Task agnostic models

Early approaches to NLP via Deep Learning created task-specific architectures.

The power of these models was enhanced by increasingly sophisticated word representations

- Obtained via a Language Model

As Language Models have grown increasingly powerful (and large)

- The realization is that the architecture for the Language Model is univeral !
    - No need to augment the Language Model with a *deep* task-specific "Head"
- Just use Transfer Learning on the Language Model !

This approach is called *Supervised Pre-training + Fine-Tuning*

- *Supervised Pre-Training*: the Language Model (e.g., predict the next word)
- *Fine-Tuning*: add a task specific head and fine-tune

Contrast this to Word Embeddings, which also use Transfer Learning

- Embeddings transfer *word-level* concepts
- Transferring entire Language Models transfer *semantic* concepts

Because the Pre-Trained model has a very specific input format (and output)

- You often have to encode your task-specific input to fit

For example:

- Consider a Pre-Trained model that performs text completion (predict the next)
- Turn your task into a text completion problem
- [See (https://arxiv.org/pdf/2005.14165.pdf)](https://arxiv.org/pdf/2005.14165.pdf) Appendix G (pages 75+) for examples

Task: Unscramble the letters

Context:	Please unscramble the letters in the word and write that word
	skicts =
Target completion:	sticks

Task: English to French

Context:	English: Please unscramble the letters in the word and write that word
	French:
Target completion:	Veuillez déchiffrer les lettres du mot et écrire ce mot

Sometimes the task encodings are not completely obvious (see [GPT Section 3.3 (https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf)](https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf))

- Task: Are two sentences similar ?
    - Issue
        - There is no natural ordering of the two sentences
        - So concatenating the two (with a delimiter) is misleading
    - Solution
        - Obtain two representations of the sentence pair, once for each ordering
        - Add them together element-wise
        - Feed sum into Classifier

- Task: multiple choice questions answering: given context, question plus list of possible answers
    - Solution:
        - Obtain representation for each answer
            - Concatenate (with delimiter): context, questions, answer
        - Feed each representation into a $\mathrm{softmax}$ to obtain probability distribution over answers

GPT: Task encoding

Picture from: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Universal model: adapting task-specific inputs

Picture from: http://jalammar.github.io/images/bert-tasks.png"

From a very practical standpoint

- In the near future (maybe even now) you will not create a new model
- You will use an existing Language Model
    - Trained with lots of data
    - At great cost
- And fine-tune to your task

Models using Supervised Pre-training + Fine-Tuning

We present a few models using this approach.

GPT: Generalized Pre-Training

GPT is a sequence of increasingly powerful (and big) models of similar architecture.

- The Decoder side of a Transformer Encoder-Decoder model
  - Masked Self-attention
  - Left to Right, unidirectional

Each generation

- Increase the number of Transformer blocks
- Increases the size of the training data

All models use

- Byte Pair Encoding
- Initial encode words with word embeddings

They are all trained on a Language Model objective.

GPT: architecture

Picture from: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

The models can be described as

$$
\begin{aligned}
h_0 &= UW_e + W_p \\
h_i &= \text{transformer\_block}(h_i) \quad \text{for } 1 \leq i \leq n \\
\text{prod}(U) &= \text{softmax}(h_n W_e^T)
\end{aligned}
$$

where

$$
\begin{array}{ll}
U & \text{context of size } k : [u_{-k}, \ldots, u_{-1}] \\
h_i & \text{Output of transformer block } i \\
n & \text{number of transformer blocks/layers} \\
W_e & \text{token embedding matrix} \\
W_p & \text{position encoding matrix}
\end{array}
$$

Let's understand this

- h_0, the output of the input layer
  - Uses word embeddings W_e on the input U
  - Adds *positional* encoding W_p to the tokens
- There are layers h_i of Transformer blocks $1 \leq i \leq n$
- The output $\text{prod}(U)$
  - Takes the final layer output h_n
  - Reverses the embedding W_e^T to get back to original tokens
  - Uses a softmax to get a probability distribution over the tokens U
    - Distribution over the predicted next token

The training objective is to maximize log likelihood on $\mathcal{U}$

$$\mathcal{L}_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

[paper (https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf)](https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf)

[Summary (https://openai.com/blog/language-unsupervised/)](https://openai.com/blog/language-unsupervised/)

- 12 Transformer blocks (37 layers)
    - $n_{\text{heads}} = 12, d_{\text{head}} = 64$
        - $d_{\text{model}} = n_{\text{heads}} * d_{\text{head}} = 768$
        - d_{model} is size of bottle-neck layer
- 117 million weights
- Trained on
    - 5GB of text (BooksCorpus dataset consisting of 7,000 books)
    - Sequence of 512 tokens
    - Training time
        - 30 days on 8 GPUs
        - 26 petaflop-days

The original Unsupervised Training to create the Language Model.

This is followed by Fine Tuning on a smaller task-specific training set $\mathcal{C}$

This can be described as:

- Add linear output layer W_y to the model used for Language Modeling:
- h_l^m is output of transformer block l on input of length m
- Using Θ from unsupervised pre-training
- Fine Tuning Objective:
    - maximize log likelihood on $\mathcal{C}$
      $$\mathcal{L}_2(\mathcal{C}) = \sum_{(x,y)} P(y|x_1, \ldots, x_m) = \mathrm{softmax}(h_l^m W_y)$$

The authors also experimented with a Fine Tuning Objective that included the Langauge Model

$$\mathcal{L}_3(\mathcal{C}) = \mathcal{L}_2(\mathcal{C}) + \lambda \mathcal{L}_1(\mathcal{C})$$

Results of Supervised Pre-Training + Fine-Tuning

- Tested on 12 tasks
- Improved state-of-the-art results on 9 out of the 12

BERT

[paper (https://arxiv.org/pdf/1810.04805.pdf)](https://arxiv.org/pdf/1810.04805.pdf)

BERT (Bidirectional Encoder Representations from Transformers) is also a *fine-tuning* (universal model) approach, like GPT

- does not use *masked attention* to force causal ordering
- uses a Masked Language Model pre-training objective

The Transformer in OpenAI's GPT uses *Masked* Self-Attention

- the Language Models/training objectives are conditioned on *prefix* and *suffix*, not full context
- So is fundamentally a left-to-right Language Model

Masked Language Model task

- Mask (obscure) 15% of the input tokens, chosen at random
- The method for masking takes one of three forms
    - 80% of the time, hide it: replace with $[\text{MASK}]$ token
    - 10% of the time: replace it with a random word
    - 10% of the time: don't obscure it

The training objective is to predict the masked word

The authors explain

- Since encoder does not know which words have been masked
- Or which of the masked words were random replacements
- It must maintain a context for **all** tokens

They also state that, since random replacement only occurs 1.5% of the time (10% * 15%), this does not seem to destroy language understanding

BERT in action

[Interactive model for MLM (https://huggingface.co/bert-base-uncased?text=Washington+is+the+%5BMASK%5D+of+the+US)](https://huggingface.co/bert-base-uncased?text=Washington+is+the+%5BMASK%5D+of+the+US)