# IMP. HYPERPARAMS FOR LLMs :

**Top K :** LLMs predict next token/word
Top K limits K top probable tokens considered (candidates)

top K = 3

| 6% | 4% | 3% | 2% | 1% | 0.5% |
|---|---|---|---|---|---|
| Studio | office | Roof | hair | Table | Laptops |

**Top P (Nucleus sampling) :** LLMs predict next token/word
Tokens are selected from most probable to least until the sum of their probabilities or cumulative prob. = top-p value
(candidates)
top-p = 10% or 0.1

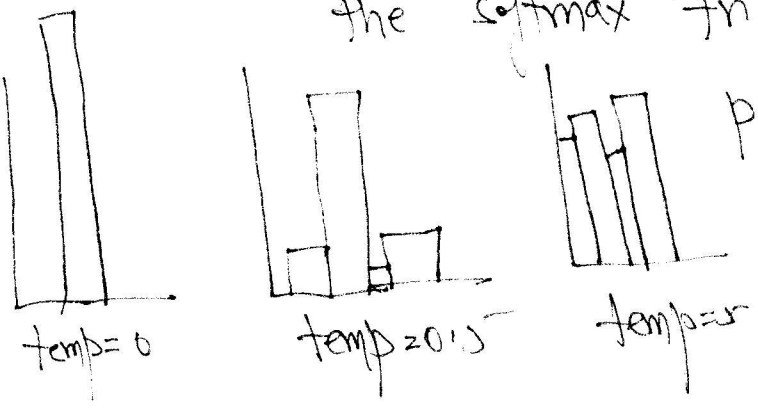
| 6% | 4% |
|---|---|
| Studio | office | Roof | Chair |

**Temp:** Controls the degree of randomness in token selection
e.g. temp = 0 : highest prob. token is selected
Lower value = factual response
Higher value = more creative response

Mathematically the temp. parameter scales the logits (raw scores) of the model's output before applying the softmax fn. Adjusted softmax formula!

$$p_i = \frac{e^{(x_i/T)}}{\sum_{j=1}^{n} e^{(x_j/T)}}$$

where $x_i$ = logits (raw scores)
$T$ = temp.
$p_i$ = prob. of o/p i after applying softmax



## LABEL SMOOTHING:

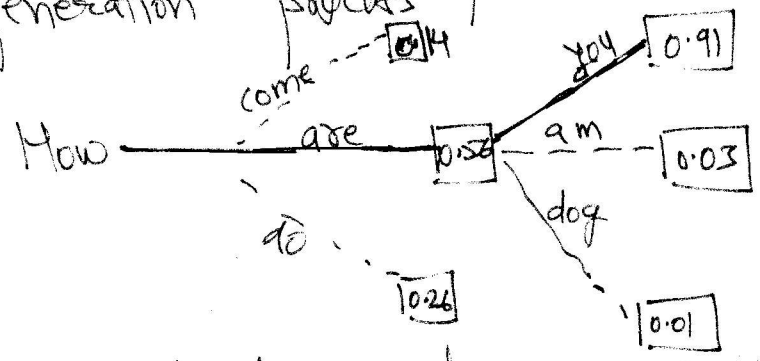
- Introduces noise for the labels e.g. instead of 1 and 0 labels become 0.98 and
- Regularization technique
- Accounts for the fact that the datasets may have mistakes in them so maximizing the likelihood of $P(y|x)$ directly can be harmful
  So, for a small constant $\epsilon$, based on softmax
  label 1 → becomes $1-\epsilon$
  label 0 → becomes $\frac{\epsilon}{K-1}$ where K = no. of o/p class

## How To SELECT TOKENS FROM MODEL'S PREDICTED PROB. DIST.

1. Deterministic < Greedy search, Beam Search
2. Stochastic < Top-p, Top-k
   (In conjunction with temp.)

## GREEDY SEARCH:

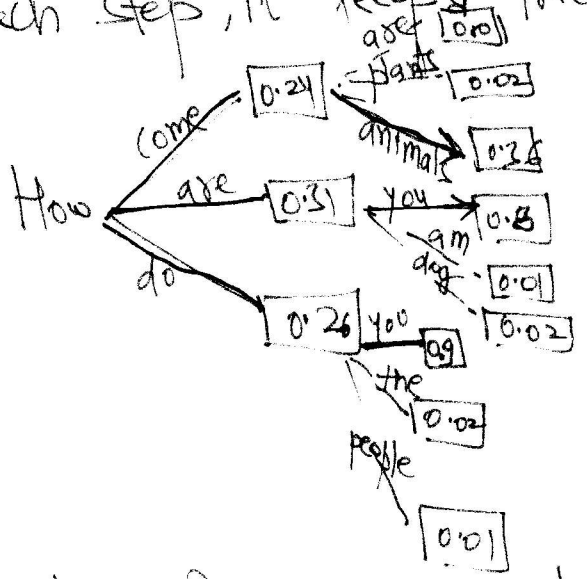
- selects token with highest prob. at each step of the generation process




- Issue: suboptimal generation as it is greedy i.e. it might miss a seq. of high prob. tokens hidden behind a low prob. token

## BEAM SEARCH:

- improvement over greedy search
- considers multiple sequences simultaneously
  At each step, it keeps track of top-K sequences where K = configurable




- Benefits: allows for more exploration and produces higher quality text than greedy
- Issues: Computationally inefficient bcz it requires evaluating multiple seq. at once.

## KV CACHE:

- In autoregressive decoding (generating one token at a time) traditional transformers recompute attention for the entire seq. at each step during inference
- KV cache stores the key & value vectors from prev tokens, avoiding redundant calculation

E.g. i) First token generation:
- Compute Q, K, V vectors for initial toke
- Generate next token using attention

ii) Subsequent tokens:
- only compute Q, K, V for new token
- Retrieve prev K, V pairs from cach
- Compute attention using K, V pairs from + new token (Q, K & V)

## LoRA: (Low Rank Adaptation)

- parameter efficient fine-tuning method
- pre-trained wts → frozen
  injects trainable rank decomposition matrices
- Inference: O/P from pre-trained wts
  +
  O/P from trained rank decomposition mat

MIXTURE OF EXPERTS: (MoE) : Experts + Router

- Expert N/ws
  - Multiple experts
  - Each expert specializes in diff. aspects of input space
  - All experts have the same architecture but learn diff. parameters

- Router / Gating N/w
  - Determines which expert should process each I/p
  - Learnt n/w via training (or routing mechanism)

- Load Balancing
  - Ensures experts are utilized evenly

Biggest Adv: Enables much larger models with similar computational cost (if all experts utilized ↑ computation cost)

Issues:
- comm. overhead b/w experts
- complex infrastructure requirements

# LLM Inference Optimization

- GPU accelerated Hardware + Model parallelism (diff. layers on diff. GPUs)
- Model optimization: — Quantization: use datatypes that take less storage e.g. float16, int8 etc
  — Distillation: smaller model that mimics
  — Pruning: very small wts ×

vLLMs
- KV cache: caches K,V pairs for tokens since they are needed for generating each new token
- Paged Attention & Dynamic batching

without vLLM: One request at a time is processed and all generated text is kept in expensive GPU mem

with vLLM: i) Paged Attention

Instead of keeping everything in expensive GPU memory it can temporarily move less used info to CPU memory and bring it back when needed
makes it possible to handle multiple requests without running out of memory

ii) Dynamic batching

Combines multiple requests into single batch to better utilize GPU parallel processing

- Flash Attention: efficient impl. of Attention
- Speculative decoding: — Small model predicts multiple tokens ahead
  — Larger model verifies all these multiple tokens at once in parallel, either accepts or rej

Eg. The cat sat on the —

i) Smaller model predicts → mat and slept

ii) Large model verifies this in parallel

iii) If 'mat' is right, we saved time by not generating tokens one by one

iv) If wrong, we still spent less compute than running the large model alone