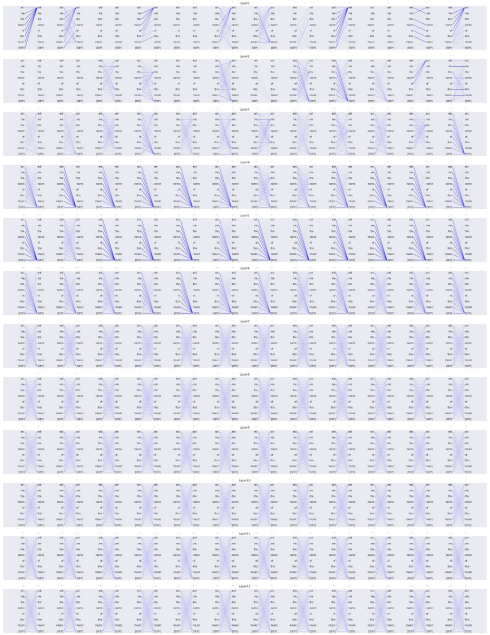
Wikipedia vs Intent

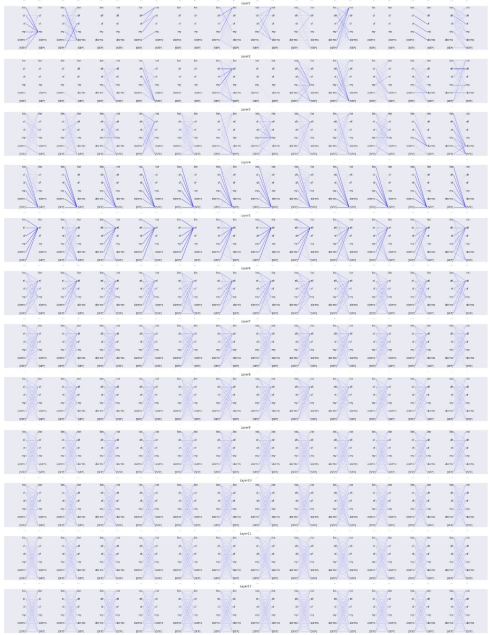




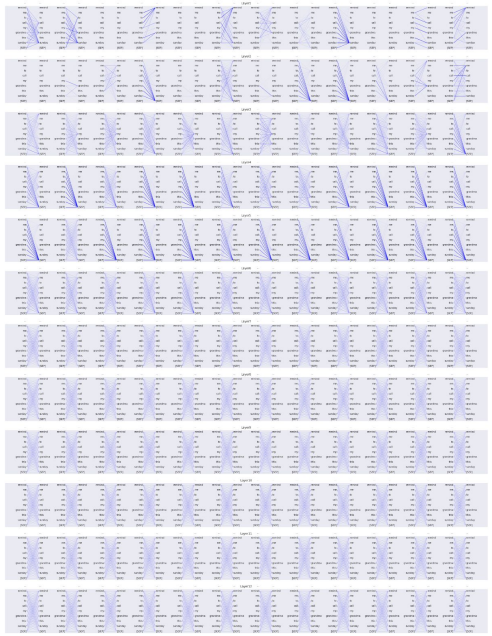


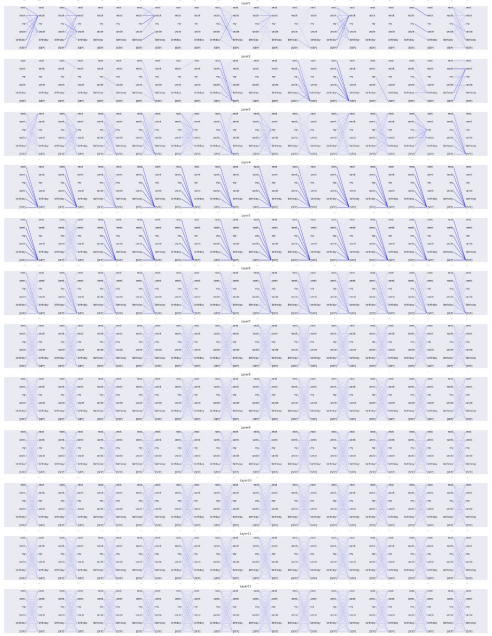
Alarm_query

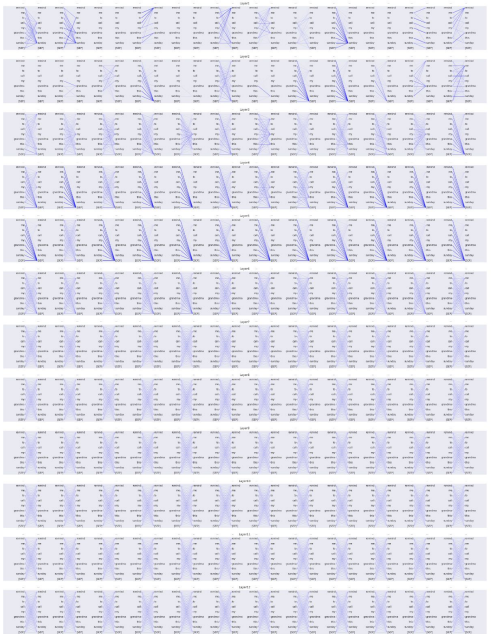


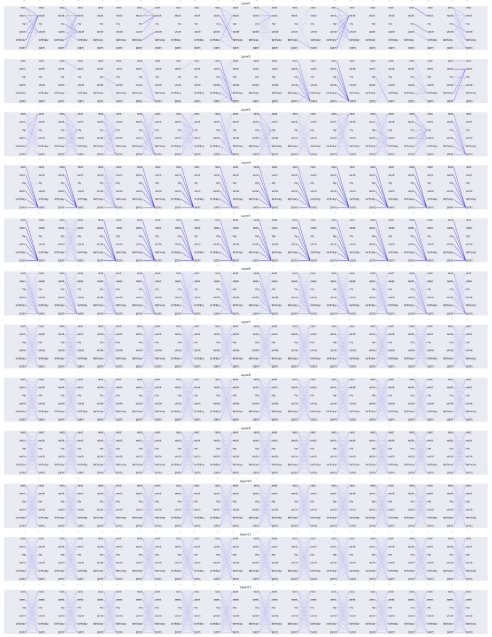


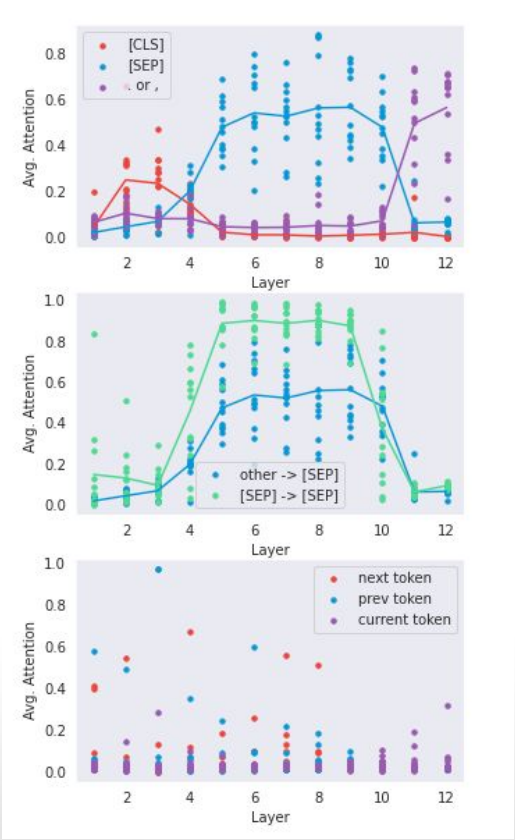
Calendar_set

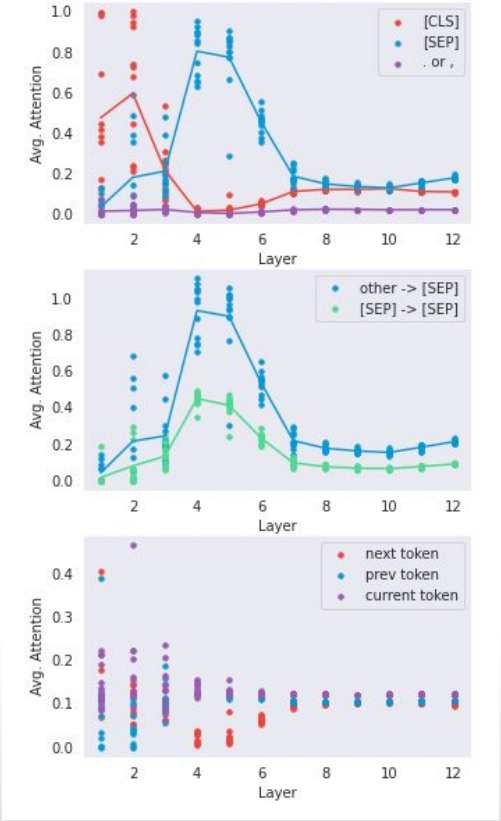


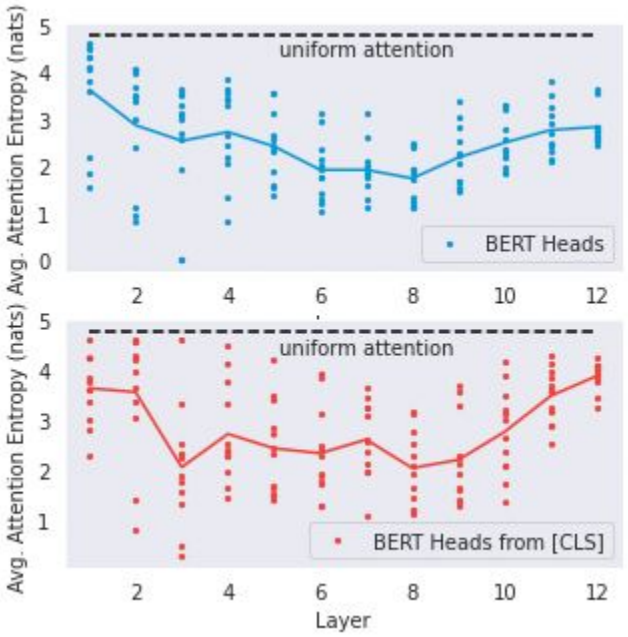


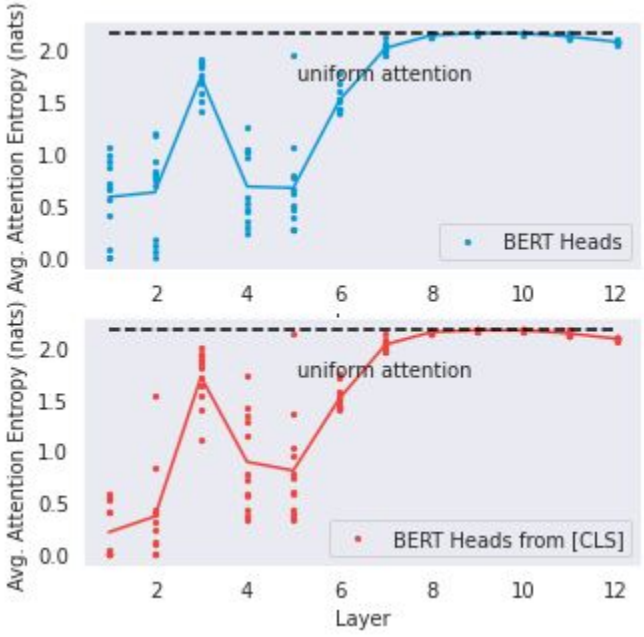


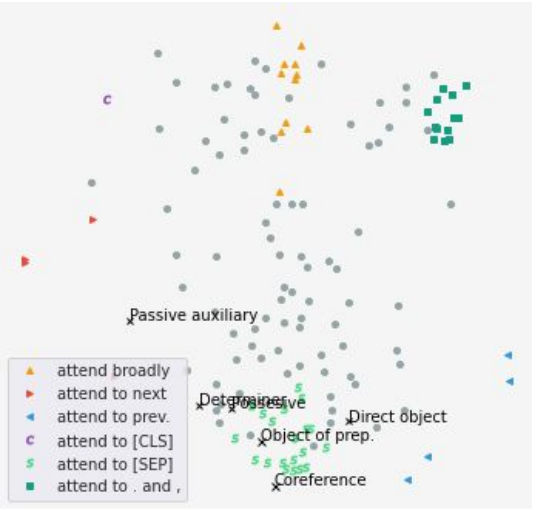


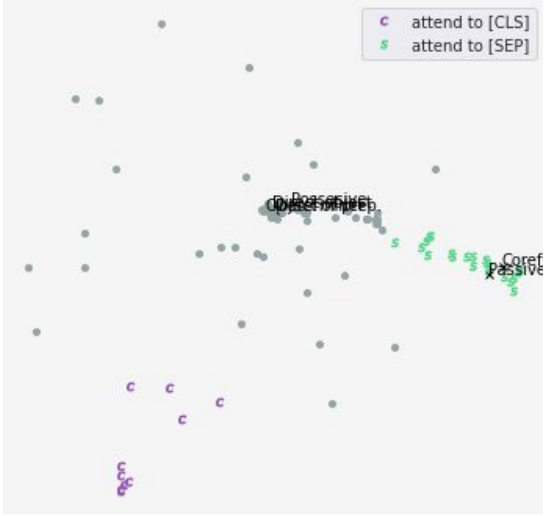

[CLS]
[SEP]
. or ,
Avg. Attention
Layer
other -> [SEP]
[SEP] -> [SEP]
next token
prev token
current token


[CLS]
[SEP]
. or ,
Avg. Attention
Layer
other -> [SEP]
[SEP] -> [SEP]
next token
prev token
current token

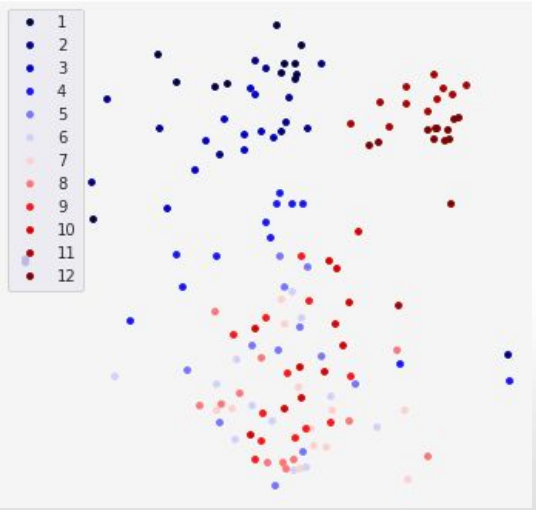

uniform attention
BERT Heads
uniform attention
BERT Heads from [CLS]
Avg. Attention Entropy (nats)
Layer


uniform attention
BERT Heads
uniform attention
BERT Heads from [CLS]
Avg. Attention Entropy (nats)
Layer

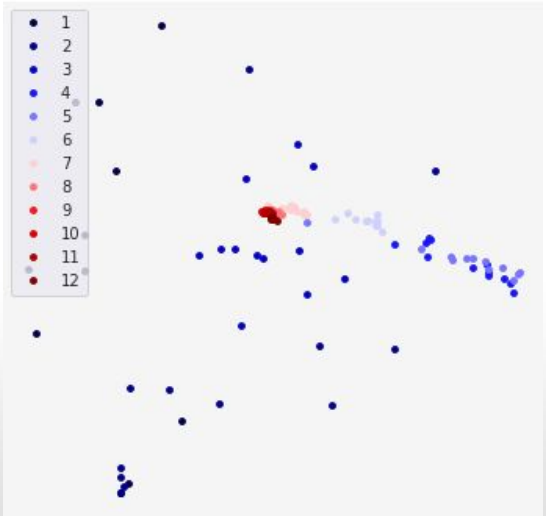




Colored by Layer



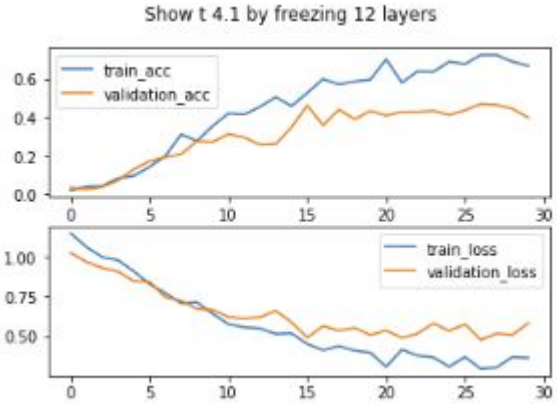
Colored by Layer



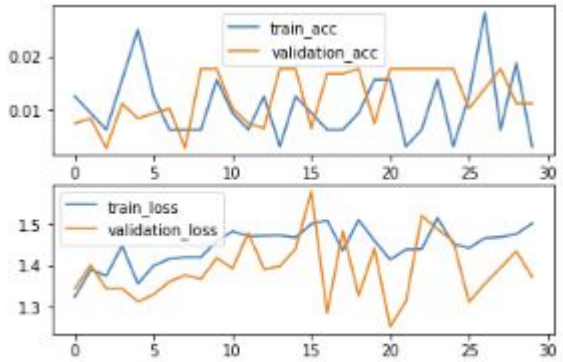
Freezing layers

Freezing layers means disabling gradient computation and backpropagation for the weights of these layers. This is a common technique in NLP or Computer Vision when working on small datasets as it usually reduces the chances of your model to overfit.

Freezing layers vs Unfreezing layers

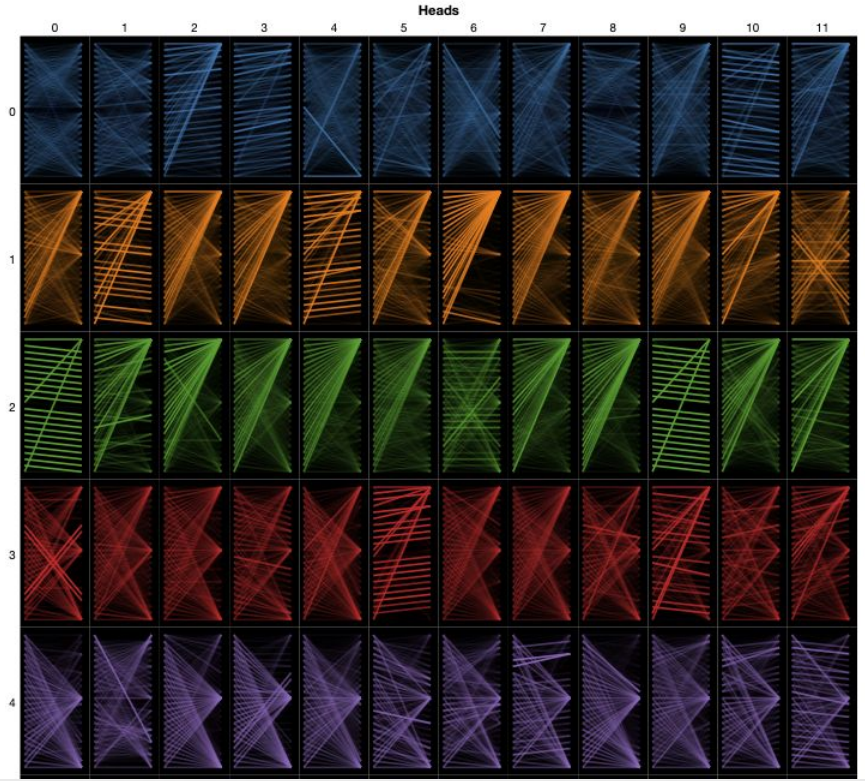


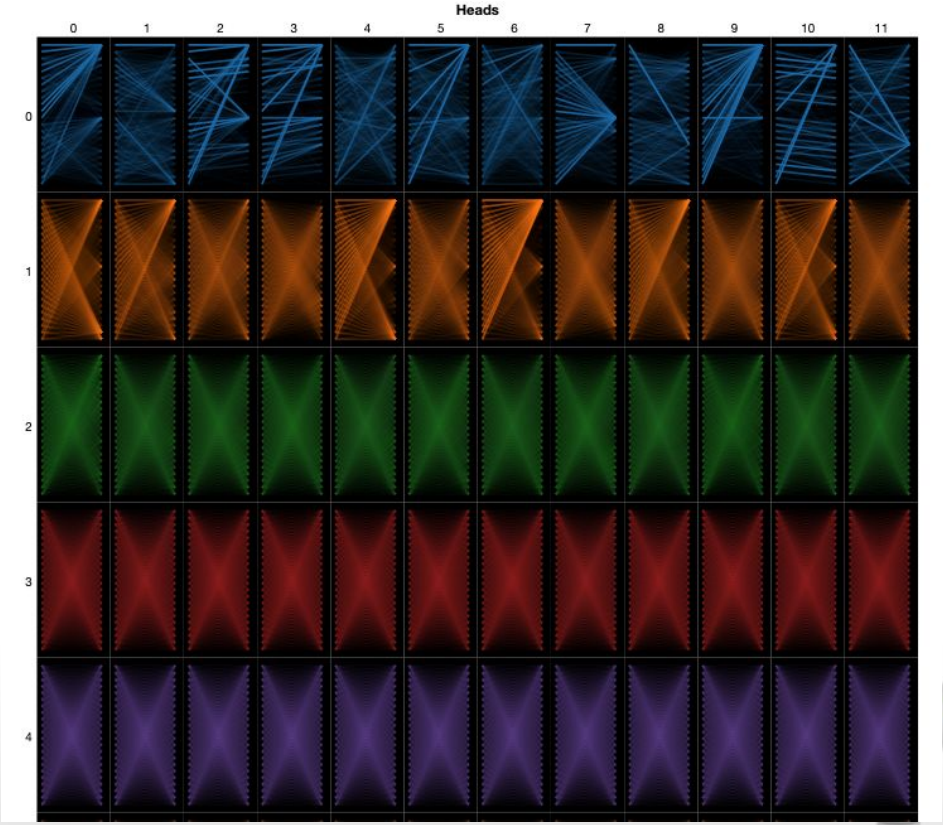
Accuracy : 40.985130111524164 %



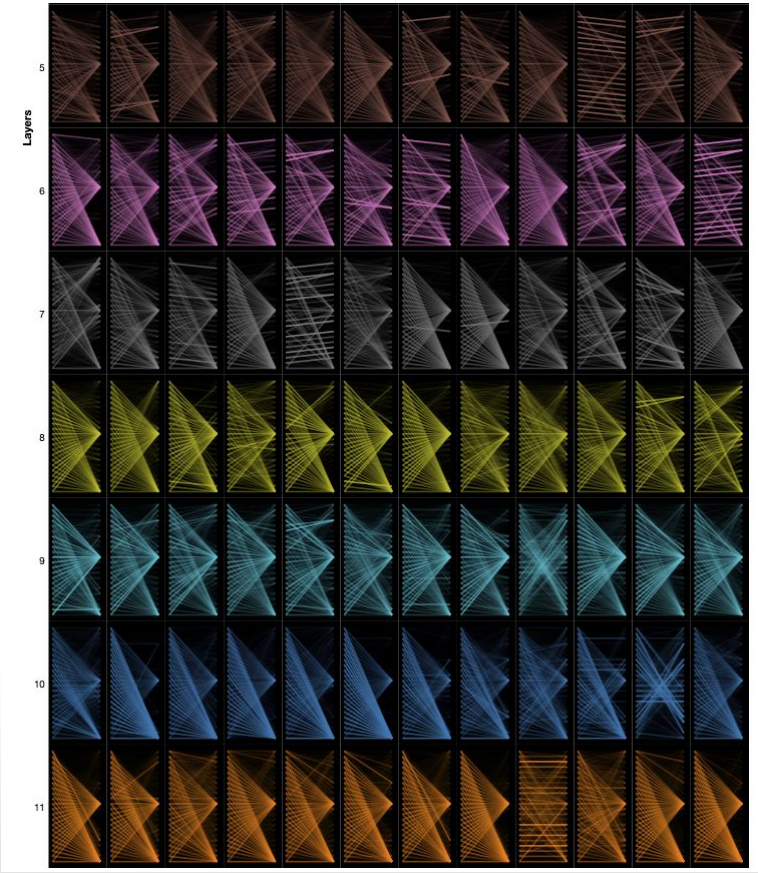
Accuracy : 1.1152416356877324 %

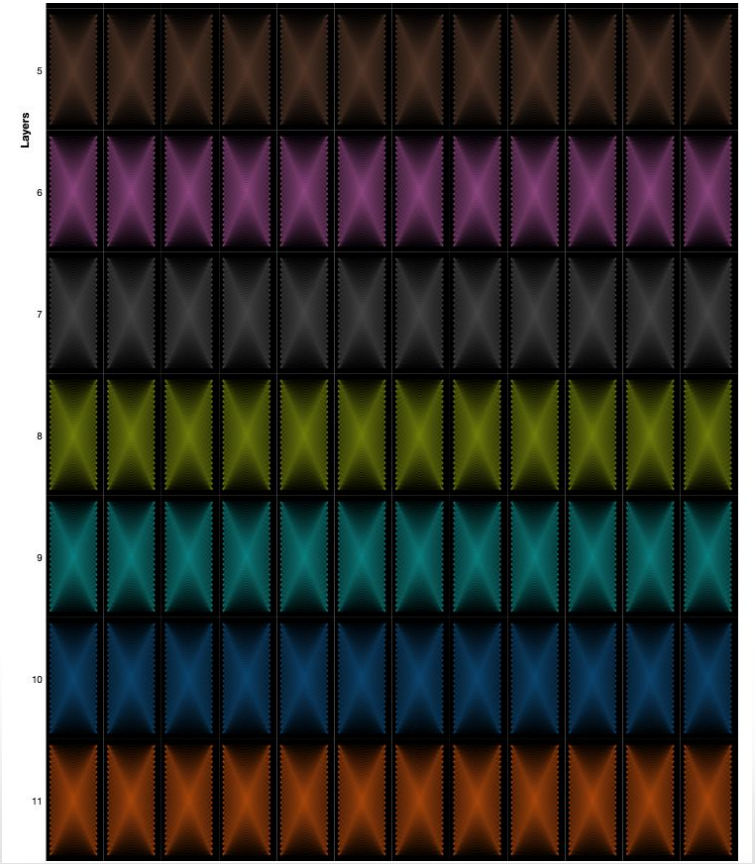
Freezing layers vs Unfreezing layers



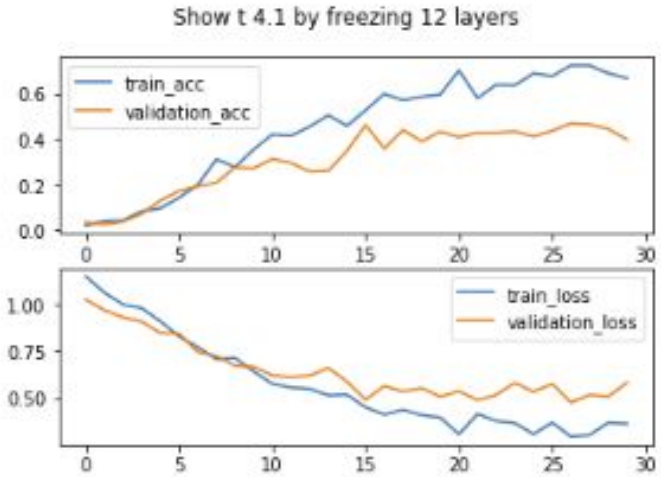


Freezing layers vs Unfreezing layers

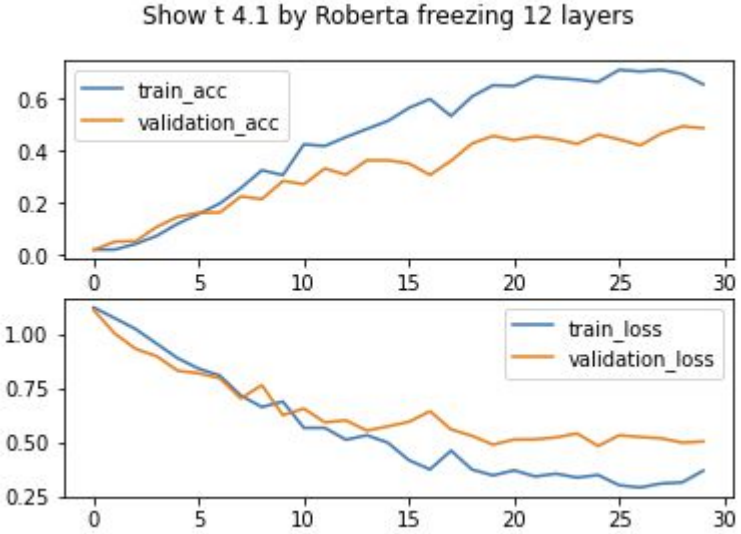




Bert vs Roberta

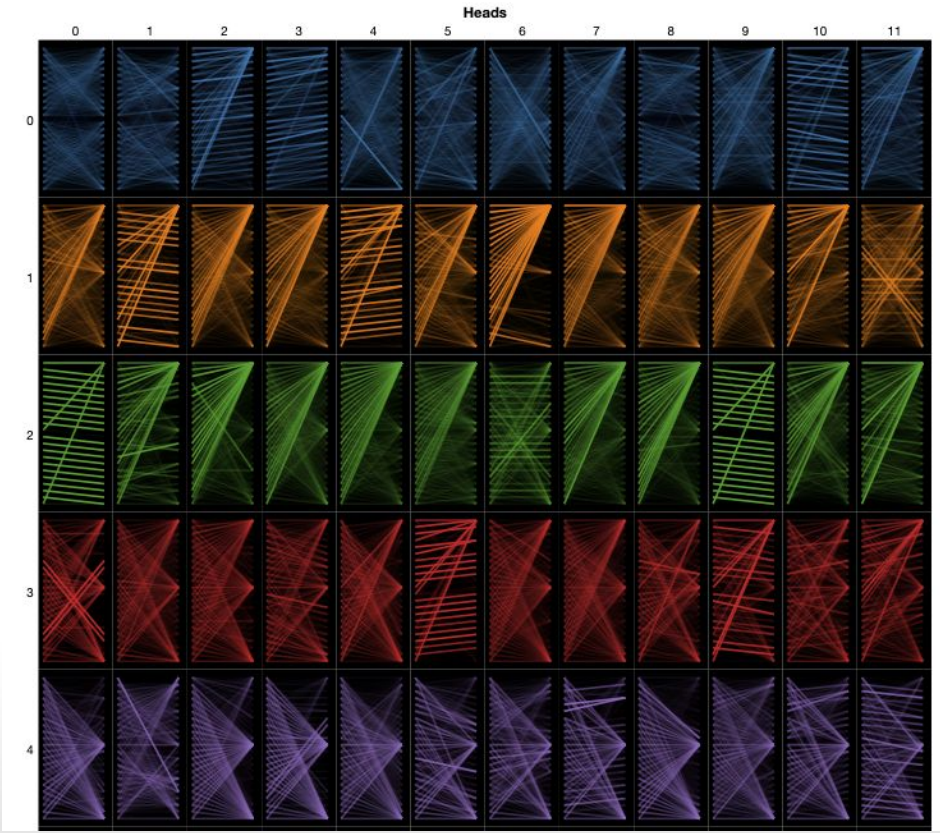


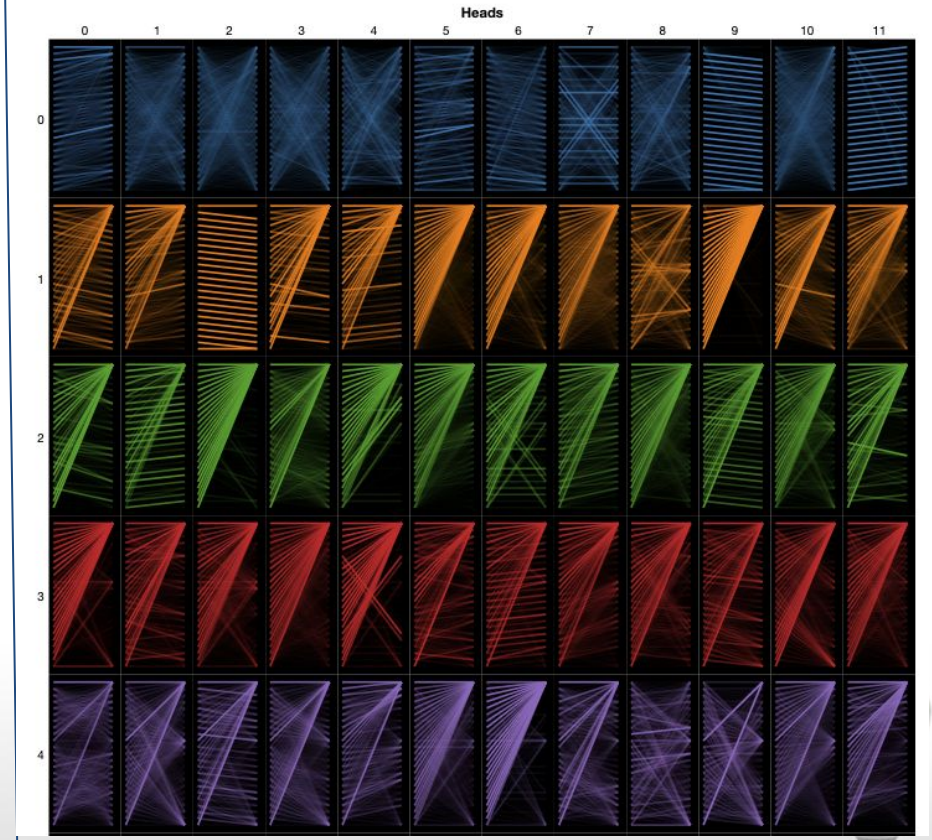
Accuracy : 40.985130111524164 %



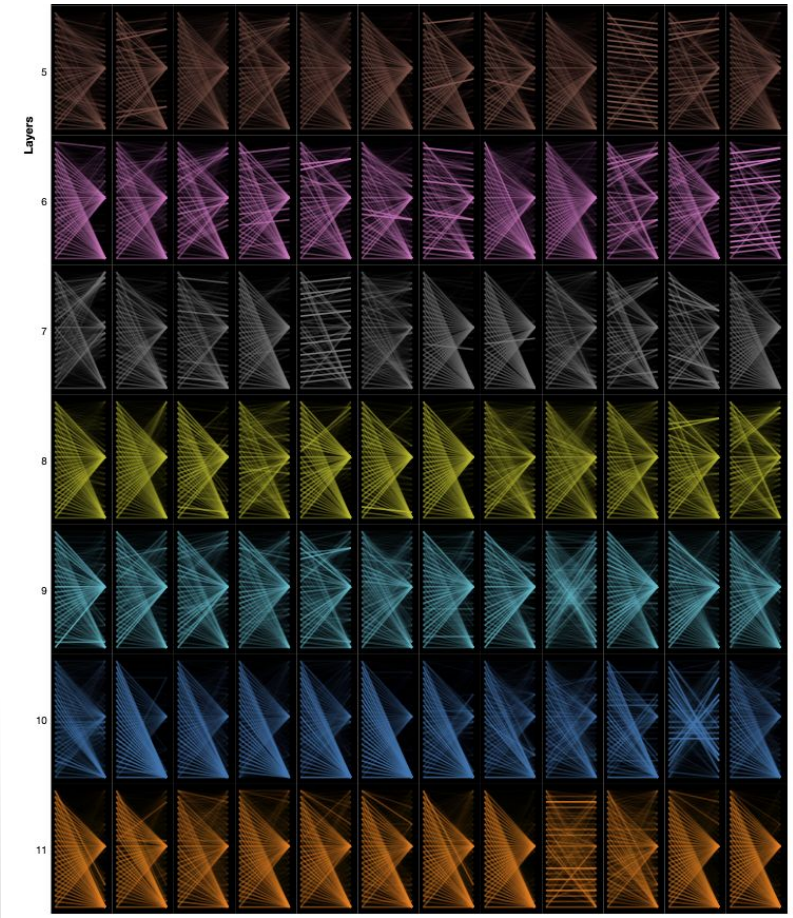
Accuracy : 47.95539033457249 %

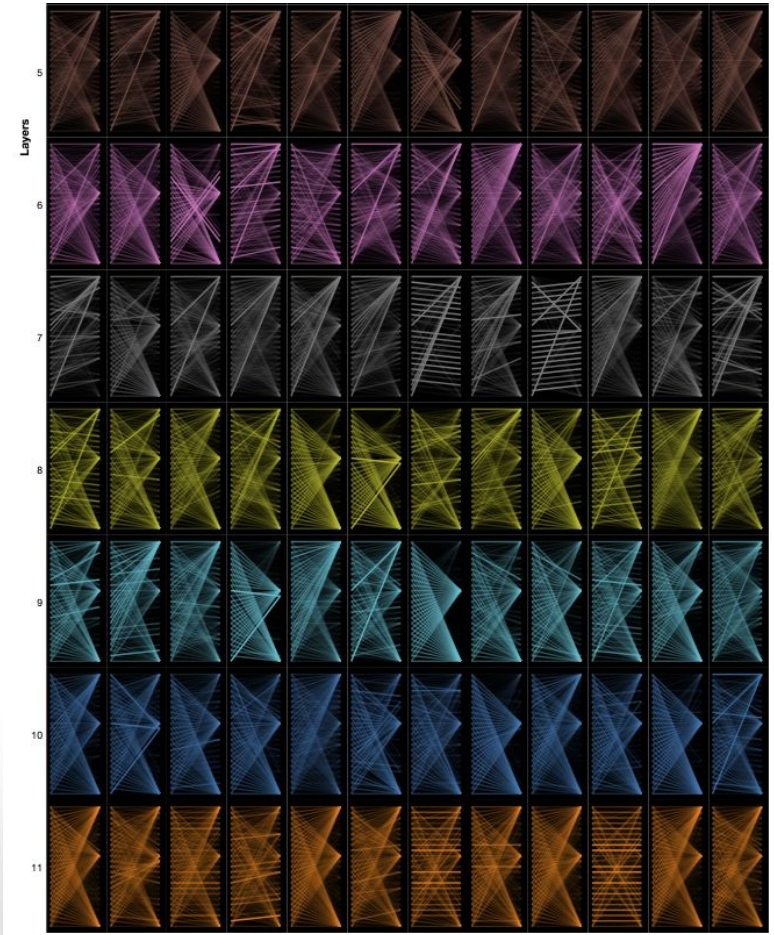
Bert vs Roberta





Bert vs Roberta





SimCSE: Simple Contrastive Learning of Sentence Embeddings

SimCSE: Simple Contrastive Learning of Sentence Embeddings

Overview :

They propose a simple contrastive learning framework that works with both unlabeled and labeled data. Unsupervised SimCSE simply takes an input sentence and predicts itself in a contrastive learning framework, with only standard dropout used as noise. Our supervised SimCSE incorporates annotated pairs from **NLI datasets** into contrastive learning by using **entailment** pairs as **positives** and **contradiction** pairs as hard negatives. The following figure is an illustration of our models.

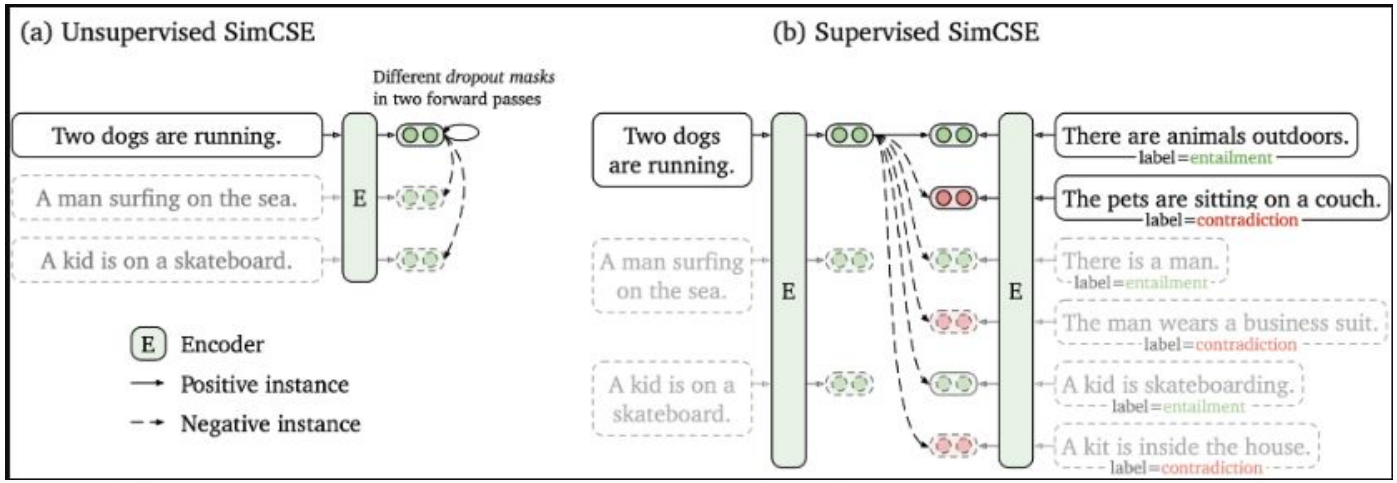
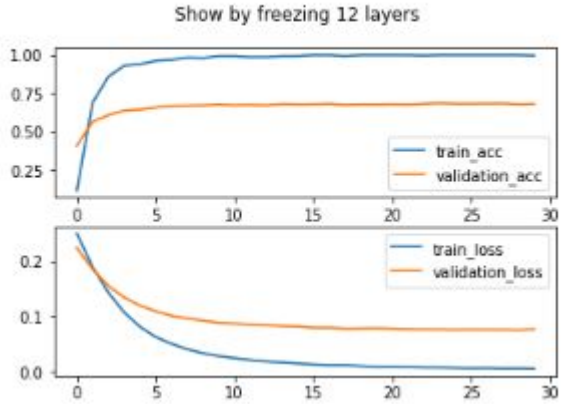


Figure 1: (a) Unsupervised SimCSE predicts the input sentence itself from in-batch negatives, with different hidden dropout masks applied. (b) Supervised SimCSE leverages the NLI datasets and takes the entailment (premise-hypothesis) pairs as positives, and contradiction pairs as well as other in-batch instances as negatives.

SimCSE: Simple Contrastive Learning of Sentence Embeddings (Unsupervised SimCS)

Intent detection classification :

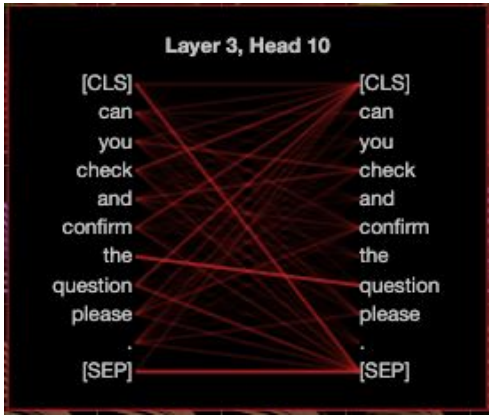
```
[11]: <matplotlib.legend.Legend at 0x7fbbe882d460>
```

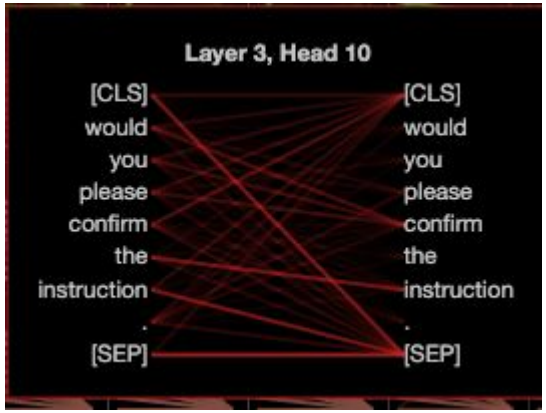


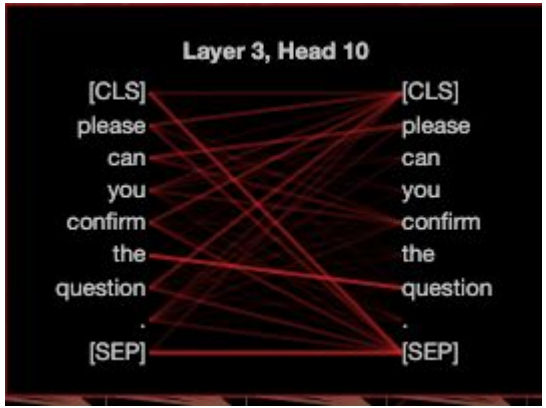

```
[12]: test_acc =test(model,device,label_maps,test_loader,len(test_data),tokenizer)

      correct : 735
      total : 1076

[13]: print(f'Accuracy : {100 * test_acc} %')

      Accuracy : 68.3085501858736 %
```

SimCSE: Simple Contrastive Learning of Sentence Embeddings (Unsupervised SimCS)

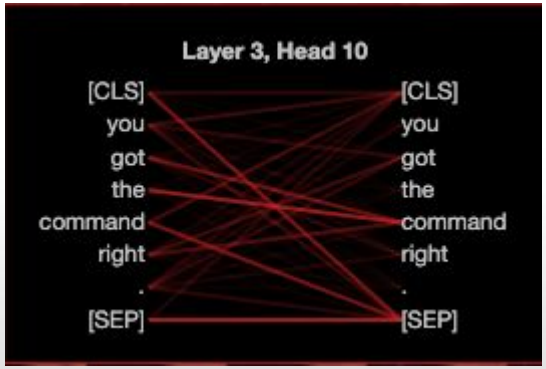
Head attention role (HWU64 : class general_confirm) **det: determiner**

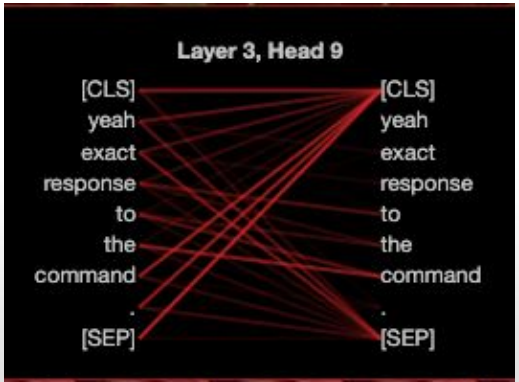




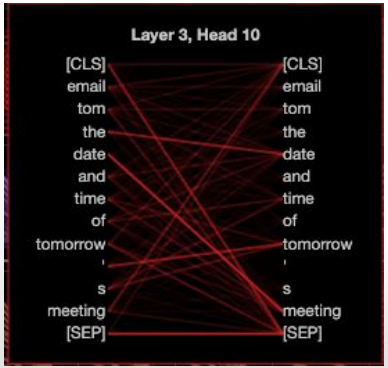


general_affirm



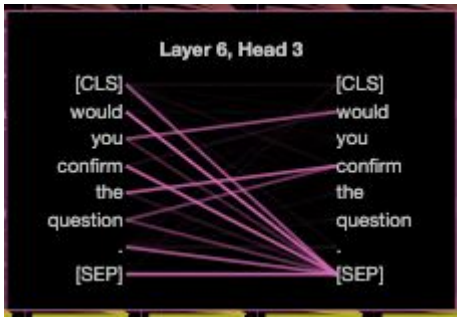


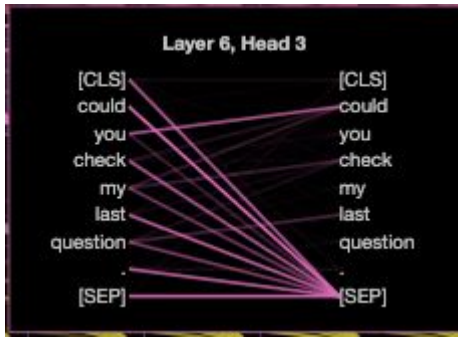
email_sendemail

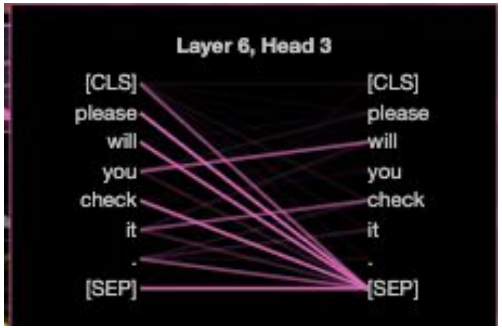


SimCSE: Simple Contrastive Learning of Sentence Embeddings (Unsupervised SimCS)

Head attention role (HWU64 : class general_confirm)

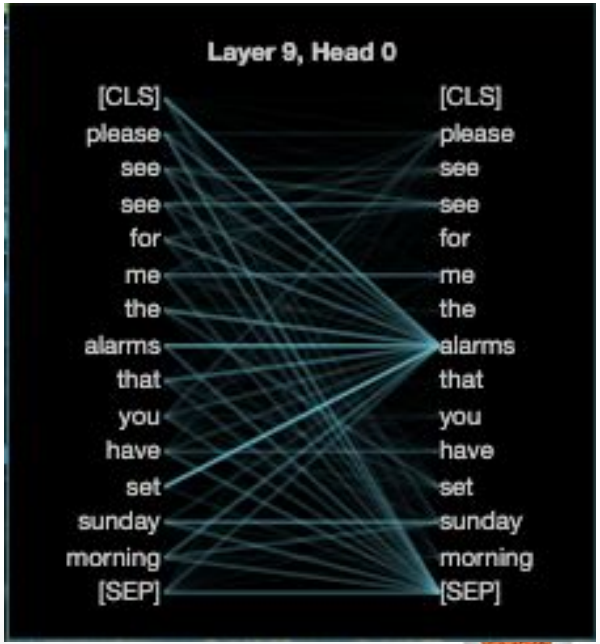






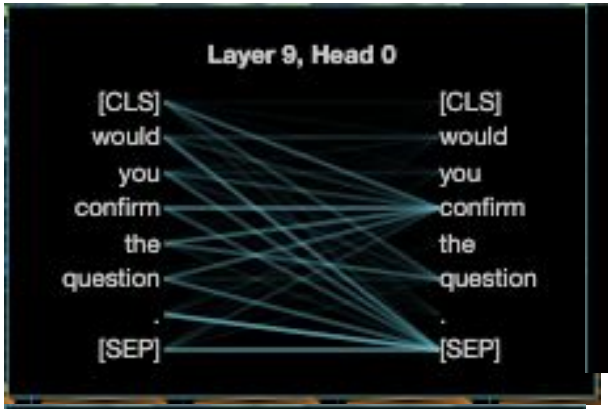
SimCSE: Simple Contrastive Learning of Sentence Embeddings (Unsupervised SimCS)

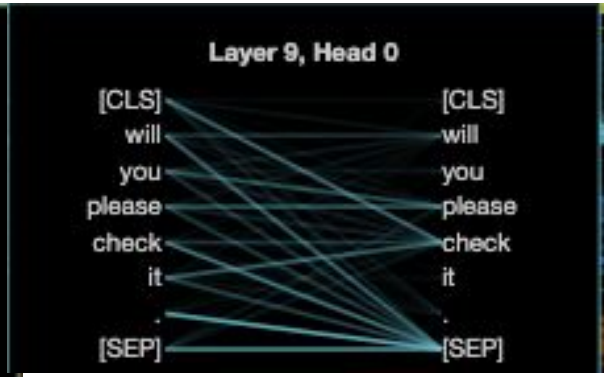
Interesting point. (alarm_query) (**Marked : *A*larm**)

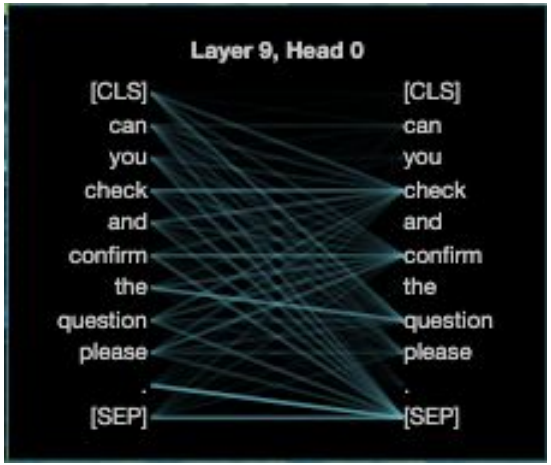


SimCSE: Simple Contrastive Learning of Sentence Embeddings (Unsupervised SimCS)

Interesting point. (general_confirm : **Marked = Confirm, Check**)







BERT
Fine-Tuned vs Not Fine-Tune

Dataset	Description	#Train	#Valid	#Test
BANKING77	one banking domain with 77 intents	8622	1540	3080

```
Train data (2310, 2)
Validate data (1540, 3)
Test data (3080, 3)
```

BERT, Fine-Tuned

```
epochs = 20
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained(
            "bert-base-uncased",              # Use the 12-layer BERT model, with an uncased vocab.
            num_labels = 77,                  # The number of output labels--2 for binary classification.
            output_attentions = True,         # Whether the model returns attentions weights.
            output_hidden_states = False,  # Whether the model returns all hidden-states.
)
optimizer = AdamW
```

Fine-Tuned vs Not Fine-Tune

	sentences	category	labels
0	how do i locate my card?	card_arrival	12
1	i still have not received my new card. i order...	card_arrival	12
2	i ordered a card but it has not arrived. help ...	card_arrival	12
3	is there a way to know when my card will arrive?	card_arrival	12
4	my card has not arrived yet.	card_arrival	12
...	...	...	...
3075	if i'm not in the uk, can i still get a card?	country_support	25
3076	how many countries do you support?	country_support	25
3077	what countries do you do business in?	country_support	25
3078	what are the countries you operate in.	country_support	25
3079	can the card be mailed and used in europe?	country_support	25

3080 rows × 3 columns

```
100%|██████████| 3080/3080 [00:54<00:00, 56.94it/s]
Test Accuracy :  87.175
```

Test

```
predict(model,df_test['sentences'][2])
```

```
The predicted class is : card_arrival
```

Query

```
predict(model,"Can you check if the card has arrived?")
```

```
The predicted class is : card_arrival
```

```
100%|██████████| 3080/3080 [00:53<00:00, 58.05it/s]
Test Accuracy :  1.364
```

Test

```
predict(model,df_test['sentences'][2])
```

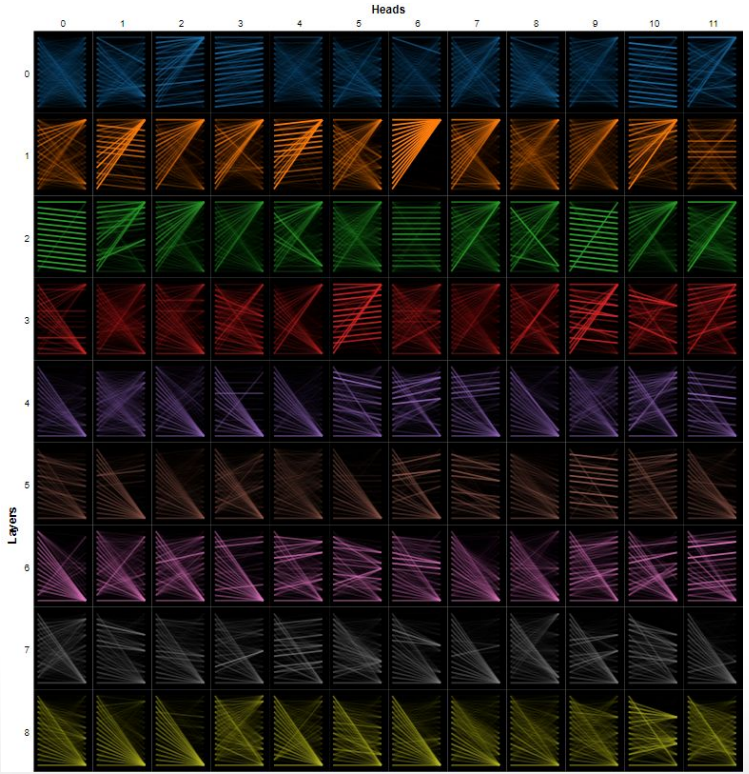
```
The predicted class is : atm_support
```

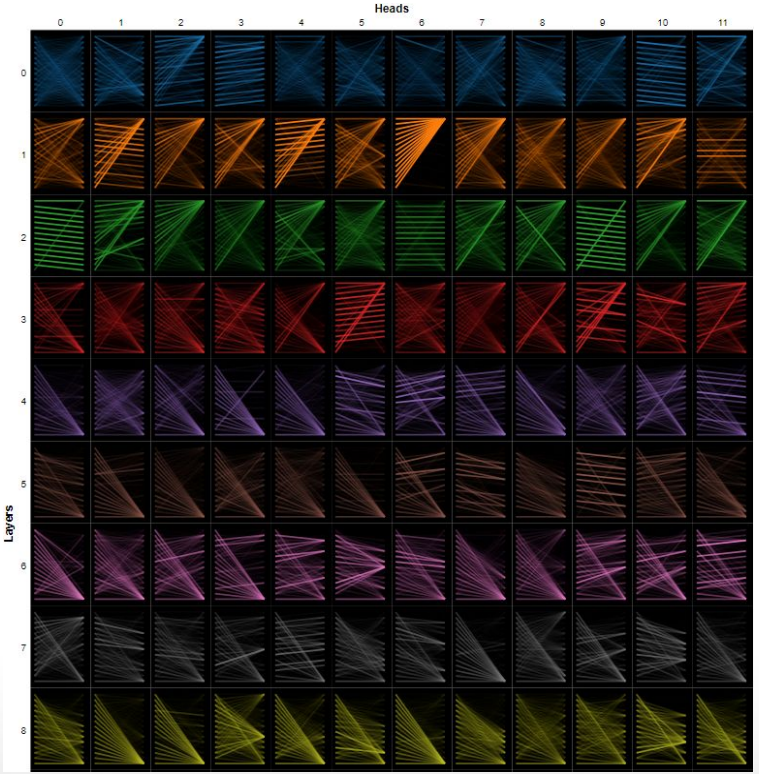
Query

```
predict(model,"Can you check if the card has arrived?")
```

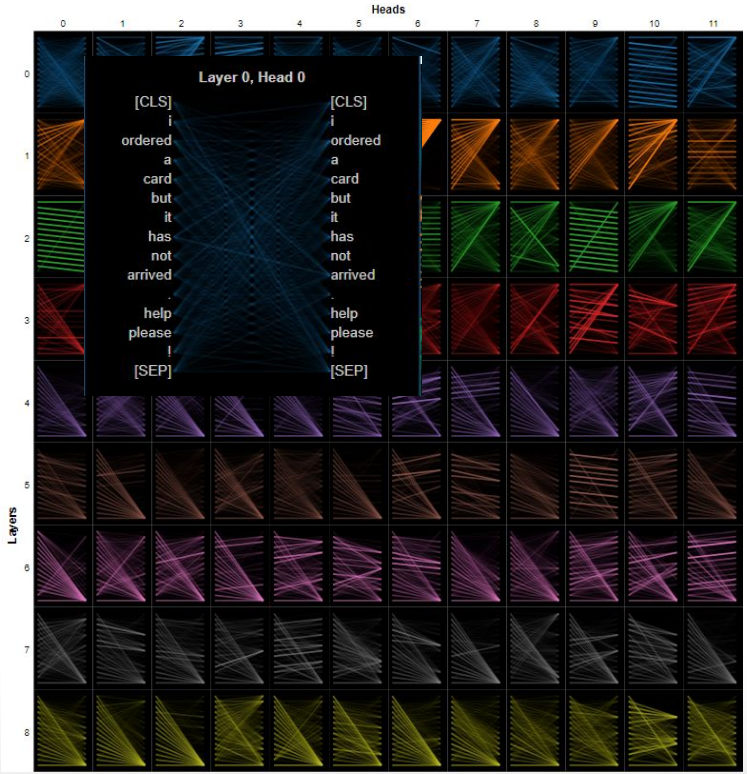
```
The predicted class is : topping_up_by_card
```

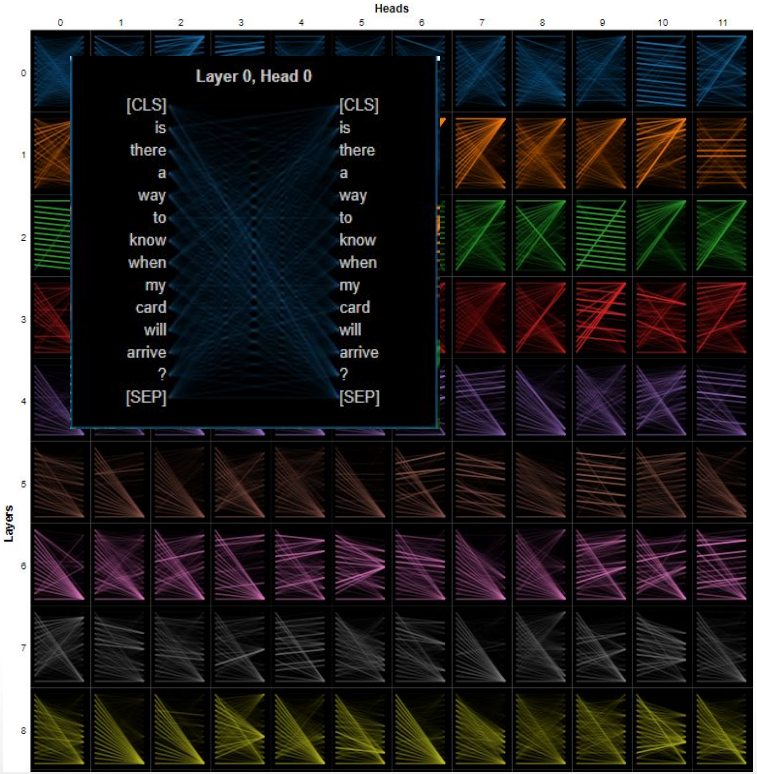
Fine-Tuned


Heads
0 1 2 3 4 5 6 7 8 9 10 11
Layers
0 1 2 3 4 5 6 7 8

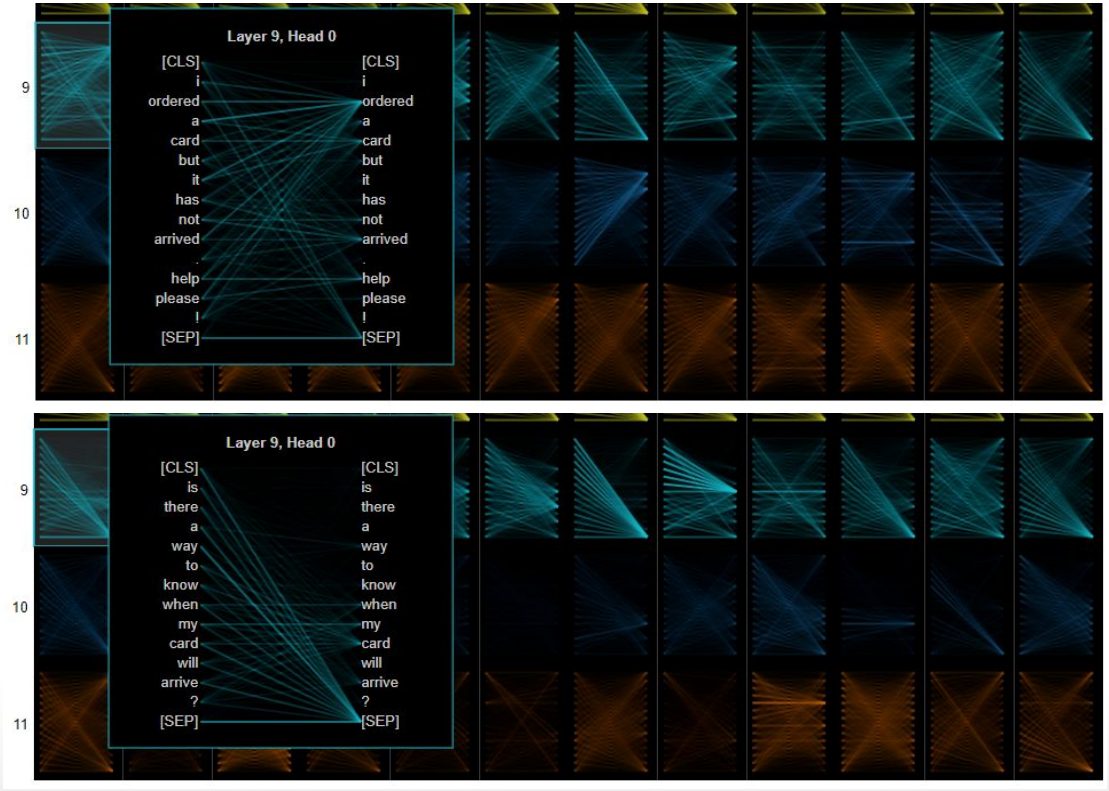

Heads
0 1 2 3 4 5 6 7 8 9 10 11
Layers
0 1 2 3 4 5 6 7 8

Fine-Tuned

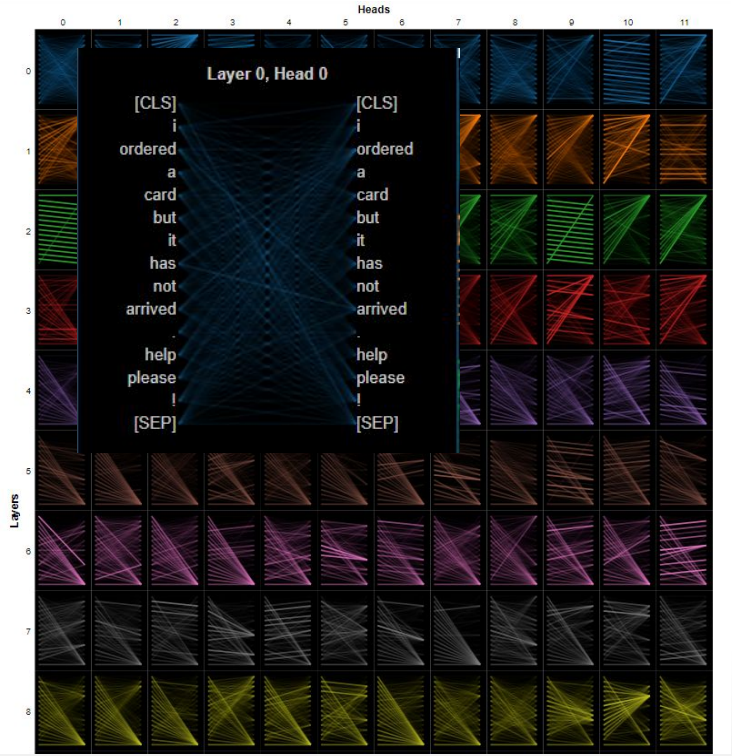

Heads
0 1 2 3 4 5 6 7 8 9 10 11
Layers
0 1 2 3 4 5 6 7 8
Layer 0, Head 0
[CLS] i ordered a card but it has not arrived . help please ! [SEP]
[CLS] i ordered a card but it has not arrived . help please ! [SEP]

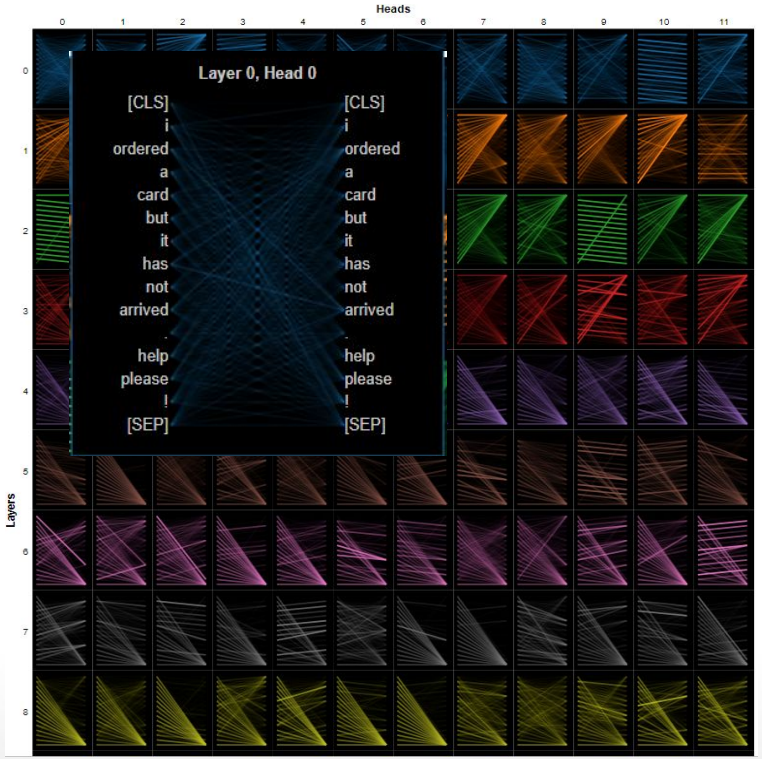

Heads
0 1 2 3 4 5 6 7 8 9 10 11
Layers
0 1 2 3 4 5 6 7 8
Layer 0, Head 0
[CLS] is there a way to know when my card will arrive ? [SEP]
[CLS] is there a way to know when my card will arrive ? [SEP]

Fine-Tuned

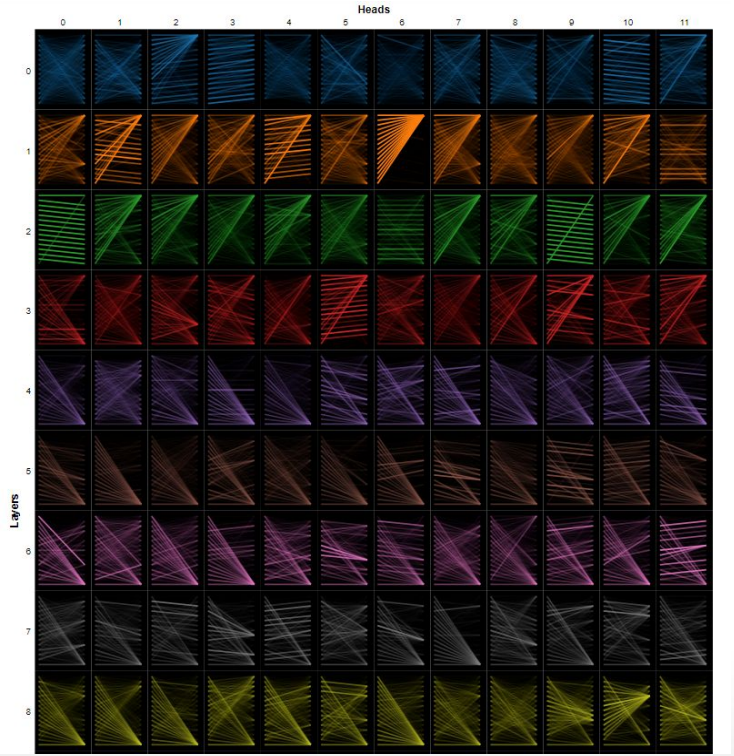

Layer 9, Head 0
[CLS] i ordered a card but it has not arrived . help please ! [SEP]
[CLS] i ordered a card but it has not arrived . help please ! [SEP]
9
10
11
Layer 9, Head 0
[CLS] is there a way to know when my card will arrive ? [SEP]
[CLS] is there a way to know when my card will arrive ? [SEP]
9
10
11

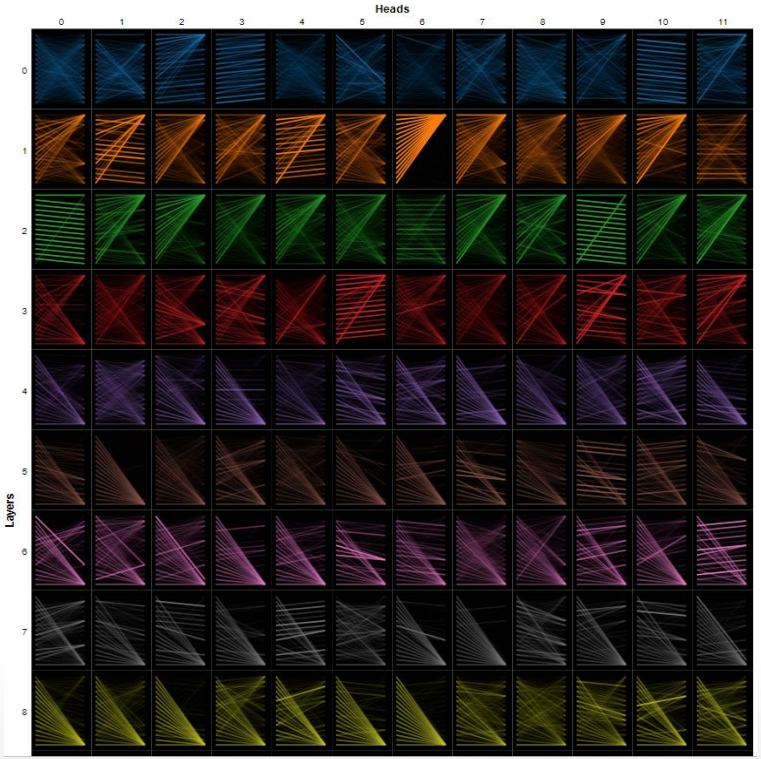
Fine-Tuned vs Not Fine-Tune





Fine-Tuned vs Not Fine-Tune

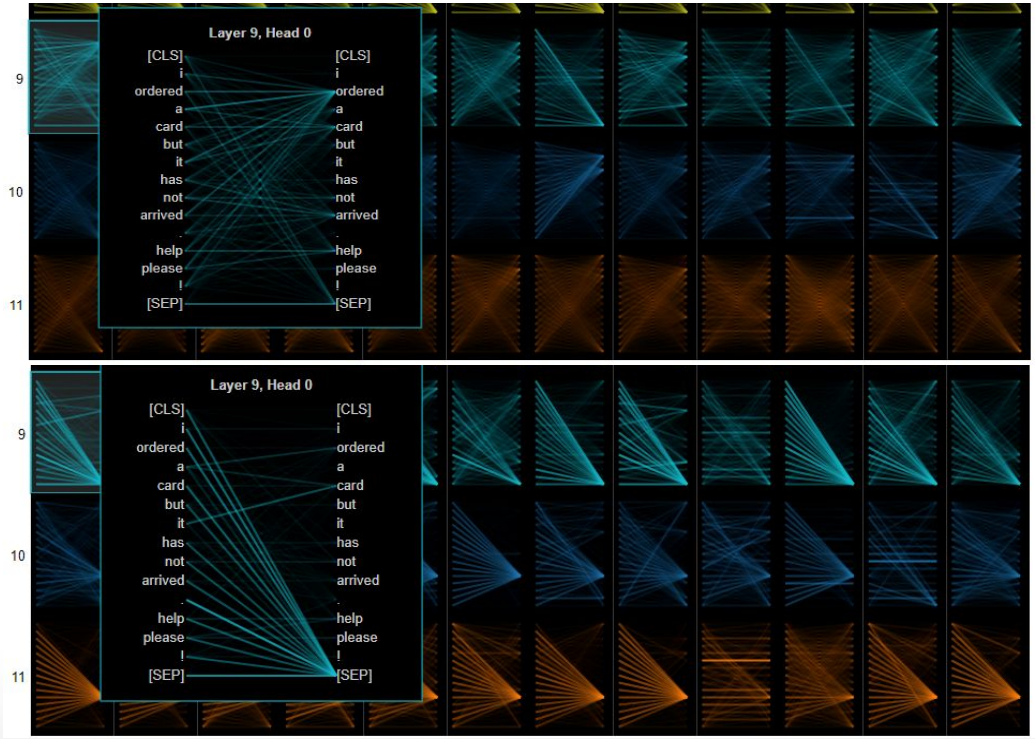




Fine-Tuned

vs

Not Fine-Tune



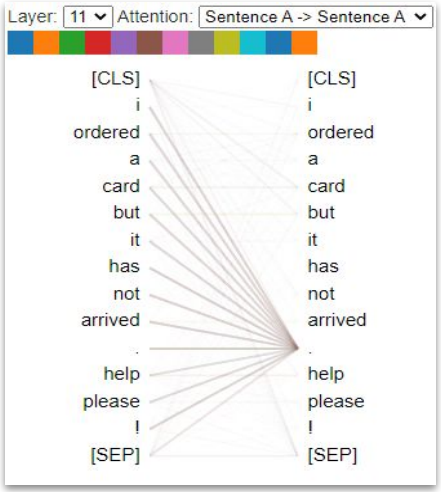
Fine-Tuned vs Not Fine-Tune

Result

- For bert pretrained model, we can see layer 0-8, the pattern are similarly in both fine-tuned and not fine-tuned model even the sentences are the same and not the same.
- For Bert model which is not fine-tuned performs very poorly but it does not mean Bert model perform that bad because we do not have to train since the beginning.

Result

Fine-Tuned **vs** **Not Fine-Tune**



- For bert model with fine-tuned, it creates more optimized weights between tokens, in other words, it creates more weighted connections between the tokens which probably helps in forming newer patterns to make sense of the interaction between words, hence the accuracy is increased.

BERT
BANKING77 vs HWU64

```
epochs = 20
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained(
        "bert-base-uncased",          # Use the 12-layer BERT model, with an uncased vocab.
        num_labels = 77, /64          # The number of output labels--2 for binary classification.
        output_attentions = True,     # Whether the model returns attentions weights.
        output_hidden_states = False, # Whether the model returns all hidden-states.
)
optimizer = AdamW
```

Test Accuracy : 0.858

Test Accuracy : 0.875