Large Language Models Under the Hood

Part 2: LLM training and inference

David Samuel

Language Technology Group, University of Oslo

Table of contents

1. Language models
2. Generative transformer-based language models
3. Prompt-based learning
4. Autoregressive generation
    - Searching for the most probable sequence
    - Random sampling
5. References

Language models

Language models are statistical models of language – they estimate the probabilistic distribution of language utterances

Language models are statistical models of language – they estimate the probabilistic distribution of language utterances

Example

- Corpus: A A B C D A B B A C
- Simplest LM: $P(A) = 40\%$, $P(B) = 30\%$, $P(C) = 20\%$, $P(D) = 10\%$

Language models are statistical models of language – they estimate the probabilistic distribution of language utterances

Example

- Corpus: A A B C D A B B A C
- **Simplest LM:** $P(A) = 40\%$, $P(B) = 30\%$, $P(C) = 20\%$, $P(D) = 10\%$
- **More complex:** $P(C|A) = 25\%$, $P(C|AB) = 50\%$

Language models are statistical models of language – they estimate the probabilistic distribution of language utterances

Example

- **Corpus:** A A B C D A B B A C
- **Simplest LM:** $P(A) = 40\%$, $P(B) = 30\%$, $P(C) = 20\%$, $P(D) = 10\%$
- **More complex:** $P(C|A) = 25\%$, $P(C|AB) = 50\%$

Generally

- $P(w_0 w_1 \dots w_n) = \prod_{i=0}^{n} P(w_i | w_0 w_1 \dots w_{i-1})$
- For example: $P(ABC) = P(A)P(B|A)P(C|AB)$

And that's it!

The problem

- Sadly we can't do that without infinite compute
- $P(w_n|w_0 w_1 \ldots w_{n-1})$ requires $|V|^n$ entries – scales exponentially

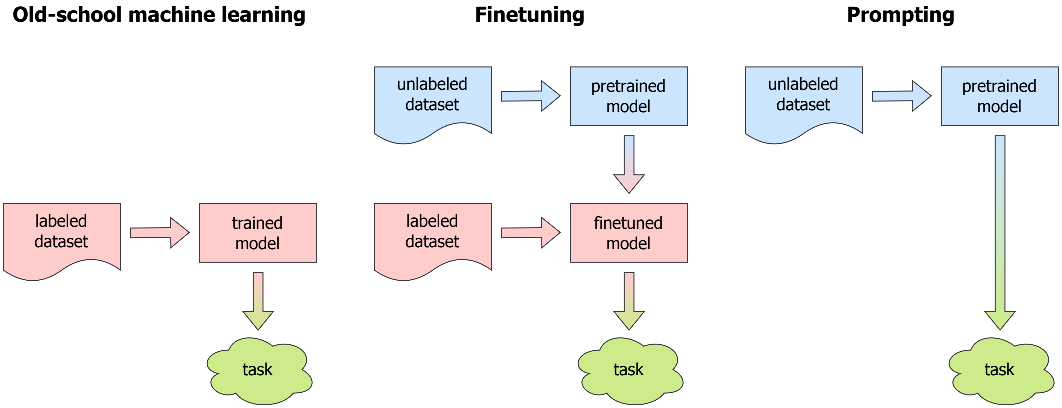
The problem

- Sadly we can't do that without infinite compute
- $P(w_n|w_0 w_1 \ldots w_{n-1})$ requires $|V|^n$ entries – scales exponentially

The solution

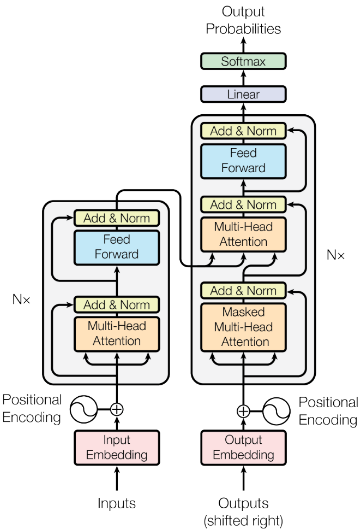
- Deep learning!
- We can use deep neural networks to approximate the intractable probability tables $P(w_n|w_0 w_1 \ldots w_{n-1})$
- The neural networks can use the patterns and structure within the language for an accurate approximation with a relatively small amount of parameters

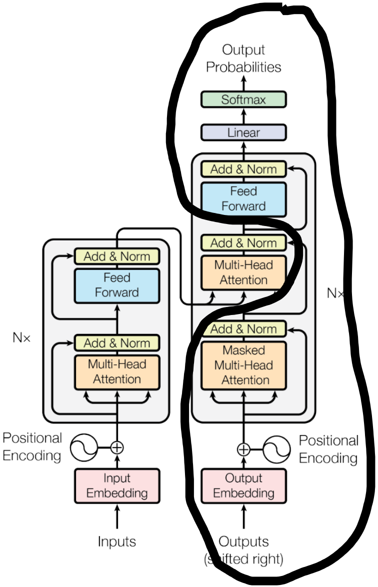
How is that useful?



1. Language models
2. Generative transformer-based language models
3. Prompt-based learning
4. Autoregressive generation
    - Searching for the most probable sequence
    - Random sampling
5. References

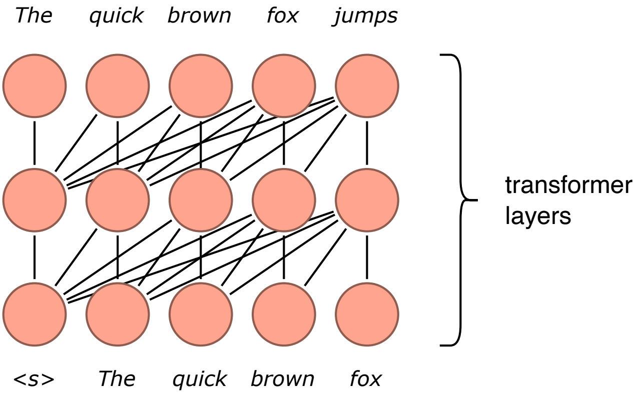
Generative transformer-based language models




Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

- Training objective: next-word prediction
- Alternative names: causal language models (CLMs), decoder-only LMs, autoregressive LMs, left-to-right LMs, GPT-like LMs or simply “language models”
- We want to estimate the probability distribution of sequences of words (or tokens in general)

Causal language model

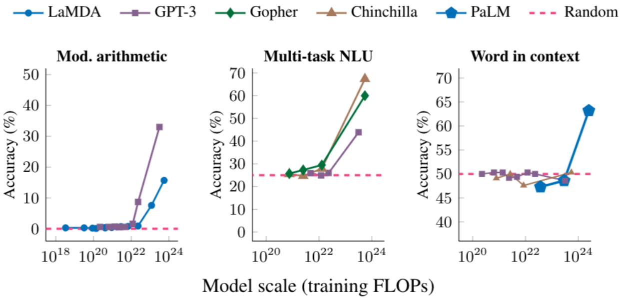


Quick history

- A giant leap forwards in history: Bengio et al. (2001) proposed *"A Neural Probabilistic Language Model"*
- Another huge jump forward – the same thing, but now with transformers: *"Improving Language Understanding by Generative Pre-Training"* by Radford et al. (2018) – the first GPT
  - The idea here is to train a transformer on CLM and then fine-tune the whole thing on a downstream task
- Same architecture in GPT-2, just bigger (Radford et al., 2019)
  - Now the model starts to solve* downstream tasks without any fine-tuning!
    - * better than random prediction

In-context learning

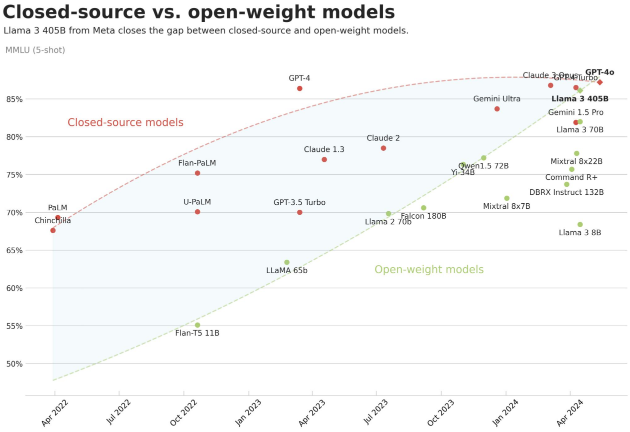
- The same thing repeated with GPT-3 (Brown et al., 2020)
  - When the size of the model grows even more (0.1B $\rightarrow$ 1.5B $\rightarrow$ 175B), the model starts to be really good in zero-shot and few-shot
  - So-called “emergent abilities” (Wei et al., 2022) [slightly controversial term nowadays]



In-context learning

- A double-edged sword for decoder language models
- Their main advantage, otherwise they perform poorly in the pre-training / fine-tuning paradigm
- Therefore they have to be huge and expensive to be somewhat useful
- In practice: is it cheaper to annotate a dataset or to infer from a huge decoder LM?
    - A small fine-tuned LM usually performs better than a huge non-fine-tuned decoder LM (Bang et al., 2023)

A lot of media attention in this field so big-tech invests a lot to train these huge models



https://www.linkedin.com/posts/maxime-labonne_

GPT and its open friends

- The scientific contributions of the closed-souecw models are questionable
  - the papers are usually not peer-reviewed or even don't exist at all (ChatGPT)
  - the models are not released and it's thus impossible to reproduce and verify the results
  - we don't know what these models were trained on
- Still, there are some initiatives and research labs that try to change that
  - GPT-J (Wang and Komatsuzaki, 2021)
  - OPT (Zhang et al., 2022)
  - LLaMA, Mistral, Qwen, OLMo...and many more

Table of contents

1. Language models
2. Generative transformer-based language models
3. Prompt-based learning
4. Autoregressive generation
    - Searching for the most probable sequence
    - Random sampling
5. References

Formal definition

- **Supervised learning:** we have a dataset of input-output pairs and we learn to assign the correct output y to any input x
  - We directly optimize $P(y|x)$
- **But:** We don't always have a large enough annotated dataset!
- **Then:** Let's use a language model that learns $P(T)$ of any text T and reformulate the task so that we can approximate $P(y|x)$ by language modeling
  - We will use a prompt template f that can transform any input-output pair x, y into natural text: $f(x, y)$

$$argmax_y \; P(y|x) \approx argmax_y \; P(f(x, y))$$

Example (filled prompt)

- $x =$ *"I really enjoyed this movie."*
- $y =$ *"positive"*, $y' =$ *"negative"*

Example (filled prompt)

- $x =$ *"I really enjoyed this movie."*
- $y =$ *"positive"*, $y' =$ *"negative"*

- $f =$ "[X] *To sum up, it was a* ['good' if $Y ==$ 'positive' else 'bad'] *movie.*"

Example (filled prompt)

- $x =$ *"I really enjoyed this movie."*
- $y =$ *"positive"*, $y' =$ *"negative"*

- $f =$ "[X] *To sum up, it was a* ['good' if $Y ==$ 'positive' else 'bad'] *movie.*"

- $f(x, y) =$ *"I really enjoyed this movie. To sum up, it was a good movie."*
- $f(x, y') =$ *"I really enjoyed this movie. To sum up, it was a bad movie."*

Example (filled prompt)

- $x =$ *"I really enjoyed this movie."*
- $y =$ *"positive"*, $y' =$ *"negative"*

- $f =$ "[X] *To sum up, it was a* ['good' if $Y ==$ 'positive' else 'bad'] *movie."*

- $f(x, y) =$ *"I really enjoyed this movie. To sum up, it was a good movie."*
- $f(x, y') =$ *"I really enjoyed this movie. To sum up, it was a bad movie."*

- Predict *"positive"* iff:

$$P(f(x, y)) > P(f(x, y'))$$

Cloze and prefix prompts

- This was an example of a filled prompt, which contains both the input x and answer y
- But some tasks have to be generated for efficiency (translation, summarization) and some APIs don't allow for probability estimation

Cloze and prefix prompts

- This was an example of a filled prompt, which contains both the input x and answer y
- But some tasks have to be generated for efficiency (translation, summarization) and some APIs don't allow for probability estimation

- Here, the prompt template f only transforms x and the missing y is generated by a language model

Example (prefix prompt)

- $x =$ *"lang og svingete vei"*
- $y =$ any possible English translation

Example (prefix prompt)

- $x =$ *"lang og svingete vei"*
- $y =$ any possible English translation

- $f =$ *"The Norwegian sentence '[X]' can be translated as:*

Example (prefix prompt)

- $x =$ *“lang og svingete vei”*
- $y =$ any possible English translation

- $f =$ *“The Norwegian sentence ‘[X]’ can be translated as:*

- $f(x) =$ *“The Norwegian sentence ‘lang og svingete vei’ can be translated as:*

Example (prefix prompt)

- $x =$ *"lang og svingete vei"*
- $y =$ any possible English translation
- $f =$ *"The Norwegian sentence '[X]' can be translated as:*
- $f(x) =$ *"The Norwegian sentence 'lang og svingete vei' can be translated as:*
- Now we have to generate text to create y

Example (prefix prompt)

- $x =$ *"lang og svingete vei"*
- $y =$ any possible English translation
- $f =$ *"The Norwegian sentence '[X]' can be translated as:*
- $f(x) =$ *"The Norwegian sentence 'lang og svingete vei' can be translated as:*
- Now we have to generate text to create y
  - *"The Norwegian sentence 'lang og svingete vei' can be translated as: long and winding road"*
- In the case of machine translation, the generated text can be directly used as y

Zero-shot and few-shot learning

- zero-shot == no supervised data
- few-shot == ”few” supervised samples of data
  - For example, “8-shot” means that we show a model 8 gold input-output pairs (so-called “shots”)
- Usually, no training is involved, we don’t change the weights as in fine-tuning
- The gold input-output pairs are a part of a prompt, it shows what is the expected behavior

Zero-shot and few-shot learning

- zero-shot == no supervised data
- few-shot == ”few” supervised samples of data
  - For example, “8-shot” means that we show a model 8 gold input-output pairs (so-called “shots”)
- Usually, no training is involved, we don’t change the weights as in fine-tuning
- The gold input-output pairs are a part of a prompt, it shows what is the expected behavior

- Much more efficient than fine-tuning, simpler, no overfitting, no need for large annotated datasets

Zero-shot and few-shot learning

- zero-shot == no supervised data
- few-shot == ”few” supervised samples of data
  - For example, “8-shot” means that we show a model 8 gold input-output pairs (so-called “shots”)
- Usually, no training is involved, we don’t change the weights as in fine-tuning
- The gold input-output pairs are a part of a prompt, it shows what is the expected behavior
- Much more efficient than fine-tuning, simpler, no overfitting, no need for large annotated datasets
- But we need a large (and good-enough) LM and the performance will be mediocre compared to fine-tuning on a sufficiently large dataset

Example (3-shot prefix prompt)

- x = *"ahh, what a waste of time :("*
- y = *"positive"*, y' = *"negative"*

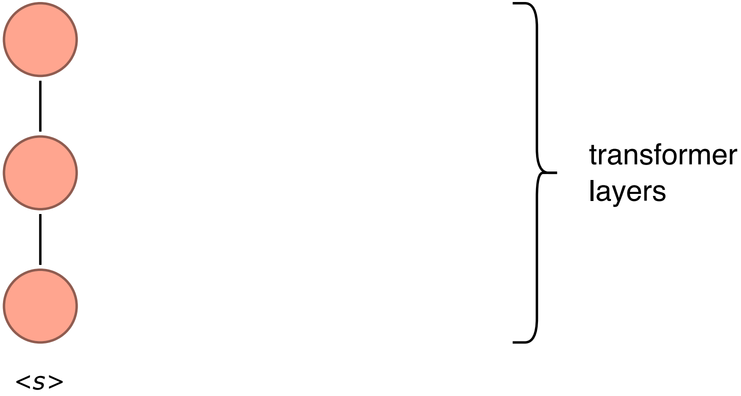
- f = *"I felt asleep in the middle. -> negative; I really enjoyed this movie. -> positive; What an amazing ending! -> positive;* [X] *->"*

- $f(x)$ = *"I felt asleep in the middle. -> negative; I really enjoyed this movie. -> positive; What an amazing ending! -> positive;* ahh, what a waste of time :(*->"*

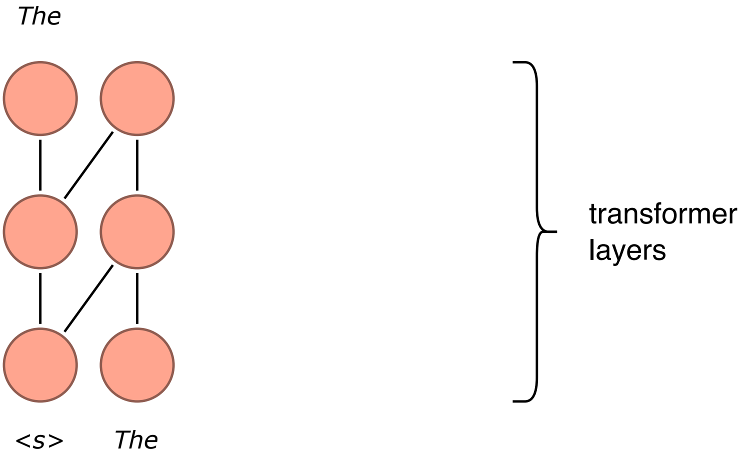
Table of contents

1. Language models
2. Generative transformer-based language models
3. Prompt-based learning
4. Autoregressive generation
    - Searching for the most probable sequence
    - Random sampling
5. References

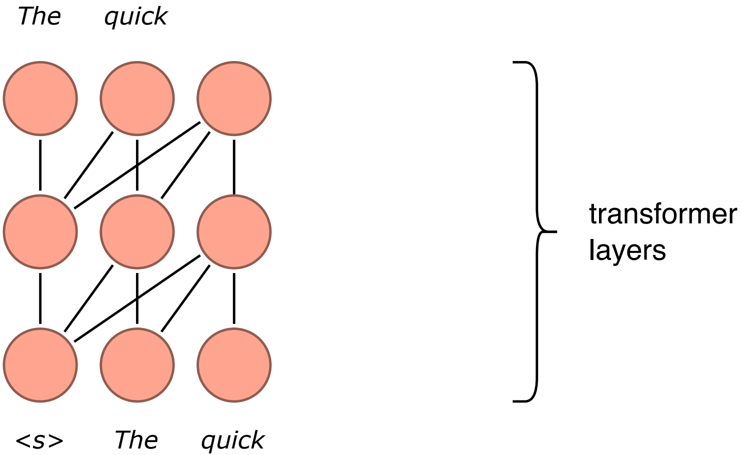
Autoregressive text generation



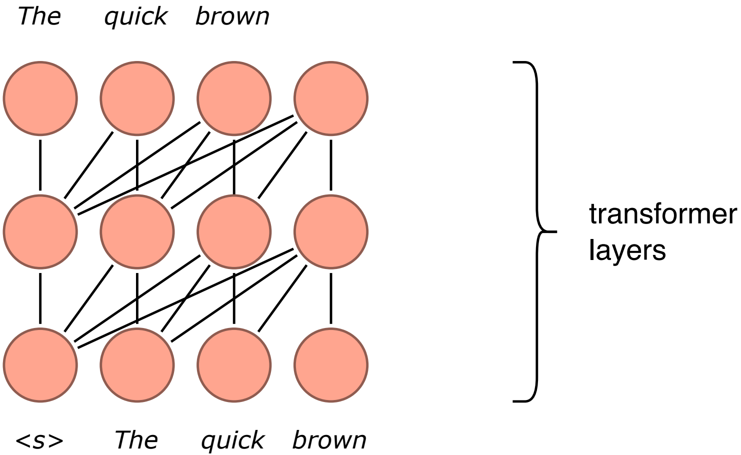
Autoregressive text generation



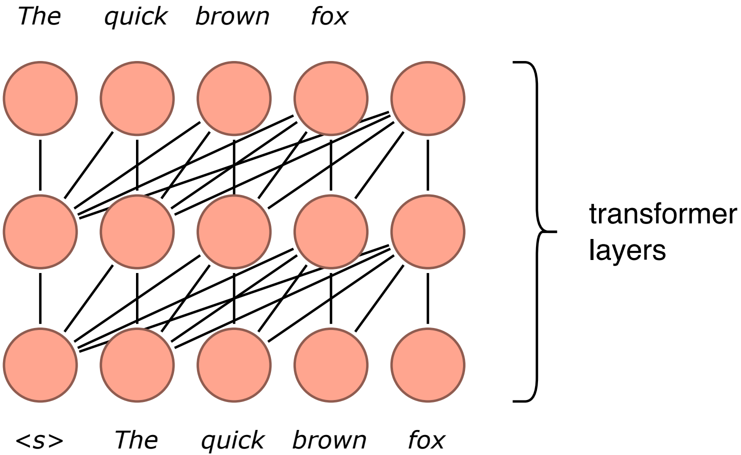
Autoregressive text generation



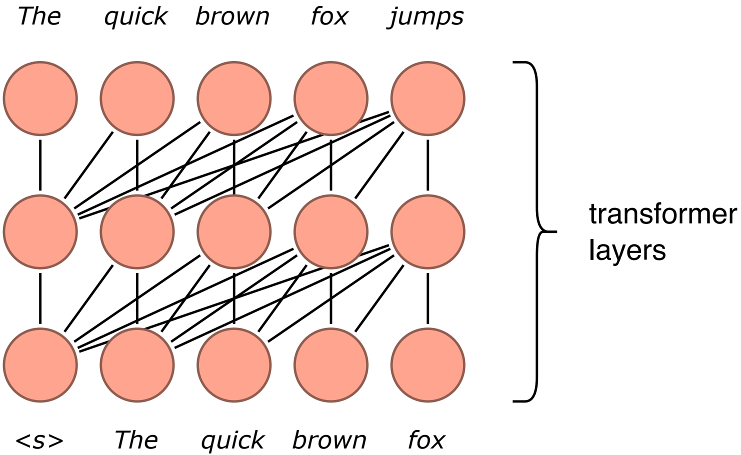
Autoregressive text generation



Autoregressive text generation



Autoregressive text generation



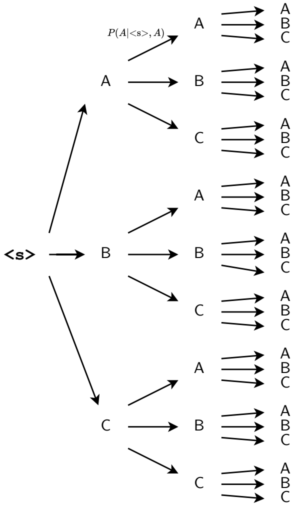
Probability of a token sequence

- Remember that the causal language models are trained to estimate the probability of a sequence of tokens:

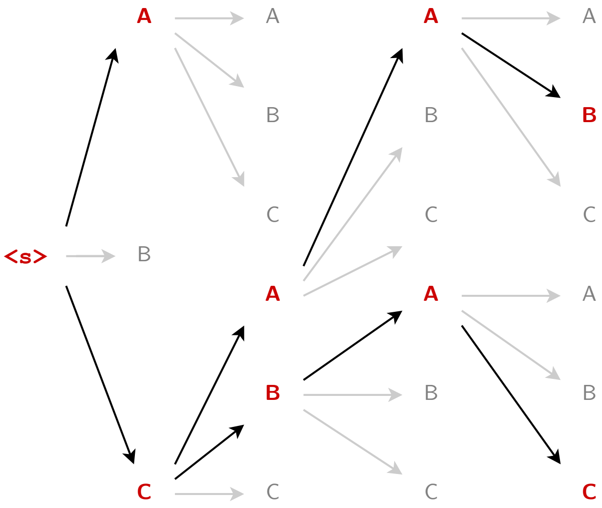
$$P(w_{0:n}) = \prod_{i=0}^{n} P(w_i|w_{0:i-1}) = P(w_0) \cdot P(w_1|w_0) \cdot P(w_2|w_0, w_1) \dots$$

- But how can we output the most probable $w_{0:n}$?
- If we have vocabulary of size $|V|$ and max sequence length t, we have to go through $O(|V|^t)$ sequences!

Naive search in the exponentially large space



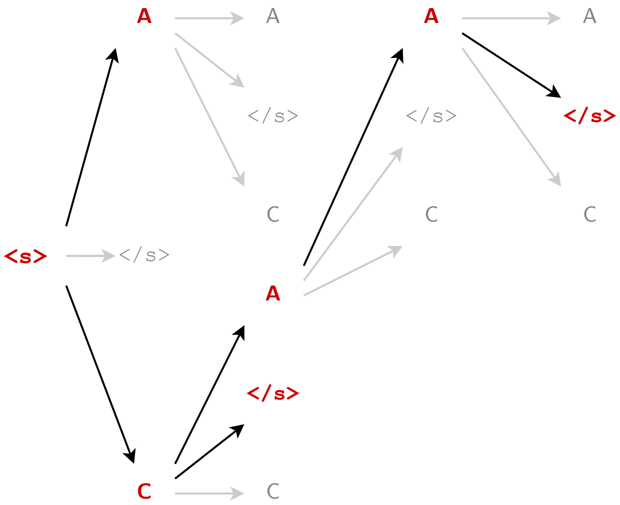
Pruning of the search-space with beam search (beam size 2)



Two candidates:

- <s>CAAB
- <s>CBAC

Beam search with a stop condition



Two candidates:

- <s>CAA</s>
- <s>C</s>

Beam search

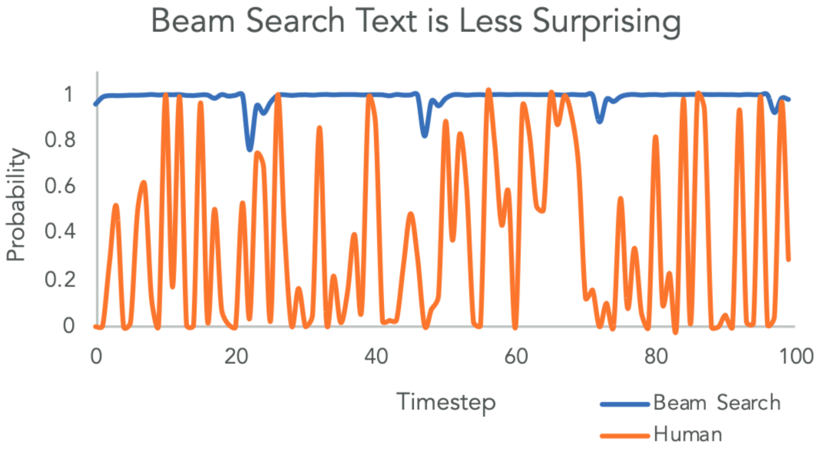
- The most popular search method when we want to estimate the most likely output
- An essential part of machine translation
- We take into account only the k most probably candidates seen so far ("beam size" hyperparameter)
- The complexity drops from $O(|V|^t)$ to $O(k \cdot t)$

Greedy search

- A special case of beam search with beam size of 1
- Simpler to implement than the general beam search and usually substentially faster (no overhead)
- The performance drop is usually not large with a good model

Random sampling

"Natural language does not maximize probability" (Holtzman et al., 2020)

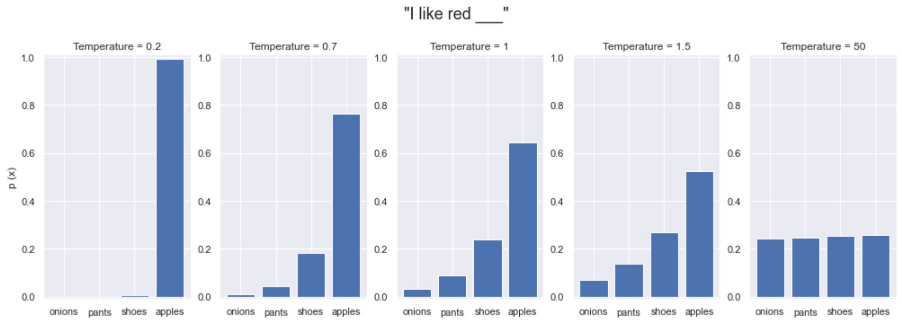


Sampling vs. beam search

- The outputs from beam/greedy search can sound stiff, generic, repetitive or simply boring
- That's exactly what we want from machine translation (most of the time) – to exactly translate the source text without any creative variation
- But it's not what we expect from chatbots, for example
- We want them to get "creative" and use less probable tokens from time to time
- Also, a practical issue of beam/greedy search is that it leads to outputs with infinitely repeated n-grams

Sampling

- Randomly picks a token from the conditional distribution $P(w_i|w_{0:i-1})$
- We can control the amount of "randomness" (entropy) with the temperature hyperparameter (T) $p_i = \frac{e^{x_i T}}{\sum_j e^{x_j T}}$



https://medium.com/@imisri1/how-to-set-sampling-temperature-for-gpt-models-762887df1fac

Sampling

- Randomly pick a token from the conditional distribution $P(w_i|w_{0:i-1})$
- We can control the amount of "randomness" (entropy) with the temperature hyperparameter (T) $p_i = \frac{e^{x_i T}}{\sum_j e^{x_j T}}$
- NB: when $T \rightarrow 0$, we get greedy search; when $T \rightarrow \infty$, we sample uniformly

Table of contents

1. Language models
2. Generative transformer-based language models
3. Prompt-based learning
4. Autoregressive generation
    Searching for the most probable sequence
    Random sampling
5. References

Bang, Y., Cahyawijaya, S., Lee, N., Dai, W., Su, D., Wilie, B., Lovenia, H., Ji, Z., Yu, T., Chung, W., Do, Q. V., Xu, Y., and Fung, P. (2023). A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity.

Bengio, Y., Ducharme, R., and Vincent, P. (2001). A neural probabilistic language model. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

References II

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Wang, B. and Komatsuzaki, A. (2021). GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022). Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. (2022). Opt: Open pre-trained transformer language models.