Homework-1
11-664/763: Inference Algorithms for Language Modeling
Fall 2025

Instructors: Graham Neubig, Amanda Bertsch

Teaching Assistants: Clara Na, Vashisth Tiwari, Xinran Zhao

Due: September 25th, 2025

Instructions

Please refer to the collaboration, AI use policy as specified in the course syllabus.

1 Shared Tasks

Throughout the semester, you will be working with data from three shared tasks. We host the data for each shared task on Hugging Face; you can access them at this link. We will generally ask for results on the "dev-test" split, which consists of 100 examples for each task, using the evaluation scripts provided. The remainder of the examples can be used for validation, tuning hyperparameters, or any other experimentation you would like to perform. The final shared task at the end of the semester will be evaluated on a hidden test set.

Algorithmic The task that the language model will tackle is N-best Path Prediction (Top-P Shortest Paths). Given a directed graph $G = (V, E)$ with $|V| = N$ nodes labeled $0, \ldots, N-1$ and non-negative integer edge weights $w : E \to 1, \ldots, W$, the task is to find the top-P distinct simple paths from source $s = 0$ to target $t = N - 1$ minimizing the additive cost

$$c(\pi) = \sum_{(u,v)\in\pi} w(u, v). \tag{1}$$

The output is a pair

$$\texttt{paths} = [\pi_1, \ldots, \pi_P], \quad \texttt{weights} = [c(\pi_1), \ldots, c(\pi_P)], \tag{2}$$

sorted by non-decreasing cost. The language model will be expected to use tool calls[1] to specify its answer.

Evaluation compares predicted pairs $(\pi, c(\pi))$ against the reference set with the score

$$\text{score} = \frac{|(\pi, c(\pi))\text{pred} \cap (\pi, c(\pi))\text{gold}|}{P}. \tag{3}$$

[1]https://platform.openai.com/docs/guides/function-calling

MMLU medicine We will use the two medicine-themed splits of MMLU: college_medicine and professional_medicine. Evaluation is on exact match with the correct multiple-choice answer (e.g. "A").

Infobench Infobench provides open-ended queries with detailed evaluation rubrics. Evaluation **requires calling gpt-5-nano**; we expect that the total cost for evaluation for this homework will be substantially less than \$5. See the paper for more information.

2 Written responses

2.1 How are MLE, CE, Entropy, and KL divergence Connected?

2.1.1 Given two discrete distributions $P(x)$ and $Q(x)$, how do we define $\mathbb{D}_{KL}(P|Q)$, the KL Divergence between the two distributions?

Solution:
The KL divergence given $P(x)$ and $Q(x)$ can be expressed as:

$$D_{KL}(P \,\|\, Q) \;=\; \sum_x P(x)\,\log\frac{P(x)}{Q(x)}.$$

which measures how much information is lost when Q is used to approximate P. □

2.1.2 MLE and KL

In the class, we learned about Maximum Likelihood Estimation (MLE). Now consider a dataset $\mathcal{D} = \{x_1, \dots, x_N\}$ draw IID from an unknown true distribution $p(x)$. Let us define $p_o(x)$ as the the observed/ empirical distribution of the data. We want to fit a parametric model $q(x|\theta)$ to this data.

Show that minimizing the KL divergence between the empirical distribution and the model $D_KL(p_o|q_\theta)$, is equivalent to maximizing the log-likelihood of the data under the model $q(x|\theta)$.

Solution:
The likelihood of the dataset $\mathcal{D} = \{x_1, \dots, x_N\}$ under the model $q(x \mid \theta)$ is

$$L(\theta) \;=\; \prod_{i=1}^{N} q(x_i \mid \theta).$$

Taking the logarithm, the log-likelihood is

$$\ell(\theta) \;=\; \sum_{i=1}^{N} \log q(x_i \mid \theta).$$

Now consider the KL divergence between the empirical distribution $p_o(x)$ and the model $q(x \mid \theta)$:

$$D_{\mathrm{KL}}(p_o \,\|\, q_\theta) = \sum_x p_o(x)\,\log\frac{p_o(x)}{q(x \mid \theta)} = \sum_x p_o(x)\log p_o(x) \;-\; \sum_x p_o(x)\log q(x \mid \theta).$$

Since the first term $\sum_x p_o(x)\log p_o(x)$ does not depend on θ, minimizing $D_{\mathrm{KL}}(p_o \,\|\, q_\theta)$ is equivalent to maximizing

$$\sum_x p_o(x) \log q(x \mid \theta).$$

By the definition of $p_o(x)$,

$$\sum_x p_o(x) \log q(x \mid \theta) = \sum_x \left(\frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{x_i = x\} \right) \log q(x \mid \theta) = \frac{1}{N} \sum_{i=1}^{N} \log q(x_i \mid \theta) \propto \sum_{i=1}^{N} \log q(x_i \mid \theta) = \ell(\theta).$$

Therefore,

$$\arg\min_\theta D_{\mathrm{KL}}(p_o \,\|\, q_\theta) = \arg\max_\theta \ell(\theta),$$

which shows that minimizing the KL divergence is equivalent to Maximum Likelihood Estimation. □

2.1.3 KL and CE and Entropy

Recall that for two discrete distributions $P(x)$ and $Q(x)$, the entropy is defined as $H(P)$, and the cross-entropy as $H(P, Q)$.

Now, you will show that the KL divergence between distributions P and Q is equivalent to the difference between Cross Entropy and Entropy.

Prove that

$$\mathbb{D}_{KL}(P||Q) = H(P, Q) - H(P)$$

Interpretation: KL divergence measures the expected number of extra bits are needed because we are using not the true distribution (Q) instead of the true distribution (P).

Solution:
By definition for discrete KL divergence distribution,

$$D_{\mathrm{KL}}(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} = \sum_x P(x) \log P(x) - \sum_x P(x) \log Q(x).$$

Since the entropy and cross-entropy are defined as

$$H(P) = -\sum_x P(x) \log P(x), \qquad H(P, Q) = -\sum_x P(x) \log Q(x).$$

Therefore,

$$D_{\mathrm{KL}}(P\|Q) = -H(P) + H(P, Q) = H(P, Q) - H(P).$$

□

2.2 Gumbel and Softmax

We looked at the Gumbel-Max Trick briefly in class.

Note that for $X \sim \text{Gumbel}(\mu, \beta)$

$$f_X(x) = \frac{1}{\beta} e^{-(z + e^{-z})} \quad \text{where } z = \frac{x - \mu}{\beta}, \qquad F_X(x) = e^{-e^{-(x-\mu)/\beta}}.$$

Now, assume we have independent random variables $X_i \sim \text{Gumbel}(\mu_i, 1)$, where $i \in \{1, 2\}$.

What is the probability that $X_1 = \max(X_1, X_2)$? In other words, what is $P[X_1 > X_2]$?

Hint:

- What is the probability of X_2<x? Recall how the CDF of a random variable is defined:

$$CDF_{X_2}(x) = F_{X_2}(x) = P[X_2 < x].$$

  This is for a specific value of x! Can you extend this by integrating over the distribution of X_1?

- **Relevance.** This shows that taking argmax of scores perturbed by Gumbel noise is equivalent to sampling with probabilities given by the softmax of the original scores.

<u>**Solution:**</u>
We want to compute

$$\mathbb{P}[X_1 > X_2] \;=\; \int_{-\infty}^{\infty} \mathbb{P}[X_2 < x]\, f_{X_1}(x)\, dx \;=\; \int_{-\infty}^{\infty} F_{X_2}(x)\, f_{X_1}(x)\, dx.$$

For $X_i \sim \text{Gumbel}(\mu_i, 1)$, we have

$$F_{X_2}(x) = \exp\big(-e^{-(x-\mu_2)}\big), \qquad f_{X_1}(x) = \exp\big(-(x-\mu_1)\big)\, \exp\big(-e^{-(x-\mu_1)}\big).$$

Thus

$$\mathbb{P}[X_1 > X_2] = \int_{-\infty}^{\infty} \exp\big(-e^{-(x-\mu_2)}\big)\, \exp\big(-(x-\mu_1)\big)\, \exp\big(-e^{-(x-\mu_1)}\big)\, dx.$$

Let $t = e^{-(x-\mu_1)}$ so that $dx = -\frac{dt}{t}$ and, noting $e^{-(x-\mu_2)} = e^{-(x-\mu_1)}e^{-(\mu_1-\mu_2)} = t\, e^{-(\mu_1-\mu_2)}$, define $\Delta = \mu_1 - \mu_2$. Then

$$\mathbb{P}[X_1 > X_2] = \int_{\infty}^{0} \exp\big(-te^{-\Delta}\big)\, t\, \exp(-t)\, \Big(-\frac{dt}{t}\Big) = \int_{0}^{\infty} \exp\big(-(1+e^{-\Delta})t\big)\, dt.$$

Evaluating the integral,

$$\mathbb{P}[X_1 > X_2] = \frac{1}{1+e^{-\Delta}} = \frac{1}{1+e^{-(\mu_1-\mu_2)}} = \frac{e^{\mu_1}}{e^{\mu_1}+e^{\mu_2}}.$$

Therefore,

$$\mathbb{P}[X_1 > X_2] \;=\; \frac{e^{\mu_1}}{e^{\mu_1}+e^{\mu_2}} \;=\; \text{softmax}(\boldsymbol{\mu})_1$$

□

2.3 How perplexed can you get?

2.3.1 Choose your prompts

Come up with 3 prompts that you would expect to result in a generated sequence with low perplexity, and 3 prompts that you would expect to result in a sequence with high per-token perplexity, assuming greedy sampling until an EOS token or 64 tokens are generated. The prompt itself may be any length within a standard context length. Assume the model is a standard 7-8B autoregressive transformer base language model (dense, not MoE, not instruction tuned). Explain your reasoning behind each hypothesis – you will provide a total of 6 prompts and 6 explanations.

Solution:

Low perplexity prompts

1. **"Water freezes at"** This is a scientific fact that appears frequently in general knowledge sources. The model will confidently continue with 0 degrees Celsius or 32 degrees Fahrenheit. Because the answer is highly predictable, perplexity should be low.
2. **"The capital of China is"** This is a common pattern in Wikipedia. The next word being Beijing is almost a sure thing, so the per-token perplexity ends up really low.
3. **"2 + 3 ="** Arithmetic facts are well-memorized by LLMs. The model is almost certain that the next token is 5, so its prediction is basically deterministic.

High perplexity prompts

1. **"Chicken dance crazily if"** This odd, nonsensical phrase doesnât follow usual patterns. The model has no strong clue what comes next, so predictions are uncertain.
2. **"*@?#"** A random string of symbols is rare in training data. With no consistent pattern, the model guesses randomly, leading to high perplexity.
3. **"Yesterday I dreamed my book turned into"** Dreams can go in countless directions, and the continuation could be anything nonsense. The model has no strong favorite, so perplexity is high.

□

2.3.2 Testing and reflection

Now, test your hypotheses, for each of your six sequences, note: the per-token and global perplexity scores; an observation (even if your prediction of perplexity score was correct, you might e.g. note that the model kept generating more variations of the prompt instead of stopping); and a possible explanation for what you observed, especially if it differs from what you predicted (you do not have to verify these, e.g. "Maybe the model was trained on math textbooks and that's why it kept generating more problems after the equation I prompted it with"). Report the model you used.

Solution:

Category	Prompt	SeqPPL	AvgPerTokenPPL
Low	Water freezes at	2.071	307.620
Low	The capital of China is	2.835	73.467
Low	2 + 3 =	1.582	90.486
High	Chicken dance crazily if	13.291	2223.226
High	*@?#	2.079	392590.218
High	Yesterday I dreamed my book turned into	4.083	199.962

The model I used is meta-llama/Llama-3.1-8B. Overall, the results match my hypotheses. Factual prompts produced low sequence-level perplexities, while creative prompts showed much higher values. For example, $2 + 3 =$ was extremely predictable, yielding the lowest SeqPPL. On the other hand, "Chicken dance crazily if" led to the highest SeqPPL, as the model struggled to confidently continue a nonsensical phrase. An observation is the wild variance in average per-token perplexity. For example, "*@?#" had SeqPPL approximately 2.08, but AvgPerTokenPPL is around 392590, which is exploding due to the modelâs uncertain and repetitive continuation with noisy symbols. Similarly, even simple prompts like "Water freezes at" showed inflated per-token perplexity because the model elaborated with additional facts (Fahrenheit, Celsius, etc.), introducing uncertainty. This highlights the difference between the two measures: global sequence perplexity is more stable and reflects overall fit, while per-token perplexity can be dominated by a few highly improbable tokens. □

Reflect: Would you have chosen different prompts if you had been asked to assume an instruction tuned model? What if the vocab size had been smaller or larger? Overall, did you find it easier or harder to intuit sequence level vs per-token level perplexity scores?

Solution:
If I had used an instruction-tuned model, I would likely have chosen different prompts. Instruction-tuned models tend to stop after concise answers, so prompts like "$2 + 3 =$" would have been cleaner. With a smaller vocabulary, perplexities would generally be lower because probabilities concentrate over fewer tokens.
I found it easier to intuit the relative SeqPPL values than the per-token perplexities, since the per-token perplexities are more sensitive to model continuation behavior. □

2.4 Beam Search Puzzle

Run beam search on `Qwen-3-1.7B` using Hugging Face. Find an example where two of the beams produce identical text. Provide the input and outputs, and explain how this is possible.

Solution:
Input: Ha Ha Ha Ha
Chooseing num_beams = 30, it producrs this identical text: Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha HaHa Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha
Beam search looks at multiple possible continuations in parallel and keeps the ones with the highest scores. But if the model is super confident about one token (like Ha), almost every beam ends up picking the same thing. When the beam size is really big (e.g., 30), it basically just explores a bunch of near-identical options. Since Ha keeps dominating the prediction at every step, different beams collapse back into the same output. □

2.5 One more step into Calibration

In Sep 2 Class, we talked about *calibration* of models: a model is well-calibrated if the confidence score is well-correlated with the probability of correctness [5]. In this question, we will delve deeper into the methods for calculating calibration scores and compare their differences.

Prior work [4] utilizes Expected Calibration Error (`ECE`) to compute the model calibration. `ECE` uses a bucketing approach that measures the *overall calibration*. It assigns examples with similar confidence to the same buckets. Given the input x, ground truth y, prediction $\tilde{y}$, and B_m denoting the m-th bucket for $(x,y,\tilde{y})$, for N model predictions bucketed into M buckets:

$$\mathrm{ECE} = \frac{1}{N} \sum_{m=1}^{M} |B_m| \cdot |\mathrm{Acc}(B_m) - \mathrm{Conf}(B_m)|,$$

where $\mathrm{Acc}(B_m)$ and $\mathrm{Conf}(B_m)$ denote the accuracy and averaged confidence for the samples in B_m, and $|B_m|$ denotes the cardinality of B_m.

More recently, [6] designs Macro-average Calibration Error (`MacroCE`) to evaluate the quality of confidence calibration with different treatments on the correct and wrong predictions. For each split, `MacroCE` accumulates the individual calibration errors in a similar way as the Brier Score [3]. Read these papers and implementations, then answer the following questions.

2.5.1 Methods

In this question, you will be given three cases C_1, C_2, C_3, and your job is to compute their `ECE`, `MacroCE`, and `Brier` scores. For `ECE`, we assume that there are **2** buckets where each contains three examples. For brier score, we assume the confidence scores are predicting whether the predictions are correct (i.e., the actual outcome). Report your results in Table 2.5.1 (rounded to three decimal places).

Each case has 6 examples, where each one is with a confidence score, a prediction, and a true answer, i.e., label, as shown in Table 1.

Cases	Conf. Scores	Predictions	True answers
C_1	$[0.9, 0.9, 0.8, 0.8, 0.7, 0.7]$	$[1, 1, 1, 1, 1, 1]$	$[1, 0, 1, 0, 1, 1]$
C_2	$[0.9, 0.9, 0.9, 0.8, 0.8, 0.7]$	$[1, 1, 1, 1, 1, 1]$	$[1, 1, 1, 0, 0, 0]$
C_3	$[0.9, 0.9, 0.8, 0.8, 0.6, 0.6]$	$[1, 1, 1, 1, 1, 1]$	$[1, 1, 1, 1, 1, 0]$

Table 1: Examples of each case. Conf. scores denote the model confidence scores.

Cases	`ECE`	`MacroCE`	Brier Score
C_1	0.133	0.538	0.280
C_2	0.433	0.433	0.300
C_3	0.067	0.400	0.103

Table 2: Expected Calibration Error (`ECE`), Macro-average Calibration Error (`MacroCE`), and Brier scores of different cases. For all of them, the lower the better.

Solution:

Cases	ECE	MacroCE	Brier Score
C_1	0.133	0.538	0.280
C_2	0.433	0.433	0.300
C_3	0.067	0.400	0.103

□

2.5.2 Details and Comparison.

Temperature (re)scaling. [4, 2] discuss the method and implications of temperature (re)scaling. Briefly describe what temperature scaling is, how you can find a good value for temperature, and give an example of how you would tune temperature for C_2 based on the results you computed above.

Solution:
Temperature (re)scaling rescales logits by a single scalar before applying the softmax. It does not change the predicted class but only adjusts confidence level ($T > 1$ reduces overconfidence while $T < 1$ reduces underconfidence).
To find a good T, we can minimize the negative log-likelihood of the validation set, utilizing grid search for example. Once we choose the best T, we can evaluate metrics ECE, MacroCE, and Brier score again.
C2 is overconfident on its incorrect items, so we expect $T > 1$ to help by reducing the incorrectly high confidences. For example, we can set $T = 2$, 0.8 then becomes

$$\sigma\left(\frac{\log \frac{0.8}{0.1}}{2}\right) = 0.667$$

which is closer to 0 than 0.8.

□

Bucket-canceling effect. The bucketing design of `ECE` can trigger cancellation effects [6], i.e., the over- and under-confident instances within the same bucket may cancel with each other and hence not contribute to the overall error. Use 1 sentence to describe what bucket-canceling effect is and 1-3 sentences to give an example to describe how `MacroCE` can help. Hint: you can reuse C_3 to compare the difference between `ECE` and `MacroCE`.

Solution:
When a bucket mixes over- and under-confident samples, their errors offset in the bucket cancel out so that $|\mathrm{Acc}(B) - \mathrm{Conf}(B)|$ becomes very small.
In C_3, the second bucket has Acc $= 2/3$ and Conf $= 2/3$, so ECE adds 0 from this bucket despite non-zero errors per example. `MacroCE`, which averages $|z_i - c_i|$ over examples, remains sensitive to these individual deviations, yielding a larger overall value than ECE and exposing the masked miscalibration. □

2.5.3 Calibration in long-form answers

In the previous sections, we described different ways to compute the calibration scores; These methods work well for measuring the overall and individual calibration for cases with categorical predictions and labels. However, for many applications of LLMs nowadays, the predictions are often a long-form answer with multiple words, e.g., IFEval [9]. The answers are commonly evaluated in a *LLM-as-a-judge* manner [8].

Given two open-weight models, one generates the answers and another serves as a judge. The LLM judge can give a $0, 1$ score for N criteria (akin to the labels), along with a text-based free-form rationale. Each criterion can either cover a part or the whole of the long-form answer. Your task is to describe your design of a generalized version of the `MacroCE` to measure the confidence calibration for long-form answers with *LLM-as-a-judge*. Use 1 sentence to describe your design, and describe 1 advantage and 1 disadvantage of your design, e.g., what cases can be captured and what can not.

Solution:
Design: Take a weighted average of the absolute differences between the judge is 0/1 decision and the modelâs confidence for each criterion, using weights that reflect its importance.
Advantage: This yields fine-grained calibration that highlights where the model is over-/under-confident on specific aspects, and the weights keep small checks from dominating.
Disadvantage: It depends on reliable confidences and appropriate weights. Poor judgments or weights can be misleading and mess up the interactions between criteria. □

3 Programming

In the programming portion of this homework, you'll implement several decoding strategies yourself. Then, you'll use standard library implementations to compare decoding strategies on the shared tasks.

3.1 Bug hunting

First, a debugging problem to help you avoid some common mistakes :)

There are two major issues with this implementation of diverse beam search [7] with hamming distance diversity scoring at each step and length normalized global sorting at the end. Describe what the issues are, what behaviors the issues would lead to, and submit your fixed implementation. Keep in mind a single issue might propagate and require multiple changes to fix completely.

The broken implementation is provided in `diverse-beam-search-broken.py`. An example call to the function is: `diverse_beam_search(model, tokenizer, prompt, beam_width=6, num_groups=3, max_length=6, device=device, diversity_strength=2.0)`, where the model and tokenizer come from a standard pre-trained Hugging Face model, and the prompt is a string.

Reflections In addition to your implementation submission, describe characteristics of a task that diverse beam search would be useful for, and a task that it would be unhelpful for. In class, it was mentioned that Hamming diversity is both relatively easy to compute and usually effective. Please also provide an example of a task or a prompt(s) that diverse beam search would be generally appropriate for, but that one might expect Hamming diversity scoring to be clearly *insufficient* for. Propose an alternative scoring method that you would hope would outperform Hamming diversity (by some quantitative or qualitative measure). You do not have to empirically test your hypotheses, but please provide explanations for your choices, and describe what settings (or by what alternative measures, e.g. some aspect of efficiency) you might expect your alternative scoring method to *not* be as useful for.

Solution:

Issues There were two major problems. First, although the code defined a `penalized_probs` to apply the Hamming diversity penalty, it mistakenly used the original `probs` for top-k selection. This meant that the diversity penalty was never applied, so all groups often collapsed into the same continuation. Second, the implementation tracked scores by multiplying the raw probabilities and then dividing by the sequence length for normalization. This makes scores shrink exponentially and makes a bias toward shorter outputs. The correct way is to accumulate the log probabilities and then apply length normalization when comparing beams. Together, these issues caused beams all to converge onto a nearly identical sequence and preferred very short sequences.

Fixed implementation In the fixed code, I correctly applied the `penalized_probs` and keep the cumulative *log-probs* instead of multiplying raw probabilities. With these changes, the algorithm produces more diverse and stable generations.

Reflections Diverse beam search is useful when we want multiple plausible but different outputs instead of one deterministic. For instance, brainstorming queries can benefit from it. However, it is not helpful for tasks like machine translation, where accuracy matters more than diversity.
Hamming diversity is cheap and effective, but only penalizes repeated tokens at the same decoding step. For example, given the prompt *"Write three different recipes using chicken"*, beams might diverge in the first token but converge to nearly identical recipes later. Here, Hamming scoring is

clearly not sufficient because it ignores semantic similarity. An alternative would be to add a semantic embedding penalty, which encourages beams to be semantically distinct. This would likely produce more diverse outputs in semantics. The cons is efficiency since computing embeddings at each step is expensive. Therefore, Hamming diversity is still suitable for a low-latency scenario.

Before fixed

Diverse Beam 1: Once upon a time, there was a little girl who loved to read. She loved to read so much that she would read

Diverse Beam 2: Once upon a time, there was a little girl who loved to read. She loved to read so much that she would read

Diverse Beam 3: Once upon a time, there was a little girl who loved to read. She loved to read so much that she would read

After fixed

Diverse Beam 1: Once upon a time, in a land far, far away, there was a little girl who loved to read. She loved

Diverse Beam 2: Once upon a time, there was a little girl who loved to read. She loved to read so much that she would read

Diverse Beam 3: Once upon a time, I had a dream. I wanted to be a writer. I wanted to write a book. I

□

3.2 Hugging Face implementations: OddSampling

The current (as of September 2025) Hugging Face `transformers` implementation of diverse beam search (and other inference algorithms) is modular and differs from the monolithic functions we provide for homework questions. Hugging Face `transformers` uses objects called `LogitsProcessors` that handle only the postprocessing of scores at inference time.

Now imagine the ranking of next tokens where 1 is the most-probable next token and |V| is the least-probable next token. Write a LogitsProcessor that applies a decoding method called OddSampling, where we only sample from odd-numbered ranks in this ordered list (you can ignore ties). Provide the full text of your function below.

Solution:

```
class OddSamplingLogitsProcessor(LogitsProcessor):
    def __call__(self, input_ids: torch.LongTensor, scores: torch.FloatTensor) -> torch.FloatTensor:
        sorted_indices = torch.argsort(scores, dim=-1, descending=True)

        # Create a mask for odd and even ranks
        batch_size, _ = scores.shape
        mask = torch.zeros_like(scores, dtype=torch.bool)
        for b in range(batch_size):
            # odd ranks: 0, 2, 4, ... (0-based indexing)
            odd_positions = sorted_indices[b, 0::2]
            mask[b, odd_positions] = True
        # Set logits of even rank to -inf
        scores = scores.masked_fill(~mask, float("-inf"))
        return scores
```

□

3.3 Implementing Mirostat

Mirostat [1] is an inference algorithm in which the k in top-k sampling is dynamically adjusted at each generation step, towards a target "surprise" value τ directly related to perplexity. Specifically, at each step, the Zipf's exponent $\hat{s}$ is estimated from the observed distribution, and $\hat{s}$ is in turn used to estimate k. This is intended to yield a generated sequence that both avoids excessive repetition and incoherence. In this problem, you are asked to implement a Mirostat sampler. Start from the scaffolding code provided in `mirostat.py` – note that, as in the buggy diverse beam search implementation from above, we assume a monolithic function implementation not seen in standard Hugging Face `transformers`.

3.3.1 Exploration and visualization

Now try out your implementation with: three different values of τ; on two different prompts (Feel free to choose from: "Once upon a time," "If you give a mouse a cookie," "The capital of France is," "It was a dark and stormy night," "$3 + 5 =$ ", and the empty string prompt.); on two different model sizes (`meta-llama/Llama-3.2-1B` and `meta-llama/Llama-3.1-8B`). Generate until the total sequence length is 128, and use a temperature of 0.9, learning rate of 0.1, and initial $\mu = 2\tau$. For each of the $3 * 2 * 2 = 12$ combinations, report:

1. The sequence generated
2. Mean, median, and standard deviation of per-token perplexity
3. Sequence-level perplexity
4. Plot k, $\hat{s}$, μ, and surprisal error against generation step. Feel free to combine these plots into a single figure for visual simplicity.

Additionally, pick a single model size, prompt, and τ value, and include a set of 3 logit distribution plots for each τ value at generation steps 1, 10, and 100. Report which model, prompt, and τ you used.

Write a few sentences about your observations. e.g. What hyperparameters and settings does Mirostat sampling seem most or least sensitive to? Did you observe anything surprising? In general, what use cases does Mirostat seem well-suited for?

Solution:

Results across all 12 runs

Table 3.3.1 reports mean, median, and sequence-level perplexity, while Table 4 reports the generated sequences.

Model	Prompt	τ	Mean PPL	Median PPL	Std PPL	Seq PPL
1B	Once upon a time	2.0	2.005	1.403	2.062	7.423
1B	Once upon a time	3.0	2.462	1.856	2.397	11.723
1B	Once upon a time	4.5	2.310	1.453	2.122	10.078
1B	Mouse a cookie	2.0	2.343	1.261	2.624	10.416
1B	Mouse a cookie	3.0	2.431	1.928	2.243	11.370
1B	Mouse a cookie	4.5	2.061	1.531	1.811	7.852
8B	Once upon a time	2.0	2.540	1.707	2.495	12.674
8B	Once upon a time	3.0	2.271	1.437	2.370	9.689
8B	Once upon a time	4.5	2.693	1.271	2.969	14.773
8B	Mouse a cookie	2.0	1.065	0.159	2.081	2.902
8B	Mouse a cookie	3.0	2.154	1.387	2.396	8.615
8B	Mouse a cookie	4.5	2.167	1.787	1.875	8.735

Table 3: Mean, median, standard deviation, and sequence-level perplexity

Model	τ	Generated Sequence (truncated)
1B	2.0	Once upon a time, the British Empire was a force to be reckoned with. Being the largest empire of its time, it controlled an area that was more than 25%, or 20 million square kilometers. Fueled by a vibrant and industrialized society, the British Empire was a super-power. But after WW2 ended, Britain was left with a deficit of influence. Its economy was shrinking, and the country was struggling to make ends meet. Brexit, and the rise of populism, was one of the major factors that dealt the brutal blow to the British Empire. To be honest, England was not in the best position, politically, economically,
1B	3.0	Once upon a time, if you wanted to watch a package youâve had delivered, you emailed the company to obtain delivery details and asked the delivery company for pickup instructions. Given that there are so many services, carriers, and delivery companies, this can get a little tedious. On top of that, there are a lot of different communication systems to learn if you plan to buy a new package delivery service. If youâve got the time, you can pick up the phone and make the calls yourself. However, with so many services and formats to learn, you might really appreciate having a handy guide to get through the complex algorithms and program requirements of the package delivery business
1B	4.5	Once upon a time, there was a mushroom field. The mushrooms were all of different shapes and sizes. It was a lovely place, but it was always very hot there. The mushrooms liked shade. They got enough rain for growing, but not enough for drinking. One day, after walking into the field, the mushroom lady looked out and noticed that the light was growing darker and darker. Something was going to happen! Only one thing could save her: a man who could bring it back to the light!

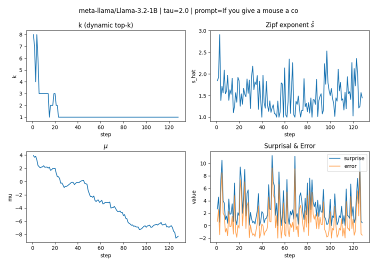
1B	2.0	If you give a mouse a cookie, do you ever think about what it thinks about? The muses have a lot to offer us, to both as individuals and as a species. In these sometimes silly but often serious musings, we learn about some muses who have experienced extraordinary and unusual life events along the way. These must-be-reads can help us understand ourselves and offer insight into the animal spirit. The mammalian muses and their association with mammals Tanaka, Yoji (2019) If you give a mouse a cookie, do you ever think about what it thinks about? Popular Science, 10 (04). pp. 64-65.
1B	3.0	If you give a mouse a cookie, youâre dancing a sort of âmouse dance.â If you live on a farm, you may want to use the same word in another way. In this lesson, weâll explain the origins of the word âmooseâ and how dairy farms are a prevalent place for this animal. African Blue Tongue Moebius Syndrome In 1991, the blue tongue Moebius Syndrome was recognized as an uncommon syndrome. Males are more likely to be the type of person afflicted by this rare disease, though females also canâ¦ Energy and Strength: The Strength and Stamina of Animals Animals are known for their energy and strength
1B	4.5	If you give a mouse a cookie, it seems, it will try to eat the cookie even if you take it back from a hungry dog. The same holds true for a certain kind of bug. In an experiment in the journal Science, mice were fed a special type of sugar and allowed to choose between eating it or letting a rat consume it. They discovered that they chose to eat the sugar no matter whether it would make them sick or not. The mice appeared to be immune to the effects of the sugar. For some reason, it was perceived to be non-toxic. The researchers found that the mice also had a taste for cinnamon. The scientists will also take into consideration

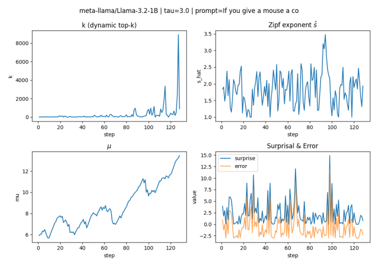
8B	2.0	Once upon a time, the current (and long-deferred) US ambassador to the UN, John Bolton, was thought to be a smart and knowledgeable guy. Well, he may well be still â but his boss, President George W. Bush, has him in charge of supplying the UN with more money than it can ever spend. False claims of tension between Palestine and Egypt, which Israel is using to divert its attention from Iran, were reported by the International Herald Tribune on 10 September 2004. The US has halted military aid to Indonesia, until the country shows it is serious about punishing those who brought down two planes over Bali using bombs disguised as
8B	3.0	Once upon a time, in a galaxy far, far away, there was a legacy not to be messed with. Lord of the Rings, arguably, is THE work of fantasy. Tolkienâs universe has the depth, the richness, a sharpness of detail which has caused many writers of the genre to turn green with envy. But the challenge is one that many writers have failed to meet. Dungeon crawls, grimdark, swords and sorcery. All titillating to be sure, but I didnât really want to write those stories any longer. They didnât seem to have the life in them they had 20 years ago. I wanted to write
8B	4.5	Once upon a time, cities throughout Europe, Asia, and the Middle East were graced with wondrous artworks. Their visitors would delight in painting, poetry, literature, and music. The streets were quiet, and the wealthy serenaded the lands with themes of love and art. Then one day, a strange craft flew over the land. Before the people knew it, people were having dreams about jet fighter planes on the utmost turrets of their castles. People began to freak out, not because of the craft, but because of what came along with it: the werewolf. The werewolf is a mythical creature that can never be tamed. Because
8B	2.0	If you give a mouse a cookie, he's going to ask for a glass of milk. When you give him the glass of milk, he's going to ask for a straw. If you give him the straw, he's going to ask for a napkin. If you give him the straw, and if after that you click this link to another website, you'll be taken to the page of your choice.

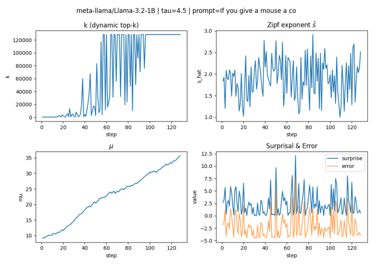
8B	3.0	If you give a mouse a cookie, he'll probably ask for a glass of milk. And if you give a glass of milk to a cat, it'll probably ask for some cookies. I knew that, but it took me quite awhile to realise that there was a major error in Dina's house-arrest stipulation. My Dina and I had been away for a few days, and the contract we had made with Mr. Justice Rouleau had stipulated that Dina return to our home, and be confined to her room other than going to work, school or medical appointments. We returned on Friday. Friday mid-afternoon, Dina's mother requested that
8B	4.5	If you give a mouse a cookie, youâll have to rinse out your teeth because thereâs goo on them. This happened to me, in a way, when I started researching the history of the cookie. I ate some cookies to help me write. I always find my creative process works better if I have food around that I donât feel guilty about eating. It also helps to have a nice glass of wine. Except my teeth were stained. I had to find out, the hard way, that cocoa is alkaline. In fact, cocoa powder, or theobroma cacao, has an approximate pH of 8.4, which is nearly the same as milk

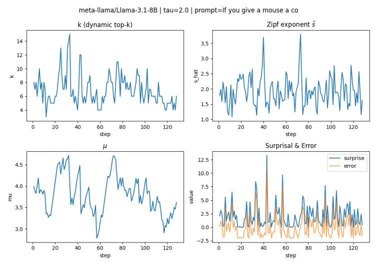
Table 4: Generated sequences for each combination

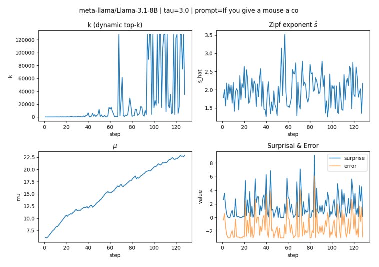
The quantities k, $\hat{s}$, μ, and surprisal error were tracked against generation step. The below figures illustrate these traces for both models across different τ values.

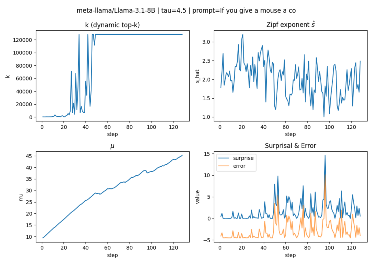






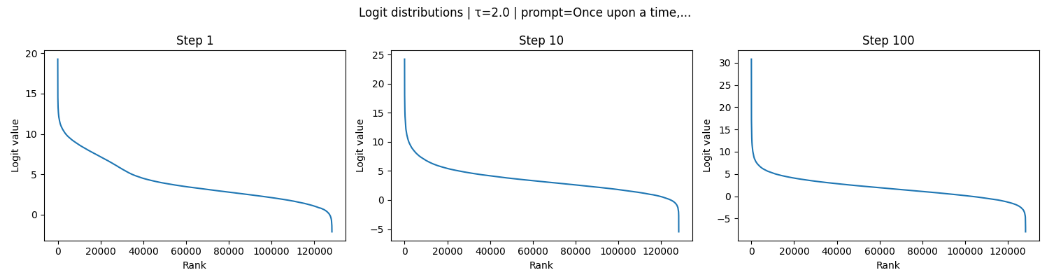


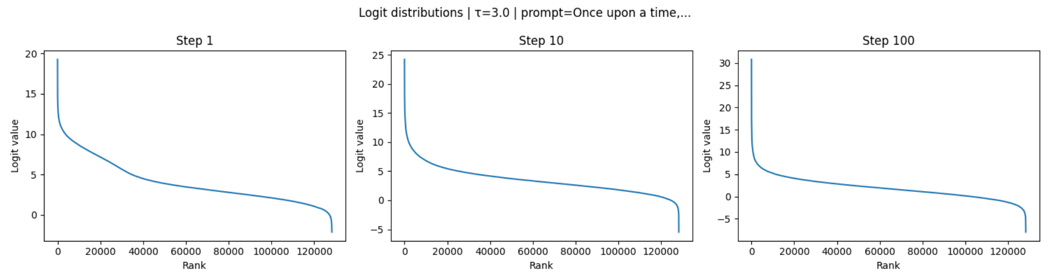


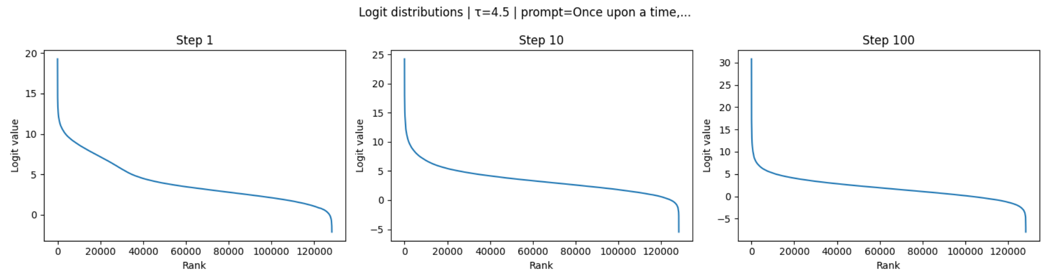


Logit distributions

`Llama-3.2-1B` was chosen with prompt "Once upon a time,". The below figures show the token logit distributions at steps 1, 10, and 100 for $\tau = 2.0, 3.0, 4.5$.







Observations

- With lower τ, the logit curves are sharper and the text is more repetitive and predictable. Higher τ allows more tokens to compete, so the text is more diverse and creative.
- As generation goes on, the overall shape of the logit curves remains pretty steady. This shows that Mirostat is good on keeping the distribution balanced, without collapsing to one token or spreading out too thin.
- The bigger 8B model generally gave lower perplexity and smoother surprisal error curves than the 1B model.
- The target τ mattered the most. Even small changes made the output noticeably more or less diverse based on the generated sequences.

Overall, Mirostat works best for creative writing or brainstorming, where you want variety without total randomness. However, it is less appropiate for questions like factual Q&A, where you usually want consistent answers.

$\square$

3.4 Benchmarking decoding strategies for each task

For this task, you'll use `Qwen/Qwen3-4B`, `Qwen/Qwen3-4B-Instruct-2507`, and `Qwen/Qwen3-1.7B`.

Report results on the "dev-test" split of each shared task using:

1. The default generation settings for each model, according to its Hugging Face `generation_config`.
2. Greedy decoding.
3. Temperature sampling with temperature $\tau = 0.25$ and $\tau = 1.5$.
4. Beam search with beam widths 3 and 25.
5. Locally typical sampling; choose your own reasonable hyperparameters

Report the scores for each model, method, and task in table(s). Write a few sentences about your observations: is the same decoding strategy the best for each task and model size? How do the instruct and base model differ? Do you notice any pathologies of the outputs?

Solution:

Decoding Strategy	Qwen3-1.7B	Qwen3-4B	Qwen3-4B-Instruct-2507
Default	0.0650	0.1017	0.0817
Greedy	0.0550	0.0967	0.0900
Temperature 0.25	0.0550	0.1017	0.0817
Temperature 1.5	0.0700	0.0933	0.0667
Beam 3	0.0683	0.0933	0.0767
Beam 25	0.0583	0.0950	0.0800
Typical	0.0650	0.0917	0.0767

Table 5: graph_dev scores

Decoding Strategy	Qwen3-1.7B	Qwen3-4B	Qwen3-4B-Instruct-2507
Default	0.57	0.77	0.85
Greedy	0.58	0.75	0.83
Temperature 0.25	0.56	0.80	0.84
Temperature 1.5	0.57	0.72	0.84
Beam 3	0.56	0.77	0.81
Beam 25	0.57	0.79	0.84
Typical	0.56	0.79	0.84

Table 6: mmlu scores

Decoding Strategy	Qwen3-1.7B	Qwen3-4B	Qwen3-4B-Instruct-2507
Default	0.74	0.81	0.81
Greedy	0.73	0.79	0.81
Temperature 0.25	0.75	0.78	0.79
Temperature 1.5	0.74	0.78	0.79
Beam 3	0.76	0.77	0.81
Beam 25	0.76	0.79	0.81
Typical	0.74	0.76	0.80

Table 7: infobench scores

Observations. Across all tasks, larger models consistently outperform the smaller one, with Qwen3-4B and Qwen3-4B-Instruct-2507 showing clear gains over Qwen3-1.7B. On the graph reasoning task, the Qwen3-4B slightly outperforms the Qwen3-4B-Instruct-2507 under all settings, whereas on mmlu and infobench the Qwen3-4B-Instruct-2507 outperforms Qwen3-4B.
Regarding decoding strategies, we observe that increasing the temperature from 0.25 to 1.5 on graph_dev improves the performance of the Qwen3-1.7B, but reduces accuracy for the larger Qwen3-4B. This may reflect that larger models already generate diverse outputs at lower temperatures, so additional randomness introduces noise rather than useful variation, while smaller models benefit from the added exploration. Beam search also shows a consistent trend that using a larger beam 25 generally improves performance compared to a smaller beam 3. However, excessively large beams can hurt performance, since they amplify biases toward shorter or higher-probability but less diverse outputs. □

3.5 Degenerate choices

Choose a pair of decoding strategies from class, A and B, and tune hyperparameters for each such that `Qwen/Qwen3-4B` with decoding strategy A underperforms `Qwen/Qwen3-1.7B` with decoding strategy B on the MMLU shared task. Report results along with the details of the decoding strategies used, including the values of any hyperparameters you chose.

Solution:

Model	Strategy	Hyperparameters	Average Score
Qwen3-4B	Temperature sampling	temperature = 20	0.56
Qwen3-1.7B	Greedy	—	0.58

Using an extremely high temperature ($\tau = 20$) with the 4B model flattens the next-token distribution so that decoding becomes almost random, which hurts performance. By contrast, the 1.7B model with greedy decoding stays stable and follows its learned patterns, ending up with a slightly higher score (0.58 vs. 0.56). $\square$

References

[1] Sourya Basu, Govardana Sachitanandam Ramachandran, Nitish Shirish Keskar, and Lav R. Varshney. Mirostat: A neural text decoding algorithm that directly controls perplexity, 2021.

[2] Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online, November 2020. Association for Computational Linguistics.

[3] W Brier Glenn et al. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[4] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[5] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

[6] Chenglei Si, Chen Zhao, Sewon Min, and Jordan Boyd-Graber. Re-examining calibration: The case of question answering. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2814–2829, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[7] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models, 2018.

[8] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[9] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.