DAPT

Datasize, number steps/epochs, learning rate, other params

# Official BERT paper

* BERT is **optimized with Adam (Kingma and Ba, 2015) using the following parameters: β1 = 0.9, β2 = 0.999, ǫ = 1e-6 and L2 weight decay of 0.01**. **The learning rate is warmed up over the first 10,000 steps to a peak value of 1e-4, and then linearly decayed.** BERT trains with a **dropout of 0.1 on all layers and attention weights, and a GELU activation function (Hendrycks and Gimpel, 2016).** Models are **pretrained for S = 1,000,000 updates, with minibatches containing B = 256 sequences of maximum length T = 512 tokens.**
* BERT is trained on a combination of BOOKCOR-PUS (Zhu et al., 2015) plus English WIKIPEDIA, which totals **16GB** of uncompressed text.3
* Yang et al. (2019) use the same dataset but report having only 13GB of text after data cleaning. This is most likely due to subtle differences in cleaning of the Wikipedia data.

# <https://github.com/google-research/bert#pre-training-tips-and-caveats> – official BERT github project

* “If your task has a large domain-specific corpus available (e.g., "movie reviews" or "scientific papers"), it will likely be beneficial to run additional steps of pre-training on your corpus, starting from the BERT checkpoint.”
* The learning rate we used in the paper was 1e-4. However, **if you are doing additional steps of pre-training starting from an existing BERT checkpoint, you should use a smaller learning rate (e.g., 2e-5)**.
* Longer sequences are disproportionately expensive because attention is quadratic to the sequence length. In other words, a batch of 64 sequences of length 512 is much more expensive than a batch of 256 sequences of length 128. The fully-connected/convolutional cost is the same, but the attention cost is far greater for the 512-length sequences. Therefore, **one good recipe is to pre-train for, say, 90,000 steps with a sequence length of 128 and then for 10,000 additional steps with a sequence length of 512**. The very long sequences are mostly needed to learn positional embeddings, which can be learned fairly quickly. Note that this does require generating the data twice with different values of max\_seq\_length.

# HateBERT: Retraining BERT for Abusive Language Detection in English

* From the RAL-E dataset, **we used 1,478,348 messages (for a total of 43,379,350 tokens)** to re-train the English BERT base-uncased model3 by applying the Masked Language Model (MLM) objective. The remaining 14,932 messages (441,271 tokens) have been used as test set. **We retrained for 100 epochs (almost 2 million steps)** in **batches of 64 samples, including up to 512 sentencepiece tokens. We used Adam** with **learning rate 5e-5.** We trained using the huggingface code4 on one Nvidia V100 GPU.

# Domain and Task Adaptive Pretraining for Language Models

* Domain adaptive pretraining is performed by iterating **5 epochs** over **Corpus *D (739M tokens)***with a **learning rate of 1e-4**.

# BioBERT

* BioBERT v1.0 trained for **470k steps** auf **Pubmed (4.5B Tokens) und PMC (13.5B Tokens)** corpora
* **Other hyper-parameters such as batch size and learning rate scheduling for pre-training BioBERT are the same as those for pre-training BERT unless stated otherwise.**
* The **maximum sequence length was fixed to 512 and the mini-batch size was set to 192**, resulting in 98 304 words per iteration

# RoBERTa

* 160 GB daten
* 100k, 300k bzw. 500k Steps (selbst 500k steps führen nicht zum overfitten, wodurch evtl. Noch länger trainiert werden kann)

# FoodBERT

* Startpunkt: BERT-base-cased mit erweitertem Vokabular (zusätzliche 4.372 Tokens)
* Weitertrainieren auf Recipe1M für 3 Epochen, nur Tokenizer wurde gemäß dem Vokabular angepasst, der Rest bleibt gleich
* Max sequ length = 128

# Pre-Training BERT on Domain Resources for Short Answer Grading

* We leverage the TensorFlow implementation of BERT-Base§ for all our experiments. It is **further pre-trained for 240K, 150K, and 240K epochs** for **BERT+TextbookPhy+Gov (1.1M words), BERT+QAPhy+Gov,(0.6M words) and BERT+QAPhy+Gov+Psy-III (1.3M words)** respectively **using the same hyperparameters until the accuracy of the two pre-training objectives converges to 100%**

# MenuNER

* We train the BERT MLM on food corpus () for **80,000 steps**, which took (34,708.82 s) ~9.5 h **for 3 epochs** on Nvidia TITAN X GPU. We use Adam optimizer with 1 × 10−8 and a **learning rate of 3 × 10−5**. Figure 5 shows the training loss at every 50 steps.

# CSBERT

* Knapp 40mio Customer-agent dialoge
* To leverage the useful knowledge from the pretrained BERT, we do not train CS-BERT from scratch, but start from the pretrained BERT-base checkpoint. As in (1), we train with a total **batch size of 256** sequences **for 1,000,000 steps.** However, we use a **maximum sequence length of 256 throughout the whole process**. We use the same Adam optimizer as in (1 🡪 official BERT paper) **with learning rate of 1e−4, β1 = 0.9, β2 = 0.999, L2 weight decay of 0.01, learning rate warmup over the first 10,000 steps, and linear decay of the learning rate**. All our CS-BERT models were trained on 8 GPUs, and the total training time is around 14 days.

# BERT: Post-Training for Review Reading Comprehension and …

* **1.121.863 training examples von amazon laptop reviews**, **700k reviews bzw. 2.6 mio examples for restaurant domain knowledge**
* We adopt **BERTBASE (uncased)** as the basis for all experiments10. Since post-training may take a large footprint on GPU memory (as BERT pretraining), we leverage FP16 computation11 to reduce the size of both the model and hidden representations of data. We set a static loss scale of 2 in FP16, which can avoid any over/under-flow of floating point computation. The m**aximum length of post-training is set to 320 with a batch size of 16 for each type of knowledge**. The **number of subbatch u is set to 2**, which is good enough to store each sub-batch iteration into a GPU memory of 11G. We use **Adam optimizer and set the learning rate to be 3e-5. We train 70,000 steps for the laptop domain and 140,000 steps for the restaurant**

Finetuning

# BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding