**Reference**

T5: Limits of Transfer Learning (https://arxiv.org/pdf/1910.10683.pdf)

# Exploring the limits of Transfer Learning for NLP

Many recent advances in NLP owe to the Transfer Learning "Pre-train and Fine Tune" approach

- A resource-rich entity trains a model on a Source Task, using a vast training dataset:
    - this is the *pre-training* step
    - Need a lot of resources to create, process and store the training dataset
    - Need a lot of resources to train a big (lots of weights) model on a big training dataset
    - Training objective usually Language Model (LM, predict the next) or Masked Language Model (MLM)
- A less resource rich entity adapts the model to a Target Task
    - checks out the model architecture and weights from a Model Hub
    - *Fine tunes* the model's weights on a much smaller Source Task-specific training dataset

Pre-training on simple tasks (like LM and MLM)

- seems to require the model to develop representations
- of *general* knowledge
- that transfer to many tasks

Each successful fine-tuned model results from a combination of discrete choices

- Pre-training objective
    - LM, MLM, Prefix LM (to be discussed)
- Dataset
    - size
    - diversity
- Architecture
- Fine-tuning method
- Benchmark

Given a combination of choices, it is not always easy to identify which choices contributed/detracted from the success of the Fine-Tuned model.

- *Ablation* studies are often conducted
    - Replace an innovative choice with the standard; compare before/after metrics

The authors conduct a systematic study in an attempt to determine

- which choices are best
- create a model (T5) inspired by the best choices

# Enablers

## Text to Text as Universal NLP API

A key enabler of the study was the formulation of a Universal NLP API: Text to Text which we studied (NLP_Text_to_Text.ipynb).

All NLP tasks can be re-expressed

- Translating (potentially structured) Input Text to flat sequences of Processed Text
- Outputting flat sequences of Text.

In particular: both the pre-training and fine tuning use the Text to Text format.

# Colossal Clean Crawled Corpus (C4 ) training dataset

Common Crawl (https://commoncrawl.org/) is consists of data obtained via *web-scraping* over many years

- Large: 20TB **per month**
- Lacking quality for NLP tasks
    - not Natural Language: source code, menus, error messages
    - offensive content

In response to this, the authors created Colossal Clean Crawled Corpus (C4), a cleansed version of Common Crawl

- retain only sentences
    - ending in proper NL punctuation (e.g., period, exclamation)
    - with at least 3 words
    - without indicators of code, e.g, `!javascript`
- **de-duplicated**
- filter out "bad words"
- filter out **non-English** pages
    - since most tasks in set of benchmarks used were English-only

See the paper's section (https://arxiv.org/pdf/1910.10683.pdf#page=5) for full details

# Downstream tasks

The research uses a predetermined set of benchmarks that measure the performance on *downstream* tasks such as

- Sentence acceptability
- Sentiment analysis
- Sentence similarity
- Sentence completion
- Question answering
- Summarization

# Choices examined

## Architecture

All models studied used the Transformer architecture

- dominates simpler RNN, LSTM alternatives
- by facilitating long range dependencies

The original Transformer architecture consists of an Encoder and Decoder

- Encoder creates alternate representation of input
    - Ordinary self-attention (not causal) to the (layer's) inputs
- Decoder acts auto-regressively to create output
    - Uses masked attention to enforce causal ordering
        - can only attend to past outputs
        - can't "peek ahead" (during training) into output that will only be created in a future step

**Architecture choices**

Encoder/Decoder

Encoder only

- BERT is an example of an Encoder only architecture

Decoder only

- typically used for Language modeling

## Attention choices

The basis for the Transformer is the *Attention* mechanism.

There are different "flavors" of Attention.

- restricts which parts of the input/output may be *attended to* (accessed) by the model at each time step $t$
- where output $\mathbf{y}_{(t)}$ is generated at time step $t$

Consider a task with

- input sequence $\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(\bar{T})}$
- output sequence $\mathbf{y}_{(1)}, \ldots, \mathbf{y}_{(T)}$

*Fully visible* attention

- usually refers to which parts of the input $\mathbf{x}$ may be attended to by the Encoder
- everything: $\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(\bar{T})}$

*Causal attention*

- usually refers to which parts of the output $\mathbf{y}$ may be attended to by the Decoder
- only those parts already generated $\mathbf{y}_{(1)}, \ldots \mathbf{y}_{(t-1)}$
    - can't peek into the future
        - at training time, the *complete* target is available, and hence, the future could be visible
        - impossible at test time

In the Text to Text encoding of a task, the input and output sequences

- are usually concatenated into a single sequence: the *Processed Input*
- with separator tokens

Let concatenated sequence $\dot{\mathbf{x}}_{(1)}, \ldots, \dot{\mathbf{x}}_{(\bar{T})+T}$ be defined

$$\dot{\mathbf{x}}_{(t)} = \begin{cases} \mathbf{x}_{(t)} & t \leq \bar{T} \\ \mathbf{y}_{(t-\bar{T})} & t > \bar{T} \end{cases}$$

where we are ignoring the separator tokens for simplicity of notation.

We need to re-state the definition of the flavors of attention

- relative to the Processed Input (concatenated sequence)
- without reference to the Encoder or Decoder part
    - e.g., there is a Decoder-only architecture possible, which can also attend to $\mathbf{x}$
        - which is part of the Processed Input

**Attention choices**

*Fully visible* attention:

- everything is visible: $\dot{\mathbf{x}}_{(1)}, \ldots, \dot{\mathbf{x}}_{(\bar{T})+T}$

*Causal* attention: $1 \leq t' < t$

- only parts prior to the current output are visible: $\dot{\mathbf{x}}_{(1)}, \ldots, \dot{\mathbf{x}}_{(t-1)}$

*Prefix Language Model (LM)* attention

- the input $\mathbf{x}$ is fully visible; the only part of $\mathbf{y}$ visible is that which has been generated

$$\begin{cases} \dot{\mathbf{x}}_{(1)}, \ldots, \dot{\mathbf{x}}_{\bar{T}} & \text{visible at } t \leq \bar{T} \quad \text{input fully visible} \\ \dot{\mathbf{x}}_{(1)}, \ldots, \dot{\mathbf{x}}_{(t-1)} & \text{visible at } t > \bar{T} \quad \text{inputs, and outputs that have been} \end{cases}$$

## Model size

A Transformer model can be characterized by a number of parameters

- Number of layers
    - number of Transformer blocks in the Encoder or Decoder stacks
- $d_{\text{head}}, d_{\text{kv}}$
    - size of each head
    - corresponds to the size of the keys and values in the Attention Lookup
- $n_{\text{heads}}$
    - number of heads
- $d_{\text{model}} = n_{\text{heads}} * d_{\text{head}}$

# Unsupervised objective

**Unsupervised objective choices**

Language Modeling (LM)

- predict the next token

Masked Language Modeling (MLM)

- a *denoising* objective
- corrects corrupted (*masked*) inputs
    - 15% of input tokens corrupted, i.e., replaced by
        - random token, 10% of the time
        - missing token `[MASK]`, 90% of the time

n.b., MLM claimed to the "standard" because it has been observed to perform better than LM (predict the next) objective

A less familiar objective is *deshuffling*

- input: re-ordered tokens
- output: tokens in correct order

# Datasets

**Unsupervised Pre-training Dataset choices**

C4

Unfiltered C4

- remove filtering for non-text, bad words
- retain English-only filter

RealNews-like C4

- C4 limited to news websites used for the RealNews dataset

Wikipedia C4

Wikipedia + Toronto Book Corpus (CBC)

- books (ebooks) are different stylistically from Wikipedia articles

WebText-like C4

- WebText was created for GPT
- filtered to pages ranked with high human scores on Reddit

## Size of training dataset

Varying total number of tokens used in pre-training

# Training Strategy

## Downstream task: Fine tuning choices

There are several variants of the Fine-Tuning step.

In our original presentation on Transfer Learning

- we grafted a Target-task specific Classification head
- on to a prefix of the pre-trained architecture

We have the choice of which parameters to adjust during fine-tuning:

- just the parameters in the newly grafted (uninitialized) Head
    - *freezing* the parameters of the base model
    - to possible corruption of base model parameters
        - due to initial large gradients caused by untrained Head
- *Gradual unfreezing* of the base parameters
    - freeze parameters of deep layers of base until Head parameters have been somewhat trained
    - gradually unfreeze parameters of earlier layers as deeper layer parameters have adapted
- all the parameters

This doesn't directly apply to the Text to Text format

- no Classification Head
- entire Decoder needs to be trained to produce the Target Task sequence

Instead, the *freezing* strategy used is to insert *adapter layers*

- Dense-ReLu-Dense blocks inserted after the Feed Forward (FF) final component of the Transformer block
    - for each layer
- number of inputs and outputs are the same
- so can be inserted without affecting rest of the architecture
- comparable to training only the Classifier Head

The analog of gradual unfreezing for the Transformer is to allow

- gradual unfreezing of parallel layers in the Encoder and Decoder

## Upstream task: Include Multi Task Learning as part of Pre-Training ?

Rather than *only* using Unsupervised Pre-Training upstream (i.e., before Fine Tuning)

- we could try Multi Task Learning
- training set a mixture of examples from task-specific training datasets
- several variants of determining the proportions in the mixture

Note that Unsupervised Pre Training **is one of the tasks included in the mixture**

# Baseline

A baseline model is fixed by making a set of choices.

Experiments are conducted by varying choices one at a time.

Here are the choices for the baseline model

**Architecture**

Encoder/Decoder Transformer

- similar to BERT base model
    - 12 layers
    - $d_{\text{model}} = n_{\text{heads}} * d_{\text{head}} = 768$
        - $n_{\text{heads}} = 12, d_{\text{head}} = d_{\text{kv}} = 64$
- Cannot completely match Encoder only models
    - so constructed model is "similar" but not identical for some models

**Training**

Maximum Likelihood: Teacher forcing, Cross-Entropy Loss

Maximum input sequence length = 512

Tokens per batch = Max input sequence length *batch size = $2^{9}* 2^{7} = 2^{16} = 65,536$

Number of pre-training steps = $2^{19} = 524,288$

Number of tokens in pre-training = Tokens per batch *Number of pre-training steps = $2^{16}* 2^{19} = 2^{35}$

Number of fine tuning steps = $2^{18}$

**Vocabulary**

- SentencePiece with 32K pieces

**Unsupervised Training Objective**

Masked Language Modeling (MLM)

# Results

Paper: reflections (https://arxiv.org/pdf/1910.10683.pdf#page=41)

## Text to Text

Comparable performance to "native" model

## Architecture

Encoder/Decoder

Parameter sharing (between Layers): didn't hurt

Reducing number of layers hurts

## Unsupervised objective

MLM

- Little difference in masking schemes
- MLM corruption rate: unimportant, up to 50%

# Datasets

Subsets of C4 limited to domain-specific examples

- performs better for a *few* downstream tasks
- **conditional** on the narrower datasets not being too small
    - which results in repeated examples when training with a minimum number of tokens constraint

## Training strategy

- Updating *all* parameters during Fine Tuning worked best
- Including Multi Task Learning was **not** beneficial
    - i.e., could not find a mixing strategy that outperformed Unsupervised Pre-Training

In [2]:
```
print("Done")
```

```
Done
```