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:

□

2.1.2 MLE and KL

In the class, we learned about Maximum Likelihood Estimation (MLE). Now consider a dataset $\mathcal{D} = \{x_1, \ldots, x_N\}$ 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:

□

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:

□

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>

□

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.

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.

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?

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.

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:

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

where $\text{Acc}(B_m)$ and $\text{Conf}(B_m)$ 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 (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			
C_2			
C_3			

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:

□

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:

□

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:

□

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:

□

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 pretrained 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.

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:

□

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?

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?

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.

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.