Advanced Natural Language Processing

1 LLMs

BERT

Bidirectional Encoder Representations from Transformers.
Basically a transformer encoder.
Training: BERT was trained on Books and Wikipedia.
Training objectives: Masked Language Modelling. Predict randomly masked tokens (by [MASK] token). Next Sentence Prediction (with [CLS] token).
Versions:
- BERT-Base: 12 transformer blocks, embedding dim: 768, 12 attention heads.
- BERT-Large: 24 transformer blocks, embedding dim: 1024, 16 attention heads.
Extensions:
- RoBERTa: Larger batches, removed NSP.
- SentenceBERT: BERT + siamese architecture (Sentence A and B in the same model, outputs are compared by some metric) + triplet loss (tune network such that distance between anchor sentence and positive sentence is smaller than anchor sentence and negative sentence)
- DistillBERT: Larger teacher network creates soft probability distribution labels. Smaller student model tries to replicate these distributions.

GPT

Generative Pretrained Transformer.
Only a transformer decoder, since its task is not seq2seq, where you might want to refer to another part of the input sentence, but instead just generate next tokens (still the decoder receives already generated tokens as inputs).
Goal is to use few-shot learning and prompting instead of fine-tuning and task specific architectures.
Autoregressive: Feeds back generated tokens in decoder input.

Other LLMs

T5

Text-to-Text-Transfer-Transformer.

Encoder and decoder block, similar to original Attention is all you need paper but with slight modifications.

RETRO

Retrieval-Enhanced Transformer.
Enhanced by large text database.
In total 4% of GPT-3 size.

Flash Attention

Self-attention is the bottleneck when training/inferencing transformers (quadratic time). During Attention, tensors are moved from HBM (Hidh Bandwith (GPU) Memory) to SRAM (Shared random-access (GPU) memory) multiple times: Before each operation, the tensors are moved from HBM to SRAM and then written back to HBM.
Flash Attention fuses multiple operations into one kernel, thus saving load and write operations. It is becoming the standard for transformers.
Approximate a complex model with a transparent one

Fine-Tuning

Traditional fine-tuning: Re-train all parameters with a small labelled dataset for example on domain knowledge.
Traditional fine-tuning with freezing: Freeze a part of the model.

Prompting

Removes the need for additional layers for a downstream task. Uses the same format as the pretraining objective of LLMs: We ask the LLM to generate the answer based on a task-specific input.
Requires no new parameter, can be practiced on a closed-source LLM.

Zero-Shot

Simple prompting is equivalent to zero-shot learning. The model predicts the answer based on the question.
Example: "Translate English to French: cheese = ".

In-Context Learning and Few-Shot Learning

Learning that contains examples of solutions to the task.
Example: "Translate English to French: sea otter = loutre de mer. cheese = ".

Prompting Terminology

Pattern:
A function that maps input to text (i.e. template for x).
Example: $f(x) =$ "$Review : x$"
Verbalizer:
A function that maps a label to text (i.e. template for y).
Example: $v(y) =$ "$Sentiment : y$"
Zero-shot prompting is using one pattern. Few-shot prompting is using many pairs of patterns and verbalizers and one final pattern.
Choosing a prompt is important and non-trivial, since experiments show different patterns and verbalizers exhibit large variance on the result accurarcy. Additionaly, it is uncertain how and why in-context learning works.
Reasons:
- Different input and output space distributions decrease/increase performance.
- LLM sees demonstrations not as ordered pairs.
- Highly dependent on choice, order and term frequency.
Prompting still works really well in practice and can perform better than smaller task-specific fine tuned models such as BERT.pretrained
Prompt-based fine tuning: [CLS] A _ master class. It was [MASK].

Parameter Efficient Fine-Tuning

Prompt Search

Learns tokens in the prompt.
Instead of fine-tuning and predicting a [CLS] token for an input sentence, use the masked prompt as input sentence.
Classical fine-tuning: [CLS] A three-hour cinema master class.
Prompt search: [CLS] A _ master class. It was [MASK].

AutoPrompt

Prompt search method.

Update tokens in the pattern using gradient-guided search.
Prompt Template: {Original sentence}[T][T][T][T][T][MASK], where T are trigger Tokens that are determined using gradient-guided search.
Example: a real joy. atmosphere alot dialoge Clone totally [MASK].

Prompt Tuning

Attaches a learnable embedding to the input pattern.
Example: [EMBED] [CLS] A _ master class. [MASK].
Goal: Learn the [EMBED] token.

BitFit

Only tunes bias in attention and linear layers.
Hypothesis: LLM does not need to learn new linguistic features, just domain knowledge.

Adapters

In the Transformer block, after each feed-forward block, add an adapter block and a skip connection.
The adapter layer is built up like an autoencoder (down- and the up-projects the input). Then, only fine tune these adapter blocks.

LoRA

Low-Rank Adaption of LLMs.
At each layer, substitute the weight update on W with an update on the low-rank decomposition of W (AB).

$$W_{finetuned} = W_{pretrained} + \Delta W = W_{pretrained} + AB$$

Low rank means the matrix has much less params the than W.

(IA)3

Infused Adapter by Inhibiting and Amplifying Inner Activations.
Element-wise rescaling of model activations with a learned vector.
Separate learned vectors for each task.

Human Preference Tuning

Fine tuning is not enough to make a model fully non-harmful/friendly and performant/helpful at the same time.
Training on human preferences (rankings, scores, etc.) is not differentiable, since no mathematically differentiable function was used to calculate the scores. Therefore, no supervised learning is possible.
Instead use Reinforcement Learning.

Reinforcement Learning with Human Feedback

Create a smaller LM, the reward model, which is trained on labelled data (4 answers to a prompt ranked from best to worst).
The reward model then calculates a reward for each output of the LLM. This reward is used with the PPO algorithm to update the LLM weights.
To not overfit and let the LLM just "please" the reward model, we calculate a second loss from the LLM output and the old, frozen LLM output. This loss is added to the reward model output before being used in the PPO algorithm.
Observations:
- Smaller models get slightly worse, bigger models get even better.
- Compatible with specialized models (e.g. model for coding)
- Requires much more human annotation than fine tuning. Possible solution: Use LLMs themselves to indentify bad answers (Constitutional AI/RLAIF)

RLAIF

Reinforcement Learning with AI Feedback.
Uses a constitution (Constitutional AI) instead of humans.
1. Make the model revise harmful answers according to a random principle in the constitution. Then use fine tuning with the revised answers.
2. Use an off-the-shelf LLM to rate the answers of the fine-tuned model.
3. Train a reward model on these pairs.
4. Train the final model with reinforcement learning.

DPO

Use Direct Policy Optimization instead of reinforcement learning.
Reformulates the problem into a single cross entropy loss.
So far better than PPO, but the debate is open.

2 Embeddings and XAI

Embeddings

Static Embeddings

Word2Vec, GloVe, FastText.
One embedding per word, the same for every context.

Contextual Embeddings

ELMo, BERT, GPT.
One embedding per word, different for every context.
Produced by a transformer encoder.

Subword Embeddings

Byte-Pair Encoding (BPE)

Merge characters into tokens until desired vocabulary size is reached.

Analyzing Contextual Embeddings

Goal: Find information encoded in the embedding such as about language, the world or harmfulness.
Techniques:
- Dimensionality reduction
- Clustering
- Probing classifier: Train a classifier to evaluate specific properties such as POS, sentiment or NER.
- Association Tests: Use cosine similarit to measure association between two concepts and an attribute A and B.
- POLAR: Subtract two polar opposite embeddings, then muliply with the target word embedding to receive the score of that embedding in the polar opposite embedding space.
Disadvantages: requirement of opposites and dictionaries, requirement of quality context examples (potential source of bias).

Topic Modelling

Identifying abstract themes in a collection of documents.
Old methods:

- Latent Dirichlet Allocation: Topics are characterized as a distribution over words.
- Variational autoencoders: Compression of documents to a topic distribution.
BERTopic:
Modular approach to topic modeling. Composed of six steps, each is interchangable:
1. Generate embeddings (SBERT, spaCy, other transformers)
2. Dimensionality Reduction (PCA, UMAP, TruncatedSVD)
3. Clustering (HDBSCAN, k-means, BIRCH)
4. Tokenizer (CountVectorizer, Jieba, POS)
5. Weighting scheme (c-tf-idf. c-tf-idf + BM25, c-tf-idf + normalization)
6. (optional) Representation Tuning (ChatGPT, Llama, MMR)

2.1 Explainability and Problem Solving

Limitations:
- Multi-step reasoning is hard for LLMs, they generally greedily produce the next token
- Hallucinations: LLM data may be incorrect or outdated, LLM output always looks highly plausible.
Explainable AI (XAI) methods:
- Transparent models: Use inherently transparent models like linear regression or decision trees.
- Model-specific analysis: Fine grained analysis of small neural networks. I.e. choose a neuron in the network and watch its activation for multiple inputs.
- Model-agnostic analysis: Find patterns between model input and output
- Prompting techniques: Make the LLM thought process more transparent (Chain-of-thought)

Model-Agnostic Analysis

Surrogates: Train a simpler, smaller, transparent model that mimics the big model
Feature Attribution: Add an importance score to each feature and measure the result.
Approaches: Remove or replace tokens in a prompt, measure the gradient of output logits.
SHAP:
- Task: Predict income of a person using a set of features
- Question: What was the impact of a certain feature on the prediction?
- Step 1: Determine all combinations of features

- Step 2: Train a model for each combination and measure impact of features(s) on prediction
- Step 3: Calculate marginal contribution of each feature to the prediction
- Step 4: Interpretation

Chain-of-Thought Promting:

Similar to few-shot learning, where an example with the thought procedure in the answer is given. Leads to better and more explainable results, but some tasks are still too difficult for Chain-of-Thought.

Tree-of-Thought:

4 Components:

- Thought Decomposition: Decompose thought into smaller, simpler thoughts
- Though Generator: Generate candidates for the next thought step
- State Evaluator: Evaluate progress made towards problem solution
- Search Algorithm: Searching for high-value thoughts

We can now expore different continuations and evaluate them. This combines pricinples such as Planning, Lookahead and Backtracking.

Probabilistic Tree-of-Thoughts: Assign probabilities to answers.

3 types of question answering:

- Open-Book: Look for answers in the web or DB
- Closed-Book: LLM generates answer
- Child-aggregated: Reason about answer by looking at the child answers.

Difference from regular Tree-of-Thought: The thought decomposition into a tree is done by the LLM instead of the user. The corrections also can be enhanced by choosing from the different question answering strategies based on the assigned probabilities to the answers.

Big Problem in XAI:

Dilemma between plausability to humans and faithfulness to the model. With our strategies we instead can let the model explain its own reasoning, which kind of solves the dilemma.