Large Language Model

What is Large Language Model

↳ A language Model estimates the probability of a token occurring within a longer sequence of tokens

In language Model, the atomic unit that the model is training on. Here a token is

→ a word eg – "dogs like cats"
There are three tokens
dogs, likes, cats

→ a character for example
Eg: Bike fish → consists of nine token
Space is also consists of one token

→ Subwords
↳ In which single word can be single token or multiple tokens

eg → unwatched → Three 3 subwords
un + watch + ed
Prefix + root + suffix

eg → Cats → cat + s
root + suffix

Note: Token is atomic unit or smallest unit of language Model.
Here tokens are also been applied to computer vision & audio generation

Eg:
When I hear rain on my roof, I ..... in my kitchen

Probability	Tokens
9.4%	cook soup
5.2%	warm up a kettle
3.6%	cower
2.5%	nap
2.2%	relax

In some situations it can be whole essay paragraph and sentence or can be called as sequence of tokens to complete the blank as shown in example model will be pick up the token having greater percentage of probability or some Threshold

Hence filling the blanks can go to more higher complexity of tasks like
① Text generation
② Translating text from one language to another language
③ Summarizing documents.

N-gram Language Models

↳ N-grams are ordered sequence used to build the language models. Here N → refers to No of words in the sequence.

when N is 2 then model is called bigram
if N is 5 then it called as 5-gram

below are given in training documents.

4 $\xrightarrow{2}$ 6.

⇒ you are very nice.

4 $\xrightarrow{3}$ 2

2-gram
① you are
② are very
③ very nice.

⇒ 3-gram
① you are very
② are very nice.

Context

↳ Human can retain very long context, while watching Act 3 play you can recollect or retain knowledge of act 1 like the punch line of long conversation.

In language model context is helpful information in front of token or after the target token

Eg: Context can help model to understand if orange is citrus fruit or color

Context can help language model to make better predictions so does 3-gram does that

So eg: Orange is are two words the 3-gram can get seeing many probabilities as answer.

→ Here 3-gram model predicts lots of wrong answers. ou does not mistakes.

Longer N-grams can provide better content than short one. However N-grams gives thier relative ourname of each instance decreases. So when N becomes very large, then model has only a single instance of each ourneme of N-tokens

Recurrent Neural Networks

L> Recurrent neural networks provides more content than N-grams

So - Recurrent Neural Network is a neural network trained on sequence of tokens.

Eg: recurrent neural network can gradually learn selected content from each word in a sentence, kind of you are listening to someone speak. Large recurrent neural network can give content from passage of several sentences.

* Though recurrent large neural network can learn more content than N-grams, but amount of useful content can retrieve is still relatively limited

Recurrent neural network evaluate information token by token

⇒ Training recurrent neural network for long contents is constrained by the vanishing gradient problem.

LLMs → Large Language Models

L> It predicts a token one sequence of tokens, sometimes many paragraphs worth of predicted tokens.

LLMs make much better predictions than N-grams or

recurrent neural networks better. because

① LLMs contain far more parameters than recurrent models

② LLMs gather far more context

Transformers

This is state of art architecture for a wide variety of language model applications, such as translations

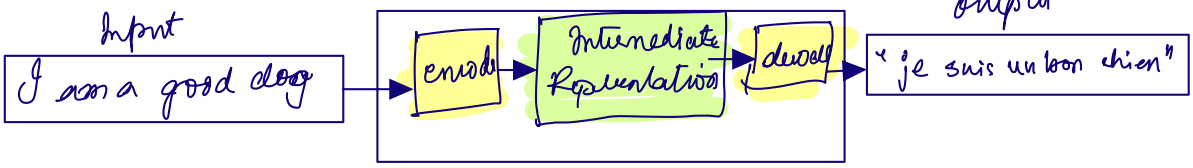
Input
"I am a good dog" → (Transformer based Translate) → "je suis un bon chien"
English
French

Transformers

① encoder → converts the text into an intermediate representation. It is enormous neural net

② decoder → Converts that intermediate representation into useful text. A decoder is also enormous neural net.

Eg: encoder Process the input to intermediate representation the english



* Partial Transformers

① L→ Encoder only architectures } this also exists

② L→ decoder only architectures

① Encoder Only architecture maps input text into an intermediate representation often an embedding layer

Use cases

⇒ predicting any token in the input sequence (which is conventional role of language model)

⇒ Creating a sophisticated embedding which could serve as input to the another system, such as classifier

* Decoder only architecture generates new token from the text already generated this model typically generate sequences decoder only model can use their generation power to create continous dialogues history and other prompts.

Self Attention

To enhance context

↳ Transformers rely on Self attention. Effectively on behalf of each token of input, self attention ask the following questions

" How much does each other token of input affect the interpretation of This token "

⇒ The self in self-attention refers to the input sequence

Self attention weights only the importance relationship between the sequence tokens in the input sequence.

Eg: " The animal dint cross the road because it was too tired "

→ Eleven words.

→ Each word is paying attention to other 10 words; wondering how much each each those ten words matters to itself.

⇒ Eg. the pronoun "it" since pronouns always refer to recent noun but in the example.

it refer ⟨ street / animal

* Self attention mechanisms determine the relevance of each nearby word to the pronoun. it.

Since animal is more important than street.

" The animal dint cross the road it was too tired "

* Here self attention would give more attention to start

* Some self attention mechanism are bidirectional
i.e they collect the content from the both sides.
and do generate representations of whole sequence.

encoders use bidirectional self attention
decoders use unidirectional self attention

* Multihead self-attention

↳ Each self attention layer consist of multiple self attention head.

Output layer is a mathematical operation
like weighted average or dot product.

⇒ Self attention are assigned by random values
different heads can learn different relationships
between each word attended to nearby words.

* Self attention layer's to learn relavence of every word in
Content. so it is $O(N^2)$

N → number of tokens in the content

⇒ Here consist of multiple heads per self attention layer
and multiple self attention layers.

so $O(N^2 \cdot L \cdot H)$ → no of heads per layer.
↳ self attention layers.

LLMs
↳ variously known as.

① Foundational LLM's

① base LLM's

② pre-trained LLM's

→ foundational are well trained on grammer, words and idioms. it can generate help full sentences about topics it is trained on foundational LLM can creative task like writing poetry.

⇒ This is not solution of common ML problems such as regression or classification.

⇒ It can treated as platform.

* Fine tuning is process where foundational LLMs can be prepared for solution for application. Secondary process called distillation with lesser parameter.

Fine tuning

additional training called fine tuning is process of training Model to predict on specified tasks

⇒ Fine tuning trains the model with purpose of your application does for.

⇒ Despite it is small but requires high computational power.

Fine tuning involve below tasks.

① updating weights & bias of every parameter on each back-propagation

② Smarter process called parameter-efficient tuning can fine tune an LLM by adjusting only a subset of parameters on each back propagation.

③ Fine tuned models are better than foundational LLMs prediction

Distillations

↳ It creates smaller version of LLMs, the distilled LLM generate predictions much faster than and requires fewer computational & environmental power.

models predictions are not good as foundation LLMs.

more parameter generate better predictions

Prompt Engineering

prompt engineering enables an LLMs end user to customize the models output. i.e end users clarify how the LLM's should respond to their prompt.

LLMs learns from example.

Showing one example to LLM is called

* One-shot prompting ↗
* Sometimes you have to give multiple examples called few-shot-prompting.

Offline Inference

No of parameters in a LLM could be so large that online inference is too slow to be practical, real world tasks like recursion or classification

* Here offline Inference is preferred. or called as batch inference or static Inference.

1 2 3 + 5 6