Group Final project: Baseline Implementation and Investigation

8-Hengke Jiang, 13-Yuting Jiang, 20-Rock, 30-Ethan, 35-Jade Lin

Table of Contents

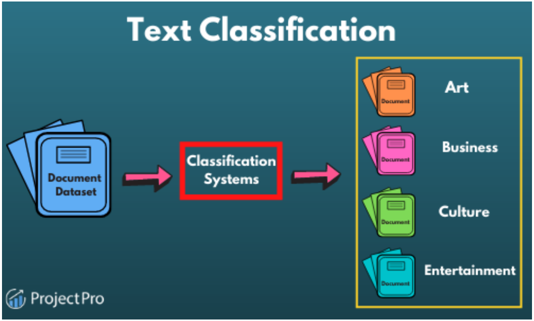
Text classification

Why Text classification is still alive?

Text classification applications

- Customer Support and Feedback Analysis: Automatically classifying customer queries into categories like technical support, billing issues, product inquiries, etc;
- Content Moderation and Filtering: Filtering and moderating user-generated content on social media platforms, forums, and online communities;
- Information Retrieval: Improving search engine results by categorizing and organizing documents or web pages based on their topics or relevance;
- Filtering incoming emails to separate spam or junk emails from legitimate messages;

Text classification



- News Categorization: Organizing news articles and updates into different categories or topics for readers.

Dataset and SOA

- 120,000 training articles, 7600 testing articles;
- Categories divided: World, Sports, Business, Science/Technology
- Format: Each article is represented as a text string along with its corresponding label indicating the topic category.

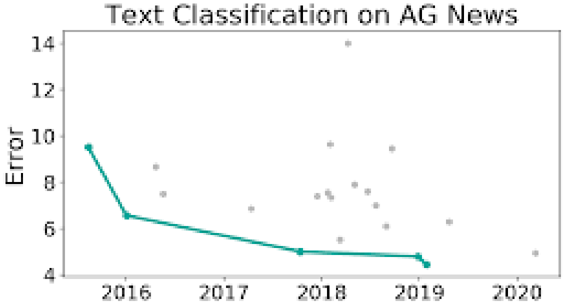
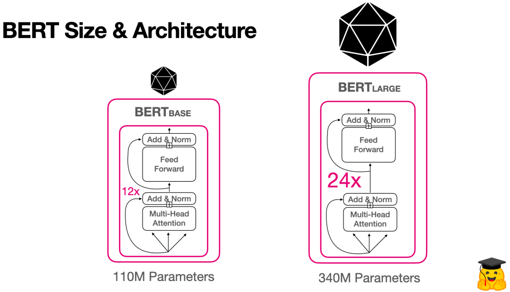


Table of Contents

BERT

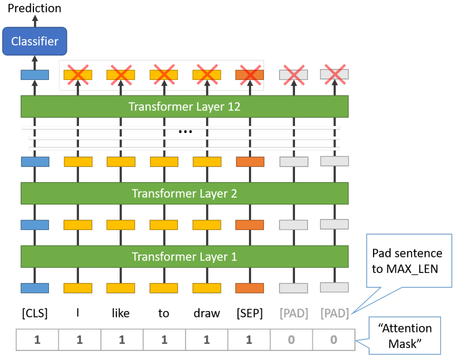
Base on encoder part of transformer[5], BERT(Bidirectional Encoder Representations from Transformers)[1] is developed to do tasks as: Text classification, Question Answering, Text Generation, Language Understanding, and Sentence Embeddings.



Attention: without decoder part, BERT is not used for translation.

Feature of BERT-base

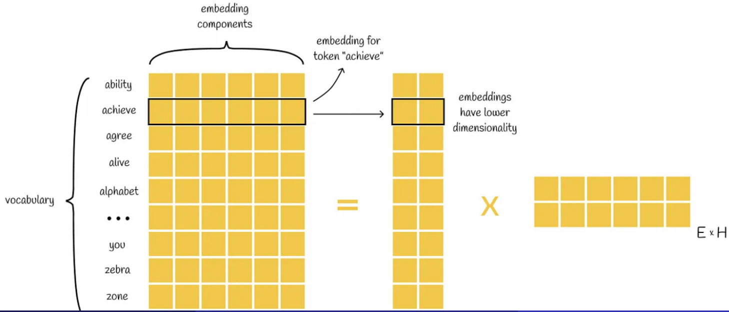
- 30,000 token vocabulary
- Transformer Blocks: 12 layers;
- Hidden Size: 768 (number of units in the hidden layers);
- Attention Heads: 12
- Number of parameters: 110M



Modified BERT: ALBERT

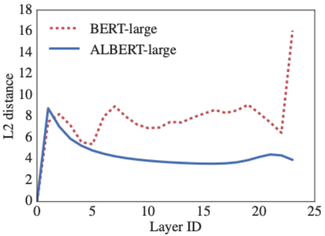
ALBERT: a lite BERT for self-supervised learning of language representations[4].
Motivation: at some point fur- ther model increases become harder due to GPU/TPU memory limitations and longer training times. Improve BERT in 2 ways with the same number of parameters(110M).

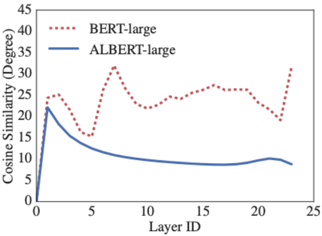
- Tech 1: Decomposing the large vocabulary embedding matrix into two small matrices to separate the size of the hidden layers from the size of vocabulary embedding.



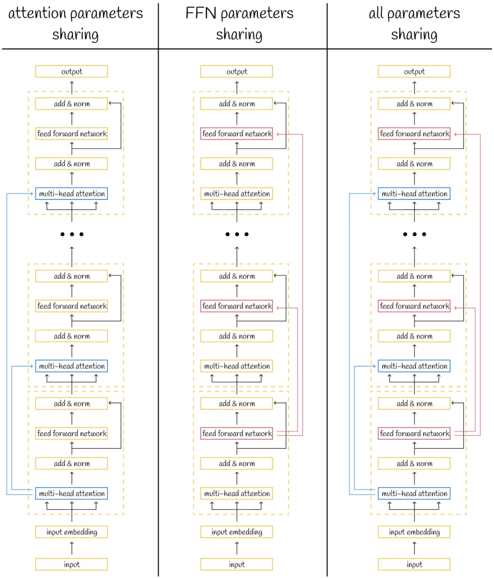
Modified BERT: ALBERT

- Tech 2: cross-layer parameter sharing prevents the parameter from growing with the depth of the network;





Modified BERT: ALBERT



Modified BERT: DEBERTA

DeBERTa: Decoding-enhanced BERT with Disentangled Attention[3]
Motivation: Improvement with the number of parameters close to BERT(89M vs. 110M) in two ways.

- 1. Disentangled attention: Each word is represented using two vectors that encode its content and position, respectively, and the attention weights among words are computed using disentangled matrices based on their contents and relative positions, respectively.

For a token at position i in a sequence, we represent it using two vectors, $\{\boldsymbol{H_i}\}$ and $\{\boldsymbol{P_{i|j}}\}$, which represent its content and relative position with the token at position j, respectively. The calculation of the cross attention score between tokens i and j can be decomposed into four components as

$$
\begin{aligned}
A_{i,j} &= \{\boldsymbol{H_i}, \boldsymbol{P_{i|j}}\} \times \{\boldsymbol{H_j}, \boldsymbol{P_{j|i}}\}^\intercal \\
&= \boldsymbol{H_i}\boldsymbol{H_j}^\intercal + \boldsymbol{H_i}\boldsymbol{P_{j|i}}^\intercal + \boldsymbol{P_{i|j}}\boldsymbol{H_j}^\intercal + \boldsymbol{P_{i|j}}\boldsymbol{P_{j|i}}^\intercal
\end{aligned} \tag{2}
$$

Modified BERT: DEBERTA

- 2. Enhanced mask decoder: Each word in DeBERTa is represented using two vectors that encode its content and position, respectively, and the attention weights among words are computed using disentangled matrices based on their contents and relative positions, respectively.

Table of Contents

parameter and devices(pretrained)

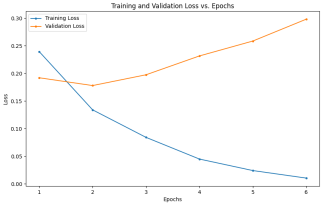
- number of epoches $= 6$, batchsize $= 32$;
- divide training set into 2 part and shuffles, each epoch cost 4$\sim$8 min;
- optimizer: AdamW, with linearly decreasing learning rate(6e-5 $\sim$ 1e-5)
- 3 models implement.

```
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 1/6 completed. Train Loss: 0.2393, Train Accuracy: 0.9185, Val Loss: 0.1920, Val Accuracy: 0.9347, F1 Score: 0.9184
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 2/6 completed. Train Loss: 0.1338, Train Accuracy: 0.9545, Val Loss: 0.1778, Val Accuracy: 0.9408, F1 Score: 0.9545
100%|██████████| 1875/1875 [04:45<00:00,  6.58it/s]
Epoch 3/6 completed. Train Loss: 0.0840, Train Accuracy: 0.9706, Val Loss: 0.1974, Val Accuracy: 0.9432, F1 Score: 0.9706
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 4/6 completed. Train Loss: 0.0448, Train Accuracy: 0.9842, Val Loss: 0.2315, Val Accuracy: 0.9411, F1 Score: 0.9843
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 5/6 completed. Train Loss: 0.0239, Train Accuracy: 0.9918, Val Loss: 0.2585, Val Accuracy: 0.9426, F1 Score: 0.9918
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 6/6 completed. Train Loss: 0.0103, Train Accuracy: 0.9965, Val Loss: 0.2980, Val Accuracy: 0.9447, F1 Score: 0.9965
Test Accuracy: 0.9447
Test F1 Score: 0.9447
```

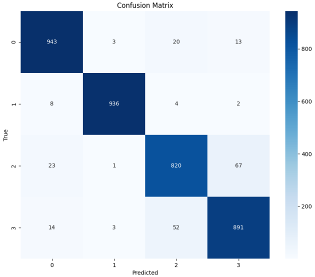
```
100%|██████████| 1875/1875 [05:21<00:00,  5.83it/s]
Epoch 1/6 completed. Train Loss: 0.5346, Train Accuracy: 0.7884, Val Loss: 1.3883, Val Accuracy: 0.2576, F1 Score: 0.7877
100%|██████████| 1875/1875 [05:21<00:00,  5.84it/s]
Epoch 2/6 completed. Train Loss: 0.7819, Train Accuracy: 0.6556, Val Loss: 0.3580, Val Accuracy: 0.8900, F1 Score: 0.6554
100%|██████████| 1875/1875 [05:21<00:00,  5.84it/s]
Epoch 3/6 completed. Train Loss: 0.2929, Train Accuracy: 0.9129, Val Loss: 0.3441, Val Accuracy: 0.8995, F1 Score: 0.9127
100%|██████████| 1875/1875 [05:20<00:00,  5.84it/s]
Epoch 4/6 completed. Train Loss: 0.2323, Train Accuracy: 0.9287, Val Loss: 0.2627, Val Accuracy: 0.9134, F1 Score: 0.9286
100%|██████████| 1875/1875 [05:21<00:00,  5.84it/s]
Epoch 5/6 completed. Train Loss: 0.1744, Train Accuracy: 0.9452, Val Loss: 0.2696, Val Accuracy: 0.9176, F1 Score: 0.9452
100%|██████████| 1875/1875 [05:21<00:00,  5.84it/s]
Epoch 6/6 completed. Train Loss: 0.1292, Train Accuracy: 0.9594, Val Loss: 0.2556, Val Accuracy: 0.9221, F1 Score: 0.9594
Test Accuracy: 0.9221
Test F1 Score: 0.9222
```

```
100%|██████████| 1875/1875 [06:19<00:00,  4.94it/s]
Epoch 1/6 completed. Train Loss: 0.2517, Train Accuracy: 0.9151, Val Loss: 0.1908, Val Accuracy: 0.9361, F1 Score: 0.9150
100%|██████████| 1875/1875 [06:19<00:00,  4.95it/s]
Epoch 2/6 completed. Train Loss: 0.1477, Train Accuracy: 0.9493, Val Loss: 0.1797, Val Accuracy: 0.9418, F1 Score: 0.9493
100%|██████████| 1875/1875 [06:19<00:00,  4.94it/s]
Epoch 3/6 completed. Train Loss: 0.0986, Train Accuracy: 0.9667, Val Loss: 0.1938, Val Accuracy: 0.9408, F1 Score: 0.9667
100%|██████████| 1875/1875 [06:19<00:00,  4.94it/s]
Epoch 4/6 completed. Train Loss: 0.0634, Train Accuracy: 0.9788, Val Loss: 0.2168, Val Accuracy: 0.9429, F1 Score: 0.9788
100%|██████████| 1875/1875 [06:19<00:00,  4.94it/s]
Epoch 5/6 completed. Train Loss: 0.0368, Train Accuracy: 0.9878, Val Loss: 0.2362, Val Accuracy: 0.9413, F1 Score: 0.9878
100%|██████████| 1875/1875 [06:19<00:00,  4.95it/s]
Epoch 6/6 completed. Train Loss: 0.0192, Train Accuracy: 0.9937, Val Loss: 0.2924, Val Accuracy: 0.9418, F1 Score: 0.9937
Test Accuracy: 0.9418
Test F1 Score: 0.9416
```

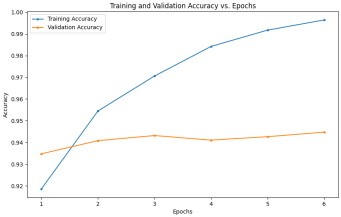
BERT



Training and Validation Loss vs. Epochs
Training Loss
Validation Loss
Loss
Epochs

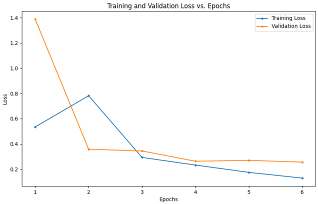


Confusion Matrix
943
3
20
13
8
936
4
2
23
1
820
67
14
3
52
891
True
Predicted

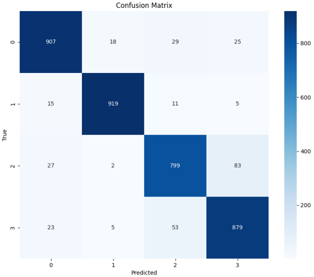


Training and Validation Accuracy vs. Epochs
Training Accuracy
Validation Accuracy
Accuracy
Epochs

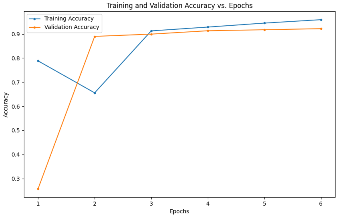
ALBERT



Training and Validation Loss vs. Epochs
Training Loss
Validation Loss
Loss
Epochs

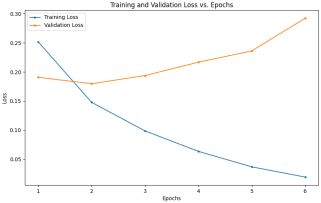


Confusion Matrix
907
18
29
25
15
919
11
5
27
2
799
83
23
5
53
879
True
Predicted

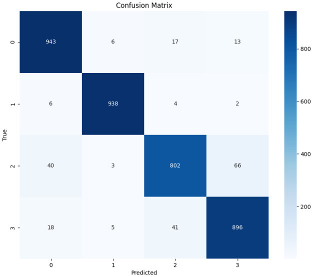


Training and Validation Accuracy vs. Epochs
Training Accuracy
Validation Accuracy
Accuracy
Epochs

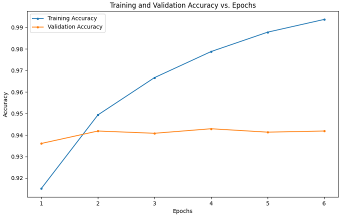
DEBERTA



Training and Validation Loss vs. Epochs
Training Loss
Validation Loss
Loss
Epochs



Confusion Matrix
True
Predicted
943
6
17
13
6
938
4
2
40
3
802
66
18
5
41
896



Training and Validation Accuracy vs. Epochs
Training Accuracy
Validation Accuracy
Accuracy
Epochs

Summarazie: BERT > ALBERT > DeBERTa, partly becanse testing part is divided along with training part.

```
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 1/6 completed. Train Loss: 0.2393, Train Accuracy: 0.9185, Val Loss: 0.1920, Val Accuracy: 0.9347, F1 Score: 0.9184
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 2/6 completed. Train Loss: 0.1338, Train Accuracy: 0.9545, Val Loss: 0.1778, Val Accuracy: 0.9408, F1 Score: 0.9545
100%|██████████| 1875/1875 [04:45<00:00,  6.58it/s]
Epoch 3/6 completed. Train Loss: 0.0840, Train Accuracy: 0.9706, Val Loss: 0.1974, Val Accuracy: 0.9432, F1 Score: 0.9706
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 4/6 completed. Train Loss: 0.0448, Train Accuracy: 0.9842, Val Loss: 0.2315, Val Accuracy: 0.9411, F1 Score: 0.9843
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 5/6 completed. Train Loss: 0.0239, Train Accuracy: 0.9918, Val Loss: 0.2585, Val Accuracy: 0.9426, F1 Score: 0.9918
100%|██████████| 1875/1875 [04:44<00:00,  6.58it/s]
Epoch 6/6 completed. Train Loss: 0.0103, Train Accuracy: 0.9965, Val Loss: 0.2980, Val Accuracy: 0.9447, F1 Score: 0.9965
Test Accuracy: 0.9447
Test F1 Score: 0.9447
```

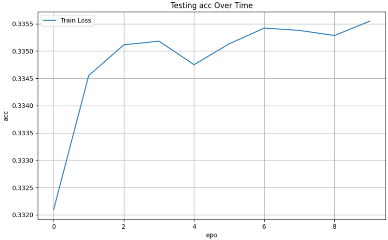
```
100%|██████████| 1875/1875 [05:21<00:00,  5.83it/s]
Epoch 1/6 completed. Train Loss: 0.5346, Train Accuracy: 0.7884, Val Loss: 1.3883, Val Accuracy: 0.2576, F1 Score: 0.7877
100%|██████████| 1875/1875 [05:21<00:00,  5.84it/s]
Epoch 2/6 completed. Train Loss: 0.7819, Train Accuracy: 0.6556, Val Loss: 0.3580, Val Accuracy: 0.8900, F1 Score: 0.6554
100%|██████████| 1875/1875 [05:21<00:00,  5.84it/s]
Epoch 3/6 completed. Train Loss: 0.2929, Train Accuracy: 0.9129, Val Loss: 0.3441, Val Accuracy: 0.8995, F1 Score: 0.9127
100%|██████████| 1875/1875 [05:20<00:00,  5.84it/s]
Epoch 4/6 completed. Train Loss: 0.2323, Train Accuracy: 0.9287, Val Loss: 0.2627, Val Accuracy: 0.9134, F1 Score: 0.9286
100%|██████████| 1875/1875 [05:21<00:00,  5.84it/s]
Epoch 5/6 completed. Train Loss: 0.1744, Train Accuracy: 0.9452, Val Loss: 0.2696, Val Accuracy: 0.9176, F1 Score: 0.9452
100%|██████████| 1875/1875 [05:21<00:00,  5.84it/s]
Epoch 6/6 completed. Train Loss: 0.1292, Train Accuracy: 0.9594, Val Loss: 0.2556, Val Accuracy: 0.9221, F1 Score: 0.9594
Test Accuracy: 0.9221
Test F1 Score: 0.9222
```

```
100%|██████████| 1875/1875 [06:19<00:00,  4.94it/s]
Epoch 1/6 completed. Train Loss: 0.2517, Train Accuracy: 0.9151, Val Loss: 0.1908, Val Accuracy: 0.9361, F1 Score: 0.9150
100%|██████████| 1875/1875 [06:19<00:00,  4.95it/s]
Epoch 2/6 completed. Train Loss: 0.1477, Train Accuracy: 0.9493, Val Loss: 0.1797, Val Accuracy: 0.9418, F1 Score: 0.9493
100%|██████████| 1875/1875 [06:19<00:00,  4.94it/s]
Epoch 3/6 completed. Train Loss: 0.0986, Train Accuracy: 0.9667, Val Loss: 0.1938, Val Accuracy: 0.9408, F1 Score: 0.9667
100%|██████████| 1875/1875 [06:19<00:00,  4.94it/s]
Epoch 4/6 completed. Train Loss: 0.0634, Train Accuracy: 0.9788, Val Loss: 0.2168, Val Accuracy: 0.9429, F1 Score: 0.9788
100%|██████████| 1875/1875 [06:19<00:00,  4.94it/s]
Epoch 5/6 completed. Train Loss: 0.0368, Train Accuracy: 0.9878, Val Loss: 0.2362, Val Accuracy: 0.9413, F1 Score: 0.9878
100%|██████████| 1875/1875 [06:19<00:00,  4.95it/s]
Epoch 6/6 completed. Train Loss: 0.0192, Train Accuracy: 0.9937, Val Loss: 0.2924, Val Accuracy: 0.9418, F1 Score: 0.9937
Test Accuracy: 0.9418
Test F1 Score: 0.9416
```

Implement withou pretrain

- Dataset: R8, smaller than AG News;
- number of epoch = 10, batchsize = 4(for time saving)

```python
class BERT(nn.Module):
    def __init__(self, vocab_size, hidden=768, n_layers=12, attn_heads=12, dropout=0.1):

        super().__init__()
        self.hidden = hidden
        self.n_layers = n_layers
        self.attn_heads = attn_heads

        # paper noted they used 4*hidden_size for ff_network_hidden_size
        self.feed_forward_hidden = hidden * 4

        # embedding for BERT, sum of positional, segment, token embeddings
        self.embedding = BERTEmbedding(vocab_size=vocab_size, embed_size=hidden)

        # multi-layers transformer blocks, deep network
        self.transformer_blocks = nn.ModuleList(
            [TransformerBlock(hidden, attn_heads, hidden * 4, dropout) for _ in range(n_layers)])

    def forward(self, x, segment_info):
        # attention masking for padded token
        # torch.ByteTensor([batch_size, 1, seq_len, seq_len)
        mask = (x > 0).unsqueeze(1).repeat(1, x.size(1), 1).unsqueeze(1)

        # embedding the indexed sequence to sequence of vectors
        x = self.embedding(x, segment_info)

        # running over multiple transformer blocks
        for transformer in self.transformer_blocks:
            x = transformer.forward(x, mask)

        return x
```


Testing acc Over Time
Train Loss
0.3355
0.3350
0.3345
0.3340
0.3335
0.3330
0.3325
0.3320
acc
0
2
4
6
8
epo

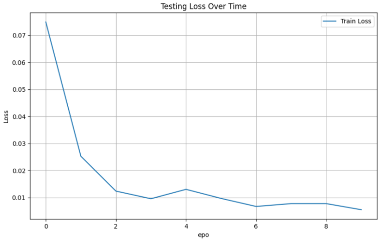

Testing Loss Over Time
Train Loss
0.07
0.06
0.05
0.04
0.03
0.02
0.01
Loss
0
2
4
6
8
epo

Table of Contents

Old tech: change the parameter used, like learning rate and batchsize;
New tech: DEBERTA V3[2]

1. improves the original DeBERTa model by replacing mask language modeling (MLM) with replaced token detection (RTD), a more sample-efficient pre-training task;
2. not yet updated in text classification task.

Bibliography I

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Pengcheng He, Jianfeng Gao, and Weizhu Chen.
Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2023.

[3] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen.
Deberta: Decoding-enhanced bert with disentangled attention, 2021.

[4] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut.
Albert: A lite bert for self-supervised learning of language representations, 2020.

Bibliography II

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.