A Primer in BERTology

Alireza Moradi

This paper reviews current knowledge about how BERT works and various research done in this area. In the first part it gives an overview of BERT's architecture which consists of a stack of transformer encoders (which consists of multiple self-attention heads). Then continues with how it works, Which is computing KVQ matrices by each head and creating a weighted representation. Finally outputs of heads in the same layer are concatenated and run through a fully-connected (FC) layer. We know that the convention workflow for BERT consists of 2 stages:

- **Pre-Training**: Which is the most expensive part of training. In this stage we basically train the network to learn a general knowledge which itself uses 2 tasks:
  - Masked Language Modeling (MLM): prediction of randomly masked input tokens.
  - Next Sentence Prediction (NSP): predicting if 2 input sentences are adjacent.
- **Fine-Tuning**: Which basically means tuning the network to be task specific (also called downstream applications) by adding one or more FC layers.

Continuing with how input representations are computed we know that:

- Words are tokenized into wordpieces.
- Run through a special embedding layer (concatenation of token, position and segment layers).
- [CLS] token is used for classification predictions and [SEP] token is used for separating input segments.

The original BERT has 2 versions: "base" and "large" which differ in the number of heads, layers, and hidden state size.
The paper the continues with analysing what knowledge does BERT have in itself and how are they encoded. Some are:

- **Syntactic Knowledge**
  - BERT representations are hierarchical rather than linear (which we saw that Hewitt-Manning used this in their paper).
  - BERT embeddings encode information about POS, syntactic chunks and roles. (enough to recover syntactic trees)
  - Syntactic structure is not directly encoded in self-attention weights.
  - Syntactic information can be recovered from BERT token representation.
  - BERT takes subject-predicate agreement into accounting when performing the cloze task.
  - BERT is better able to detect the presence of NPIs (e.g. "ever") and the words that allow their use (e.g. "whether") than scope violations.
  - BERT doesn't understand negation (bad sign) and is insensitive to malformed input. Which could mean that either BERT's syntactic knowledge is incomplete (**why?**) or it doesn't need to rely on it for solving tasks(**why?**).
- **Semantic Knowledge**
  - BERT has some knowledge of semantic roles.

    - BERT encodes info about entity types, relations, semantic roles and proto-roles(what is this?).
    - BERT struggles with representation of numbers. Mainly because BERT uses wordpiece tokenizer.
    - BERT is very sensitive to named entity replacement.
- **World Knowledge**
    - BERT struggles with pragmatic inference and role-based event knowledge, and with abstract attributes of objects, as well as visual and perceptual properties that are likely to be assumed rather than mentioned.
    - For some relation types, vanilla BERT is competitive with methods relying on knowledge bases.
    - BERT can't reason based on its world knowledge (can guess).

So BERT has a great deal of semantic, syntactic and world knowledge.

The paper the goes on with some limitations in knowing BERT better.

“Embedding” in BERT related studies refers to the output of the [usually] final Transformer layer. The difference between BERT embeddings and conventional static embeddings is that BERT embeddings are contextualized, Meaning that vectors representing tokens have at least some info about that particular context.

Regarding embeddings measurements show that later layers of BERT produce more context-specific representations. These embeddings form distinct clusters corresponding to word sense, making them powerful at word sense disambiguation. BERT embeddings occupy a narrow cone in the vector space(**what does this mean?**), and this effect increases from the earlier to later layers.

Heads with linguistic functions (didn't understand much)

Some researches show that most of self-attention heads do not directly encode any non-trivial linguistic information.

Many researches show that syntactic information is most prominent in the middle layers of BERT, saying that “the basic syntactic information appears earlier in the network while high-level semantic features appear at later layers” while another paper suggests that semantic is spread across the entire model, which is expected because semantics are spread through all language.

Observations by researchers show that middle layers of Transformers are best performing overall, and final layers of BERT are the most task specific, meaning that middle layers are more transferable (pre-training) and explaining why the final layers change the most during fine-tuning.

About the experiments in BERT architecture, Results show that the number of heads is not as significant as the number of layers and the middle layers are the most transferable (because they are more task-invariant). So a deeper model means more capacity to encode task-invariant information. Other experiments show that many of the self-attention heads in BERT seem to learn the same patterns, That’s why pruning most of heads doesn’t have much impact.

About changes to the training regime, The important one is that self-attention patterns in higher and lower layers are similar, so we can train the model recursively, meaning we train a shallower network and copy the trained parameters to the deeper layers.

Regarding pre-training of BERT (which is supposed to provide task-independent knowledge) multiple studies have come up with ideas on changing training objectives:

- How to mask
- What to mask
- Where to mask
- Alternatives to masking
- NSP alternatives

Another source of improvement is the size of pre-training data.

About fine-tuning (teaching the model to rely more on the representations useful for a specific task), these are the possible improvements:

- More layers
- 2-stage fine-tuning
- Adversarial token perturbations
- Adversarial regularization
- Mixout regularization

Overparameterization is a concern because, although human language in incredibly complex and might require much more parameters than what current state-of-the-art models have, current models don’t make good use of these parameters that they already have (like earlier we said that most of the self-attention heads learn similar patterns, meaning they are almost useless and can be pruned without significant impact on performance), even some experiments show that some BERT heads/layers are harmful to some downstream tasks performances. In general, large BERT models perform better, but not always.

Relying on above observations, we can efficiently compress BERT with minimal impact on accuracy. Pruning is one of the ways of compressing which can be done in 2 ways:

- **Structured pruning**: architecture blocks are dropped, as in LayerDrop
- **Unstructured pruning**: the weights in the entire model are pruned irrespective of their location, as in magnitude pruning or movement pruning.

Researches suggest that training a large model and compressing it heavily is better that compressing a smaller model lightly. Other techniques are: *decomposing BERT’s embedding matrix into smaller matrices*, *progressive module replacing*, *dynamic elimination of intermediate en- coder outputs*.