Generative Language Model

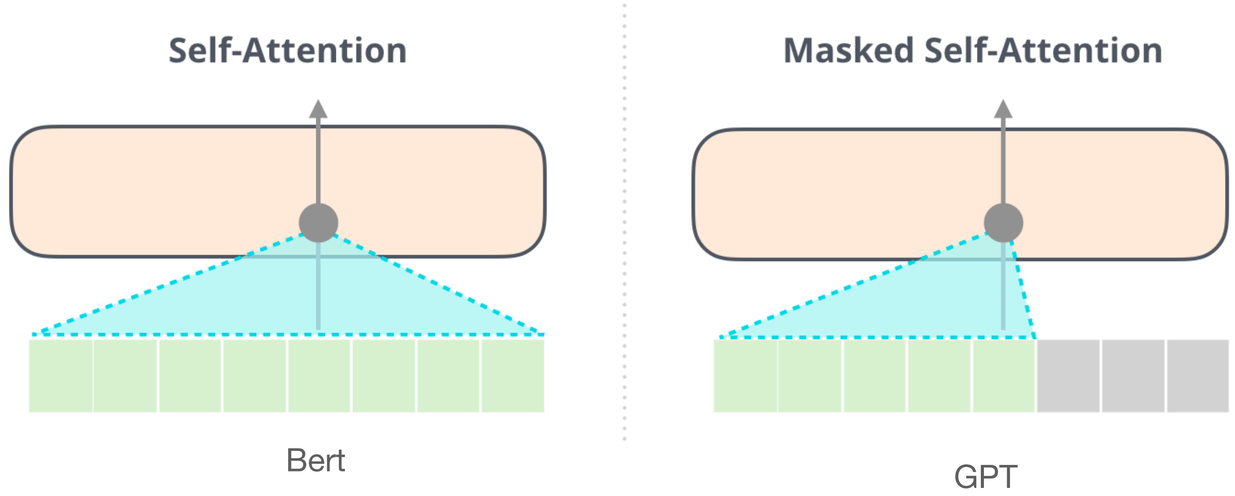
Training Components

- Training
  - Data
    - Just Text
  - Model Architecture (Transformer / LSTM ….)
  - Loss
    - Cross-Entropy loss (default)
    - Next-Token Prediction (Auto-regressive prediction)

Type of transformer

- Encoder only (BERT)
- Encoder & Decoder (T5)
- Decoder only (GPT-2 / 3)

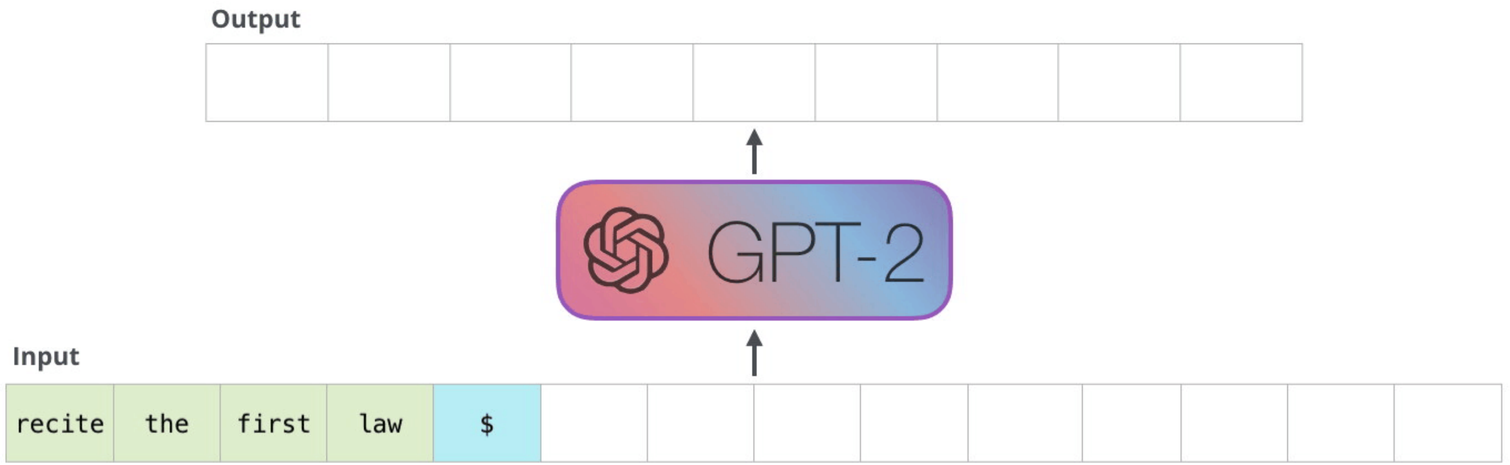
Difference of architecture



- Encoder
  - Can attend bi-directional token
  - Cannot predict next token (because it is already attend to next token)

- Decoder
  - Can attend only left-side token
  - Auto-regressive ability (predict next token, only one at a time)

Decoder only

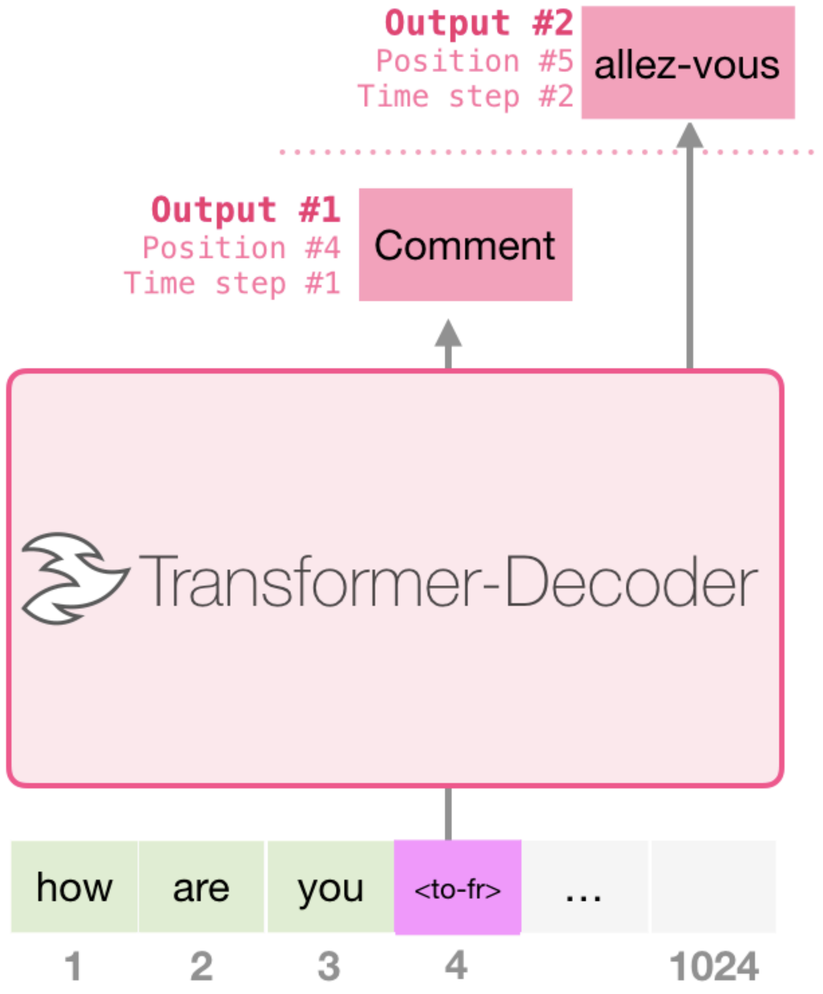


Beyond next token prediction

Conditional Generation (Translation)

Training Dataset

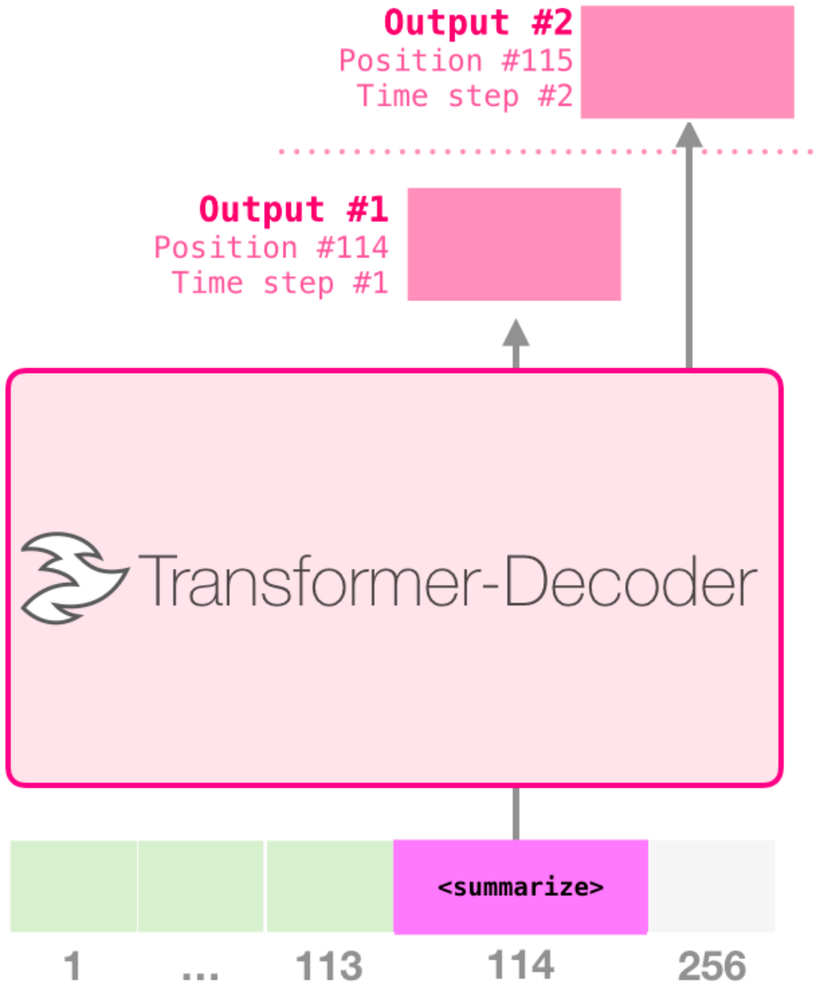
I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



Conditional Generation (Summarize)

Training Dataset

Article #1 tokens		<summarize>	Article #1 Summary
Article #2 tokens	<summarize>	Article #2 Summary	padding
Article #3 tokens		<summarize>	Article #3 Summary



What is important on Training

- Hyperparameter
  - Batch size (gradient accumulation * batch size)
  - Learning rate
  - number of training step
- Implementation
  - Padding or not / if padding do you calculate loss on pad token?
  - BOS / EOS token is consistance to pretrained ?
  - Always print generate sentence (sampling sentence) because you may cannot tell that you implementation is correct or not based on only loss
  - If the training set is large enough, you can ignore to do validation set split, because training set is a validation set because you are training only for 1 epoch)
  - Most implementation bug / problem should be detect in first 100 - 500 step
  - Implementation detail & hyperparameter are also important but you can mostly stick with huggingface trainer default param
- Data
- Sampling algorithm

What to know about data?

- More clue to a model = better performance
- Input English & Thai together, the loss will be lower than put only one
- Similar pattern to pretrained data, the better the performance
- Human liked sentence > set of word

What to know about data?

- Balance between hard & easy data is important
  - If task is too easy, model will not learn anything, because it got low loss, without learning
    - If input & output is the same, except one token, the model will always copy token, without learning anything
  - If task is too hard, the model will overfit because it need to remember the thing that is not related.
    - We want to learn about paraphrase, but if input have difference content from output, it need to remember how to create content.

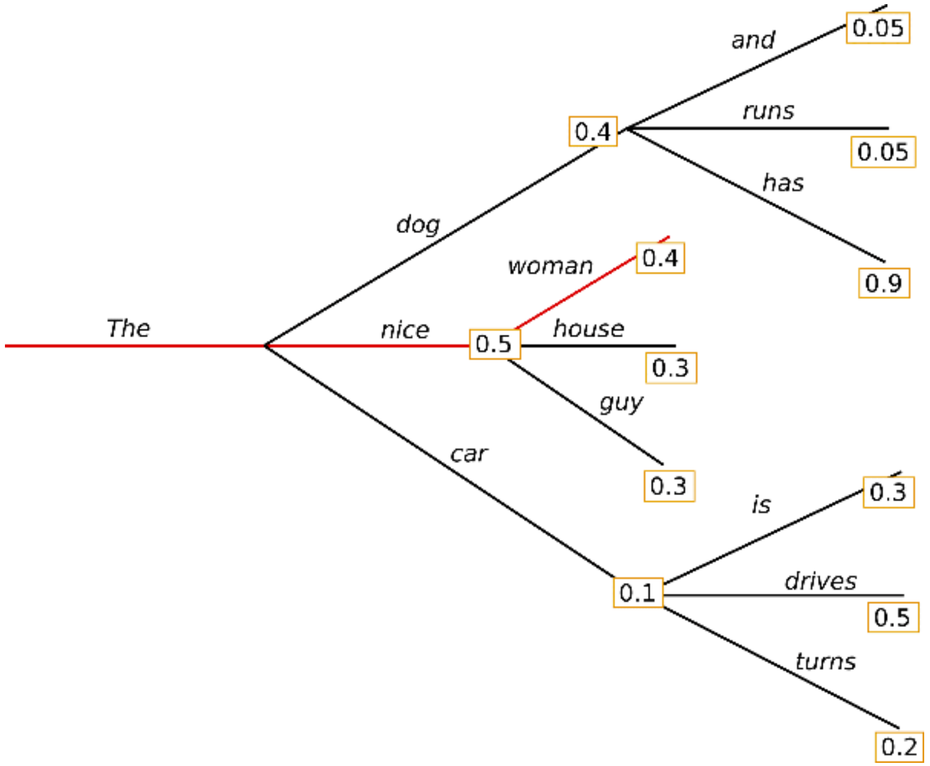
Sampling algorithm

- Greedy
- Beam-Search
- Sample

- https://huggingface.co/blog/how-to-generate
- https://docs.cohere.ai/docs/controlling-generation-with-top-k-top-p

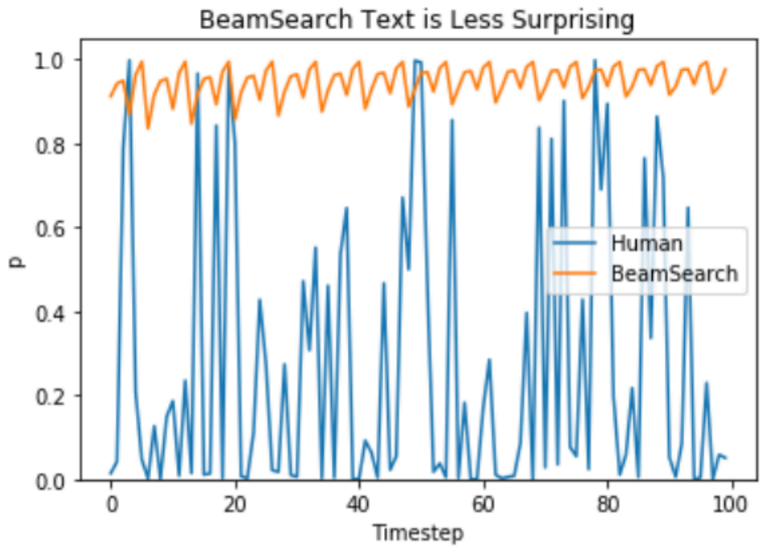
Greedy

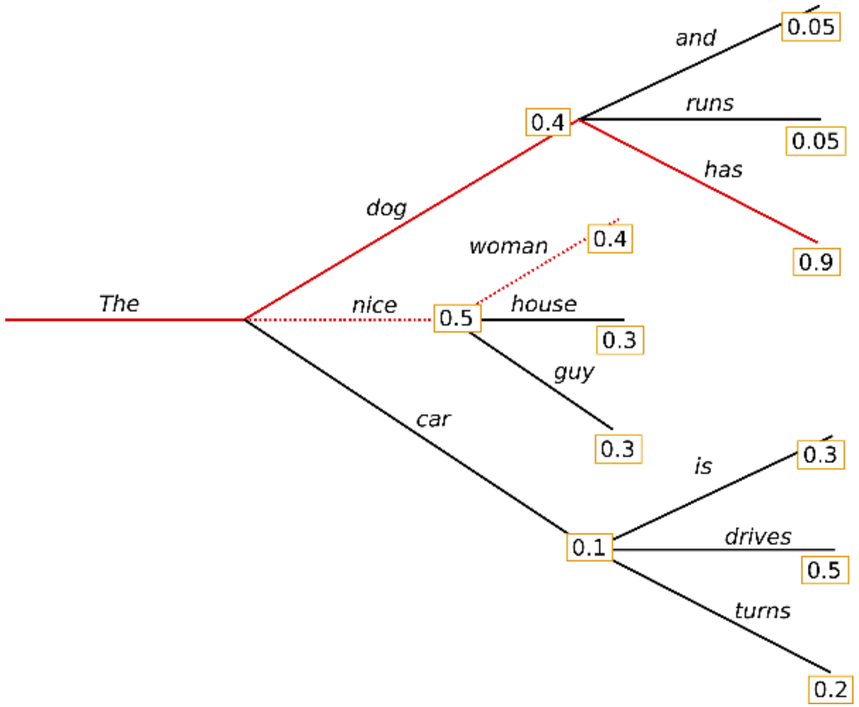
- Simple



Beam-search (mostly used on task that need high accuracy)

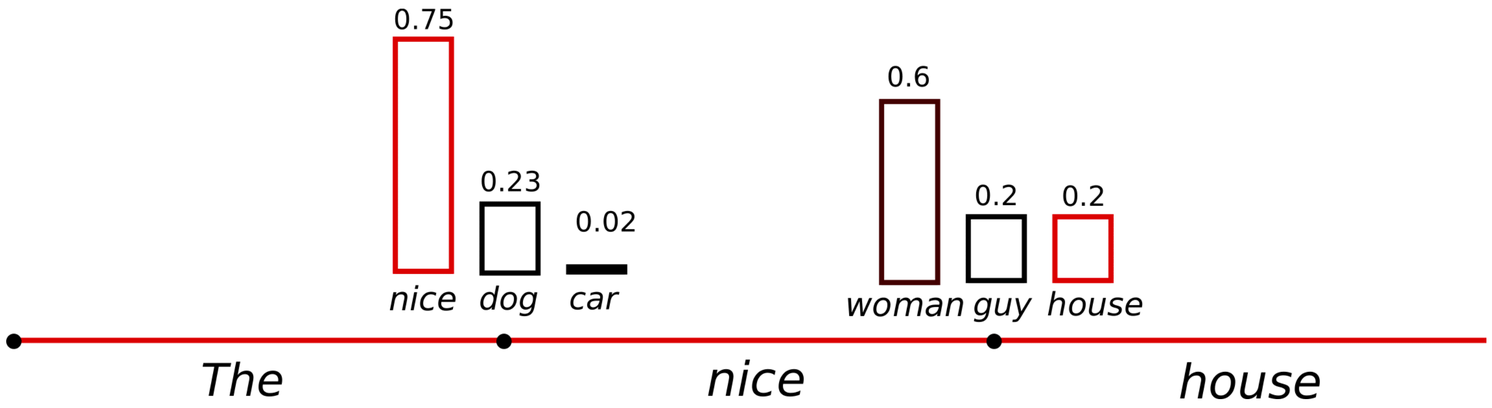
- Important params
  - num_beam





Sampling (mostly used on task that need creativity)

- Important params
  - temperature
  - top_p
  - top_k



Additional method

- Contrastive search (sampling token using its own prob and prob of that token is still contrast with previous token)
  - Can combine with sampling & greedy & beam?
  - Best decoding method
  - Hyper param
    - top_k
    - penalty_alpha

Experiment

- Check input & output data
- Select Architecture (Decoder only / Encoder & Decoder)
- Make sure the implementation is correct (it can learn, generation look make sense)
- Find a setting that lead to biggest batch size, then can be run for ~ 1 day (because colab has max running duration = 24 hours)
- LR tuning (may start with default LR ~1e-5)
    - Running for around ~1000 step
    - Try increase LR by [* 5] (5e-5) if loss is lower than first run, try increase the (power) but change the multiplier to 1 (1e-4)
    - If the loss is higher, do another way (try (5e-6))
    - Try in range of (1e-4, 1e-6) (it should be in this range)
- Set the sampling method (greedy, beam, sampling) (in this case I think we should use greedy or sample ?
- Train for ….. step (try with 1 epoch when there are lot of example)
- Check the result
- May be change the data pattern (if it is similar you can ignore the LR test process and use same LR)