BERT and ClinicalBERT

2018 Oct	2019 Jan	2019 Mar	2019 April
BERT	BioBERT	SciBERT	ClinicalBERT

2018 Oct 2019 Jan 2019 Mar 2019 April

BERT BioBERT SciBERT ClinicalBERT

BERT = Bidirectional Encoder Representation from Transformers

- BERT is published by Google in 2018. It obtained the best accuracy in 11 different NLP tasks.

Why was BERT needed?

The lack of training data was a big challenge in NLP, as deep learning models require large amounts of annotated data to perform well.

To address this, researchers have developed pre-training techniques such as BERT to utilize unannotated text data.

These kind of pre-trained models (e.g., BERT) can be fine-tuned on smaller task-specific datasets (e.g., MIMIC notes) to get fine-tuned models (e.g., BioBERT, SciBERT, ClinicalBERT) to achieve better accuracy in specific domain.

Read more: https://towardsml.wordpress.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/

What is the core idea behind BERT?

- BERT takes advantages of multiple models

(1) BERT predicts word from given context - Word2Vec CBOW

(2) 2-layer bidirectional model –ELMO (a word embedding method for representing a sequence of words as a corresponding sequence of vectors)

(3) Transformer instead of RNN –GPT (Generative Pre-training)

Use Transformer proposed in *Attention is All you need* in 2017 to replace RNN

Bert : Pretraining and fine-tuning

The corpus BERT uses for pre-training is BooksCorpus and English Wikipedia.

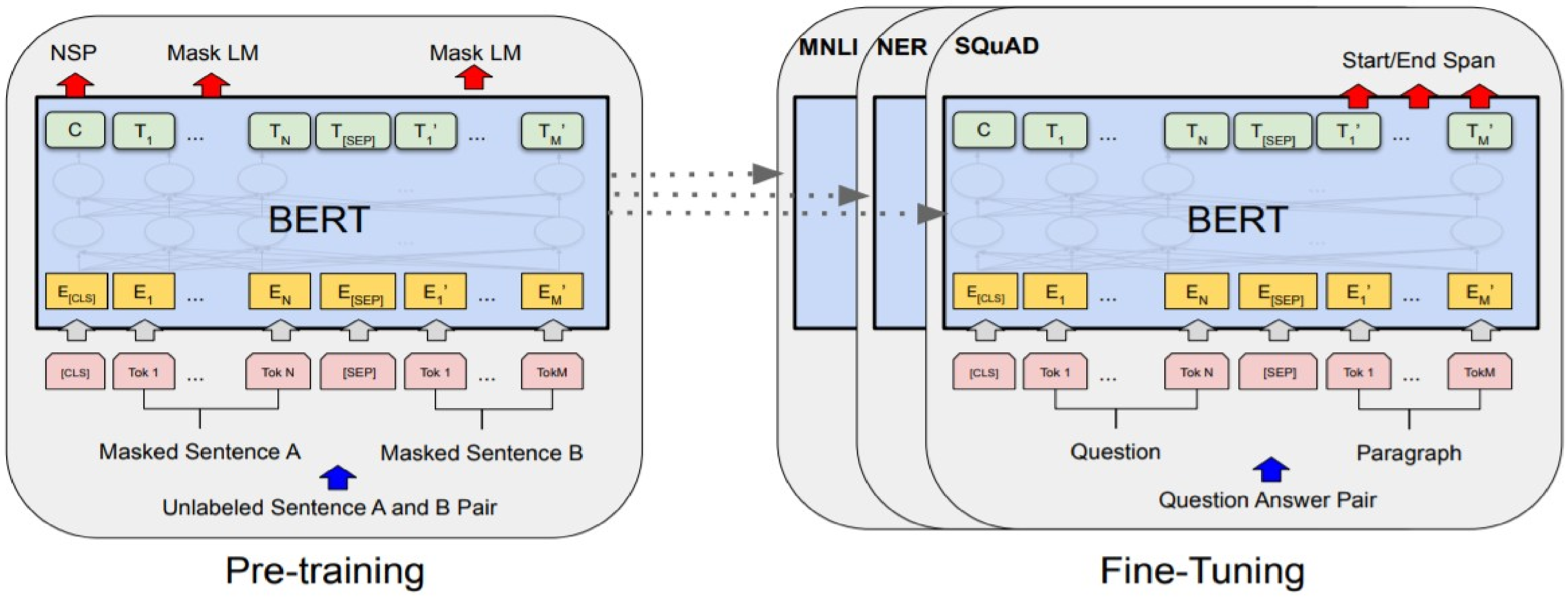
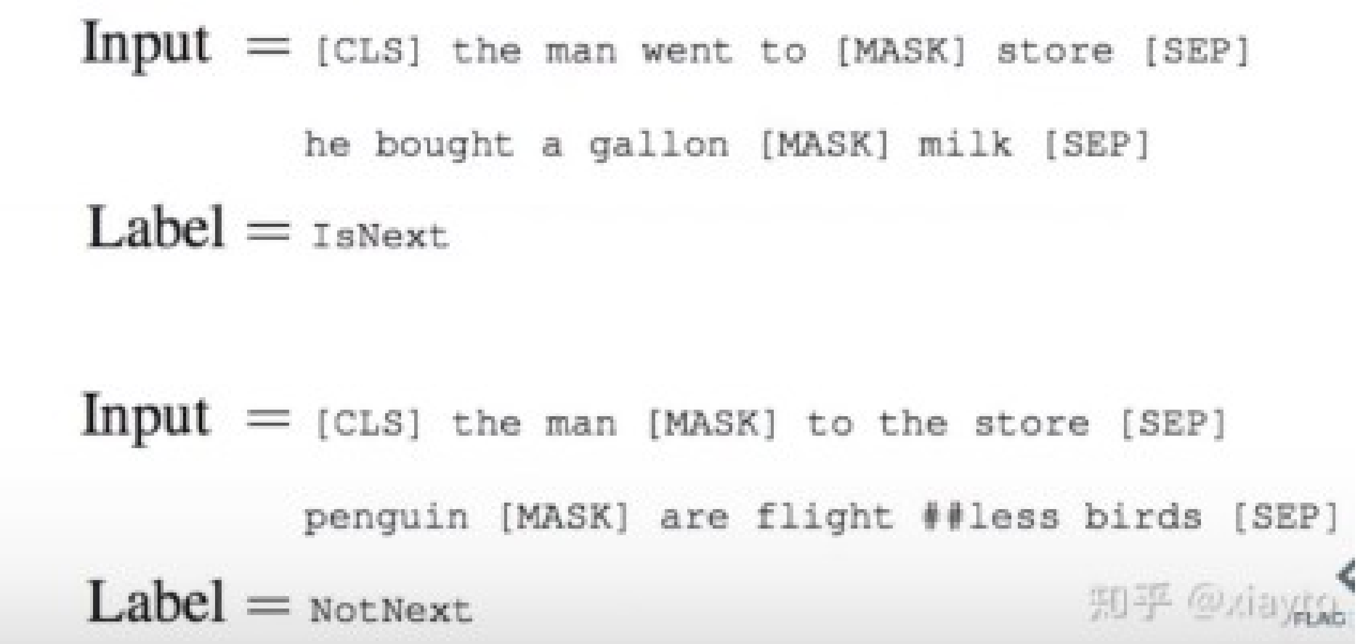


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

https://arxiv.org/pdf/1810.04805.pdf

https://huggingface.co/blog/bert-101

Bert-Pretraining



1. Uses Masked Language Model to train model.

It masks 15% words of doc:

80% use "[mask]"

10% use original word

10% use a random word

e.g., To be or [mask] to be, that is the question

2. Continuous sentence or not

To be or not to be, that is the question VS To be or not to be, or to take arms against a sea of troubles

Fine tuning tasks

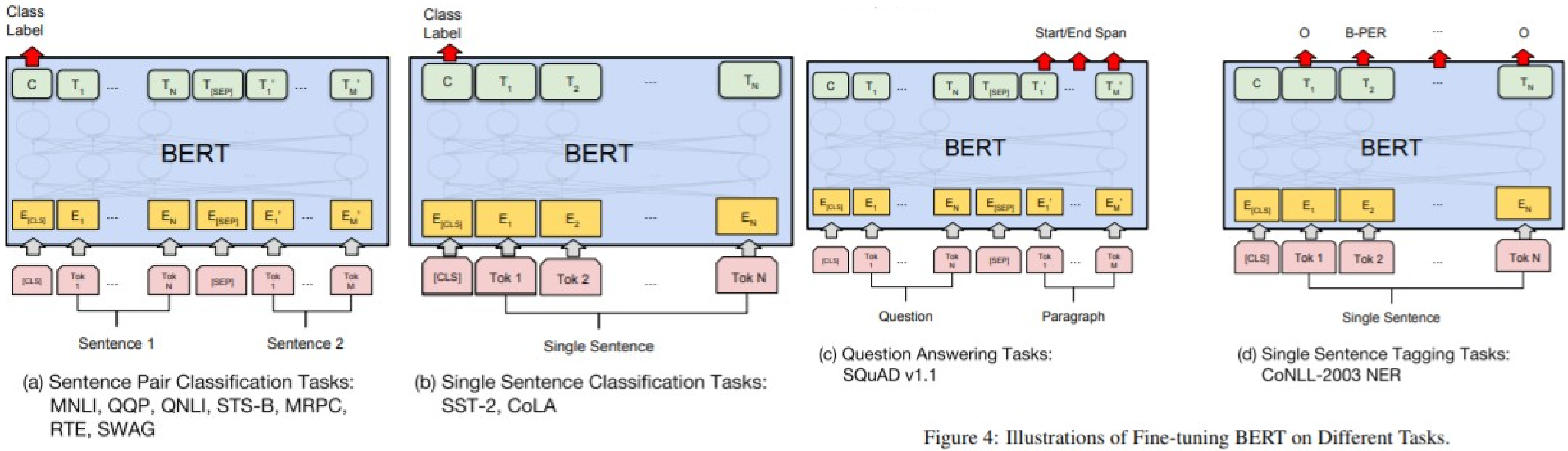


Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

https://arxiv.org/pdf/1810.04805.pdf

Why ClinicalBERT?

Directly applying BERT to biomedical NLP tasks is not promising because of a word distribution shift from general domain corpus to biomedical domain corpus.

Thus, other models e.g, BioBERT [2] and BlueBERT [3], SciBERT, ClinicalBERT pretrained on biomedical domain corpus are proposed.

ClinicalBERT finetuning

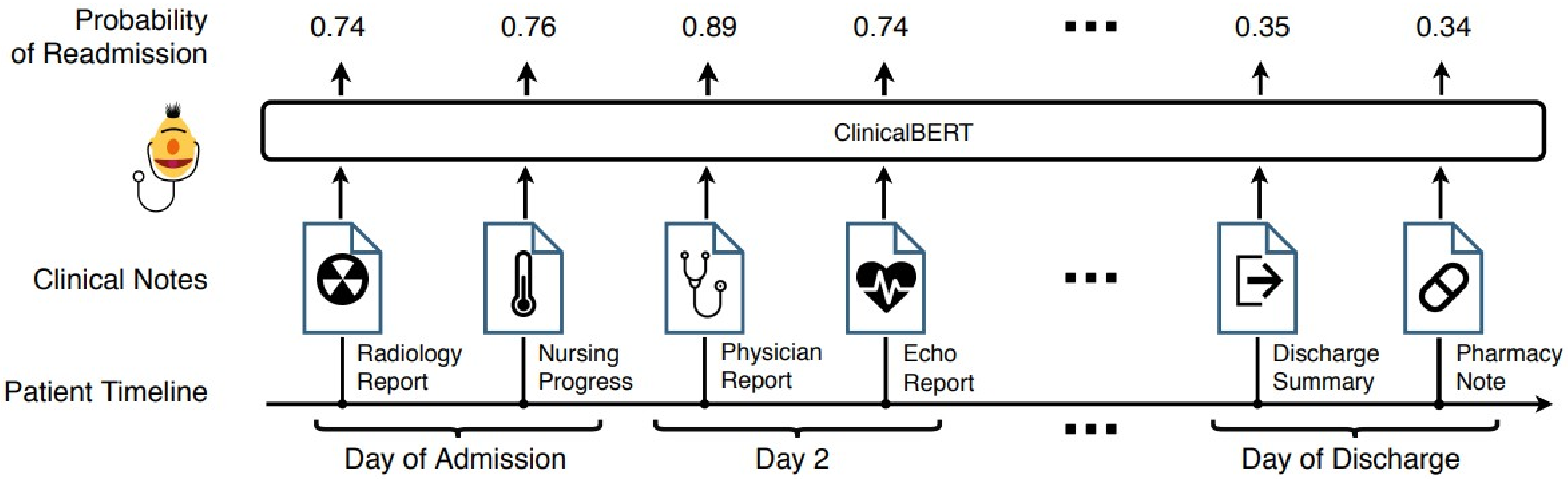


Figure 1: ClinicalBERT learns deep representations of clinical notes that are useful for tasks such as readmission prediction. In this example, care providers add notes to an electronic health record during a patient's admission, and the model dynamically updates the patient's risk of being readmitted within a 30-day window.

https://arxiv.org/pdf/1904.05342.pdf

Finetuning task:
Readmission prediction

Table 4: ClinicalBERT outperforms competitive baselines on readmission prediction using clinical notes from early on within patient admissions. In MIMIC-III data, admission and discharge times are available, but clinical notes do not have timestamps. The cutoff time indicates the range of admission durations that are fed to the model from early in a patient's admission. For example, in the 24–48h column, the model may only take as input a patient's notes up to 36h because of that patient's specific admission time. Metrics are reported as the mean and standard deviation of 5 independent runs.

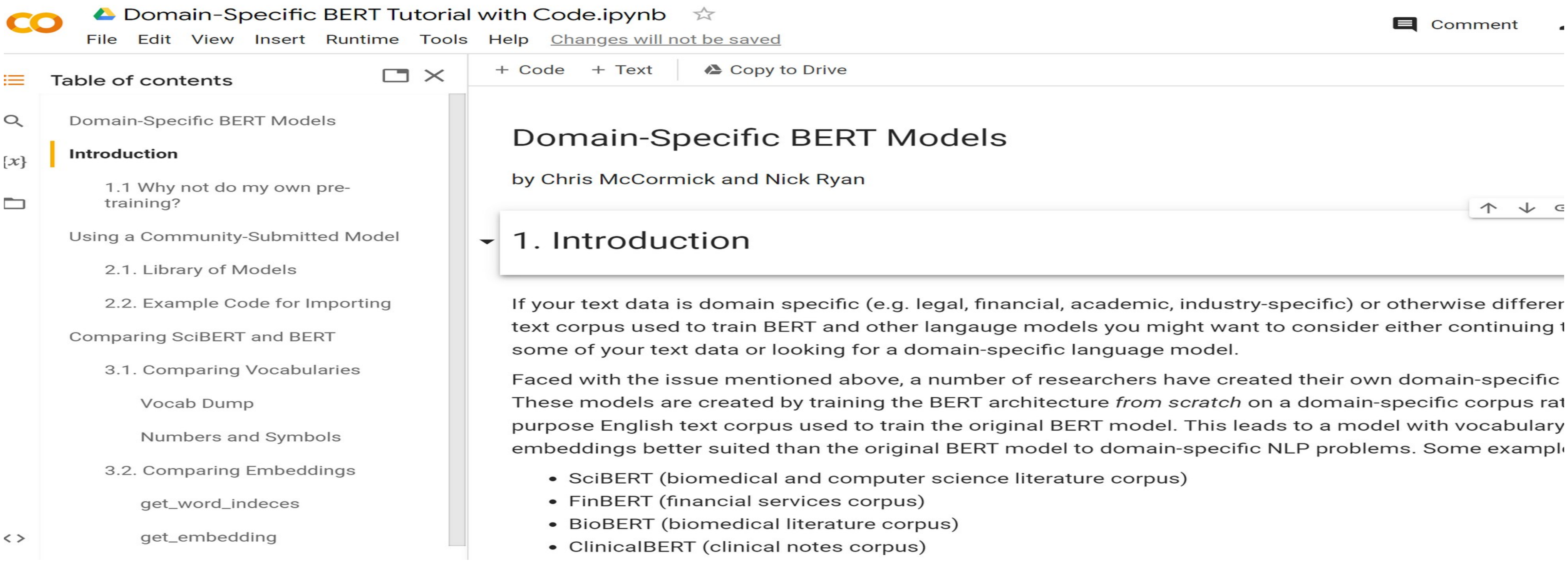
Model	Cutoff time	AUROC	AUPRC	RP80
ClinicalBERT	24–48h	0.674 ± 0.038	0.674 ± 0.039	0.154 ± 0.099
	48–72h	0.672 ± 0.039	0.677 ± 0.036	0.170 ± 0.114
Bag-of-words	24–48h	0.648 ± 0.029	0.650 ± 0.027	0.144 ± 0.094
	48–72h	0.654 ± 0.035	0.657 ± 0.026	0.122 ± 0.106
BI-LSTM	24–48h	0.649 ± 0.044	0.660 ± 0.036	0.143 ± 0.080
	48–72h	0.656 ± 0.035	0.668 ± 0.028	0.150 ± 0.081
BERT	24–48h	0.659 ± 0.034	0.656 ± 0.021	0.141 ± 0.080
	48–72h	0.661 ± 0.028	0.668 ± 0.021	0.167 ± 0.088

RP80: Recall at Precision 80%, which is used to control false positive.

https://arxiv.org/pdf/1904.05342.pdf

ClinicalBert Tutorial

- Modified from Chris McCormick and Nick Ryan's SciBERT Tutorial

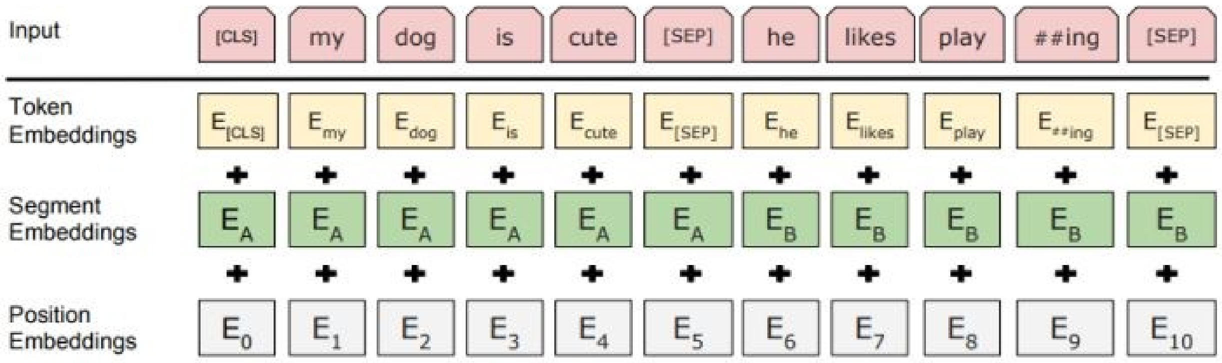


https://colab.research.google.com/drive/19loLGUDixGKv4ulZI1m3hALg2ozNvEGe#scrollTo=uXKyKe3NZONV

Bert/ClinicalBERT Architecture

BERT_base has 13 hidden layers:

the first layer = sub-word embedding layer = "input embeddings"= Token embeddings+ Segment Emb+ Position Emb)

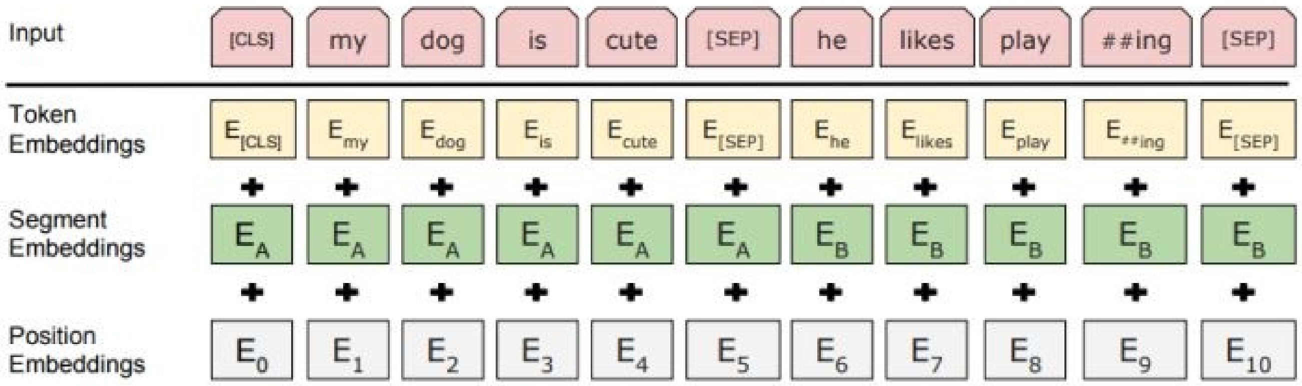


The last layer = Contextual representations = final output of BERT = we usually use this as embeddings for other tasks

https://arxiv.org/abs/1810.04805

BERT (clinical bert)’s first layer: Input embeddings

1. **Token embeddings**: A [CLS] token is added to the input word tokens at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. **Segment embeddings**: A marker indicating Sentence A or Sentence B is added to each token. This allows the encoder to distinguish between sentences.
3. **Positional embeddings**: A positional embedding is added to each token to indicate its position in the sentence.



Tokenizer

The token embeddings are obtained from WordPiece. An example of the tokenized word is “coronaviruses” => “Co##rona##virus##es”.

For the named entity recognition task, [CLS] is added at the beginning of each sentence and [SEP] is added at the end of each sentence.

```
[CLS]
E
##pid
##em
##ic
size
of
novel
co
##rona
##virus
.
.
[SEP]
```

https://arxiv.org/pdf/1609.08144v2.pdf

```python
for i in range(len(notes)):
  text = notes[i]
  bert_tokens = bert_tokenizer.tokenize(text)
  clinical_tokens = clinical_tokenizer.tokenize(text)
  bluebert_tokens = blue_tokenizer.tokenize(text)
  biobert_tokens = biobert_tokenizer.tokenize(text)
  # Pad out the clinical bert, bluebert list to be the same length.
  while len(clinical_tokens) < len(bert_tokens):
      clinical_tokens.append("")

  while len(bluebert_tokens) < len(bert_tokens):
      bluebert_tokens.append("")
  while len(biobert_tokens) < len(bert_tokens):
      biobert_tokens.append("")
  # Label the columns.
  print('{:<12} {:<12} {:<12} {:<12}'.format("BERT", "ClinicalBERT","bluebert","biobert"))
  print('{:<12} {:<12} {:<12} {:<12}'.format("----", "-------", "-------", "-------"))

  # Display the tokens.
  for tup in zip(bert_tokens, clinical_tokens, bluebert_tokens, biobert_tokens):
      print('{:<12} {:<12} {:<12} {:<12}'.format(tup[0], tup[1], tup[2], tup[3]))
```

```
BERT         ClinicalBERT bluebert     biobert
----         -------      -------      -------
50           50           50           50
year         year         year         year
old          old          old          old
female       female       female       female
presents     presents     presents     presents
after        after        after        after
having       having       having       having
fallen       fallen       fallen       fallen
in           in           in           in
the          the          the          the
bath         bath         bath         bath
##tub        ##tub        ##tub        ##tub
4            4            4            4
days         days         days         days
ago          ago          ago          ago
and          and          and          and
hitting      hitting      hitting      hitting
the          the          the          the
back         back         back         back
of           of           of           of
her          her          her          her
head         head         head         head
.            .            .            .
since        since        since        since
then         then         then         then
she          she          she          she
has          has          has          has
had          had          had          had
a            a            a            a
massive      massive      massive      massive
```

Compare all BERT and its variant Tokenizer

Get word embedding and sentence embeddi

```python
import numpy as np

def get_word_indeces(tokenizer, text, word):
    '''
    Determines the index or indeces of the tokens corresponding to `word`
    within `text`. `word` can consist of multiple words, e.g., "cell biology".

    Determining the indeces is tricky because words can be broken into multiple
    tokens. I've solved this with a rather roundabout approach--I replace `word`
    with the correct number of `[MASK]` tokens, and then find these in the
    tokenized result.
    '''
    # Tokenize the 'word'--it may be broken into multiple tokens or subwords.
    word_tokens = tokenizer.tokenize(word)

    # Create a sequence of `[MASK]` tokens to put in place of `word`.
    masks_str = ' '.join(['[MASK]']*len(word_tokens))

    # Replace the word with mask tokens.
    text_masked = text.replace(word, masks_str)

    # `encode` performs multiple functions:
    #   1. Tokenizes the text
    #   2. Maps the tokens to their IDs
    #   3. Adds the special [CLS] and [SEP] tokens.
    input_ids = tokenizer.encode(text_masked)

    # Use numpy's `where` function to find all indeces of the [MASK] token.
    mask_token_indeces = np.where(np.array(input_ids) == tokenizer.mask_token_id)[0]

    return mask_token_indeces
```

Get word embedding and sentence embeddi

```
def get_embedding(b_model, b_tokenizer, text, word=''):
    '''
    Uses the provided model and tokenizer to produce an embedding for the
    provided `text`, and a "contextualized" embedding for `word`, if provided.
    '''

    # If a word is provided, figure out which tokens correspond to it.
    if not word == '':
        word_indeces = get_word_indeces(b_tokenizer, text, word)

    # Encode the text, adding the (required!) special tokens, and converting to
    # PyTorch tensors.
    encoded_dict = b_tokenizer.encode_plus(
                        text,                      # Sentence to encode.
                        add_special_tokens = True, # Add '[CLS]' and '[SEP]'
                        return_tensors = 'pt',     # Return pytorch tensors.
                )

    input_ids = encoded_dict['input_ids']

    b_model.eval()

    # Run the text through the model and get the hidden states.
    bert_outputs = b_model(input_ids)
```

Get word embedding and sentence embeddi

```
with torch.no_grad():

    outputs = b_model(input_ids)

    # Evaluating the model will return a different number of objects based on
    # how it's  configured in the `from_pretrained` call earlier. In this case,
    # becase we set `output_hidden_states = True`, the third item will be the
    # hidden states from all layers. See the documentation for more details:
    # https://huggingface.co/transformers/model_doc/bert.html#bertmodel
    hidden_states = outputs[2]

# `hidden_states` has shape [13 x 1 x <sentence length> x 768]

# Select the embeddings from the second to last layer.
# `token_vecs` is a tensor with shape [<sent length> x 768]
token_vecs = hidden_states[-2][0]

# Calculate the average of all token vectors.
sentence_embedding = torch.mean(token_vecs, dim=0)

# Convert to numpy array.
sentence_embedding = sentence_embedding.detach().numpy()

# If `word` was provided, compute an embedding for those tokens.
if not word == '':
    # Take the average of the embeddings for the tokens in `word`.
    word_embedding = torch.mean(token_vecs[word_indeces], dim=0)

    # Convert to numpy array.
    word_embedding = word_embedding.detach().numpy()

    return (sentence_embedding,word_embedding)
else:
    return sentence_embedding
```

Embeddings of Clinical Bert

```
text = notes[0]
word = 'headache'

text = clean_text(text)
clinical_model.eval()
# Get the embedding for the sentence, as well as an embedding for 'pneumothorax'..
(sen_emb, word_emb) = get_embedding(clinical_model, clinical_tokenizer, text, word)
print('Embedding sizes:')
print(sen_emb.shape)
print(word_emb.shape)
print(sen_emb)
print(word_emb)
```

```
year old female present fallen bathtub day ago hitting back since massive resolve state high threshold pain realize bad day work got home night noticed patient noticed vision trouble
[101, 1214, 1385, 2130, 1675, 4984, 10919, 25098, 1285, 2403, 6886, 1171, 1290, 4672, 10820, 1352, 1344, 11810, 2489, 4663, 2213, 1285, 1250, 1400, 1313, 1480, 3535, 5351, 3535, 415
Embedding sizes:
(768,)
(768,)
[-2.59695621e-03 -3.62596899e-01 -4.72184896e-01  1.52986467e-01
  5.89456618e-01  3.75910960e-02  3.39461684e-01 -8.42615496e-03
 -2.98118830e-01  1.59292206e-01 -1.04289986e-01  4.81323302e-02
  3.85086864e-01 -3.49685520e-01  1.93859991e-02  4.93307635e-02
  3.41039419e-01 -1.23526350e-01 -7.07958162e-01 -8.50893706e-02
  1.00666471e-01 -4.44369376e-01  1.45851985e-01 -2.65941676e-02
 -2.36840129e-01  2.46712938e-01  2.38447070e-01  9.33627665e-01
  4.21336532e-01  1.72178000e-01  1.13777060e-04 -2.21060179e-02
 -5.70820689e-01  3.36988494e-02 -8.95482395e-03 -1.63546167e-02
 -1.68886393e-01  3.48216891e-01 -2.35619158e-01 -1.08994760e-01
  4.46761668e-01  1.60398632e-01  3.99597436e-01  2.88559049e-01
  3.56576890e-01  5.50093770e-01  6.87661394e-02 -1.79895043e-01
 -5.10067701e-01  3.86664152e-01  1.29374996e-01  2.28122115e-01
  5.11318803e-01 -5.56026220e-01 -1.85265586e-01 -7.20484078e-01
 -2.70037383e-01 -9.79691893e-02 -2.76802808e-01  5.15383422e-01
```

Embeddings of Bio Bert

```
text = notes[6]
word = 'headache'

text = clean_text(text)
biobert_model.eval()
# Get the embedding for the sentence, as well as an embedding for 'pneumothorax'..
(sen_emb, word_emb) = get_embedding(biobert_model, biobert_tokenizer, text, word)
print('Embedding sizes:')
print(sen_emb.shape)
print(word_emb.shape)
print(sen_emb)
print(word_emb)
```

```
46 yo female significant medical problems initially presented pcp initial 2 weeks reports two weeks incresing shortness sob worse lying flat bending better lies stomach two pillows s
[101, 3993, 26063, 2130, 2418, 2657, 2645, 2786, 2756, 185, 1665, 1643, 3288, 123, 2277, 3756, 1160, 2277, 1107, 13782, 4253, 1603, 1757, 20295, 4146, 4009, 3596, 16571, 1618, 2887,
Embedding sizes:
(768,)
(768,)
[-1.58392727e-01 -2.43201435e-01  2.10461065e-01  5.07230461e-01
  2.58089006e-01 -5.72208762e-01 -2.94255137e-01 -3.58504802e-02
 -5.33184886e-01  1.02361488e+00  6.82240948e-02  2.85808861e-01
  4.02891815e-01 -3.37186716e-02 -1.58934280e-01  1.49866760e-01
 -2.82176137e-01 -8.14104497e-01 -4.16680694e-01  5.54907858e-01
  3.67150046e-02 -6.65370524e-02  1.83616266e-01 -5.18894084e-02
 -3.50257248e-01 -3.79871339e-01 -3.48687291e-01  7.35900760e-01
 -3.43888998e-02  4.32009727e-01  1.02412619e-01 -5.36484301e-01
  3.08051050e-01 -2.89696157e-01  3.41139697e-02  3.23256761e-01
  1.08741317e-02  2.98423469e-01  5.60583221e-03  4.55488771e-01
  5.74148774e-01 -1.58784732e-01 -1.17233105e-01 -5.60602434e-02
 -1.10690948e-03 -1.28745586e-01  2.33522326e-01 -1.55903995e-01
 -8.91071796e-01 -2.49175448e-02  8.75999406e-02  4.83856469e-01
 -5.70980869e-02 -6.04957283e-01 -2.85521686e-01 -8.49213362e-01
 -1.50462627e-01 -2.09039807e-01 -5.24491668e-01  2.10274696e-01
```

Embeddings of Blue Bert

```
text = notes[8]
word = 'headache'

text = clean_text(text)
blue_bert_model.eval()
# Get the embedding for the sentence, as well as an embedding for 'pneumothorax'..
(sen_emb, word_emb) = get_embedding(blue_bert_model, blue_tokenizer, text, word)
print('Embedding sizes:')
print(sen_emb.shape)
print(word_emb.shape)
print(sen_emb)
print(word_emb)
```

```
82 year old female aaa repair presenting severe [MASK] substernal chest states [MASK] similar past hypertensive hospitalized similar presentation patient reports 3 days shortness sor
[101, 6445, 2095, 2214, 2931, 13360, 7192, 10886, 5729, 103, 4942, 6238, 12032, 3108, 2163, 103, 2714, 2627, 23760, 25808, 3512, 24735, 2714, 8312, 5776, 4311, 1017, 2420, 2460, 2791
Embedding sizes:
(768,)
(768,)
[-2.28606537e-02  7.11676359e-01  2.40363792e-01 -2.13769078e-01
  1.80517972e-01 -3.96378338e-01 -5.34179201e-03 -4.54831012e-02
  5.49564473e-02  2.01288443e-02  3.65286529e-01 -1.06542699e-01
 -8.47535580e-02 -1.72049701e-02 -2.79080182e-01  2.17607930e-01
 -5.11982217e-02  3.31517190e-01 -3.11478853e-01 -1.00415520e-01
  1.58038333e-01  2.14160964e-01 -2.08019000e-02  1.21937573e-01
  2.55045108e-02  1.19889945e-01  1.74404293e-01 -2.62050450e-01
  1.30542710e-01  3.29620481e-01  1.61104009e-01  4.03197289e-01
 -8.35945010e-02 -1.16183497e-01  2.38502458e-01 -2.33222455e-01
 -2.13407561e-01 -8.88664052e-02 -5.68621568e-02  1.93007573e-01
  5.51218679e-03 -1.24709852e-01 -7.55844340e-02  1.16101682e-01
 -9.39535648e-02 -1.54121101e-01  2.70845145e-01 -9.93169621e-02
  1.91113591e-01 -4.95880619e-02 -7.69155204e-01 -9.40782055e-02
 -6.92091510e-02  6.72205389e-02  3.24252754e-01  1.08465649e-01
  2.14605927e-02  5.51130474e-02  1.38889506e-01 -1.01631999e-01
```

Embeddings of Sci Bert

```python
from numpy.lib import scimath
text = notes[9]
word = 'headache'

text = clean_text(text)
scibert_model.eval()
# Get the embedding for the sentence, as well as an embedding for 'pneumothorax'..
(sen_emb, word_emb) = get_embedding(scibert_model, scibert_tokenizer, text, word)
print('Embedding sizes:')
print(sen_emb.shape)
print(word_emb.shape)
print(sen_emb)
print(word_emb)
```

```
last seen normal pm found mother em bed per found c fentanyl patch various stage sign trauma per pt able comment motor deteriorated gc bp patient received total 550mcg ativan propof
[102, 2442, 2187, 1346, 3181, 797, 5303, 562, 5630, 309, 797, 115, 23534, 17582, 7940, 1711, 2410, 423, 7872, 309, 3471, 2357, 8854, 3850, 10911, 224, 6723, 4448, 1454, 2072, 1114,
Embedding sizes:
(768,)
(768,)
[-5.62204480e-01  3.40861231e-01 -2.92428434e-01  2.90717661e-01
 -2.07172737e-01  7.86308423e-02 -2.77380317e-01 -3.46231386e-02
 -2.99004525e-01  2.46844906e-02  4.21388447e-01 -8.66736397e-02
 -3.94841641e-01  4.00352627e-01 -2.03845501e-01 -2.05087334e-01
 -1.70268372e-01 -3.64337489e-02  1.76167533e-01 -1.75243005e-01
  4.04502809e-01  2.06966534e-01  1.10509560e-01  3.90179120e-02
  2.79990584e-01 -2.27908164e-01  3.92367184e-01 -2.63694495e-01
  7.84668505e-01 -8.02371427e-02 -6.50882363e-01 -3.92508477e-01
 -5.69213144e-02 -7.88997233e-01  1.71396747e-01  4.43511695e-01
  2.16898888e-01  1.37020037e-01 -8.18131939e-02 -4.11595285e-01
  3.07779968e-01 -3.64042968e-01  9.88555312e-01  4.41974580e-01
 -1.65313587e-01  2.88916945e-01 -5.62239051e-01  3.82391423e-01
 -4.73407477e-01 -8.35857466e-02  1.23212352e-01 -4.13845718e-01
 -9.78135783e-03  8.98335814e-01  3.76708210e-01 -6.49179399e-01
 -1.43938577e-02 -7.11868942e-01  1.07748294e+00  6.47266448e-01
```

References:

1. Bert: https://github.com/google-research/bert

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

2. BioBert: https://github.com/dmis-lab/biobert

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2019). BioBERT: pre-trained biomedical language representation model for biomedical text mining. arXiv preprint arXiv:1901.08746.

Notes: this tutorial is built based on these reference:

1. https://towardsml.wordpress.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/
2. Transformer:
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
3. https://www.youtube.com/watch?v=Po38Dl-XDd4
4. https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a