CS 489/698 Introduction to Natural Language Processing
Assignment 2: Decipher PCFG from Transformers

Instructors: Freda Shi and Victor Zhong
Released: Friday, February 13th, 2026
Due Date: Friday, March 6th, 2026 at 11:59 PM ET (Waterloo Time)

Winter 2026

1 Overview

In this assignment, you will act as a "*grammar cryptanalyst*." For each subtask, we (the instructors) secretly define a probabilistic context-free grammar (PCFG), sample a large synthetic corpus from it, and train a small Transformer language model on that corpus using the standard AdamW optimizer. For simplicity, the Transformers' tokens are exactly the terminals of the PCFG, i.e., there is no subword tokenization. You will be given the trained Transformer language model, but **not** the original PCFG.

Your goal is to **reconstruct the underlying PCFGs** (their production rules and probabilities) as accurately as possible *from the Transformers*. You may use the Transformers in any way you like (e.g., as a distribution proposer, by analyzing its hidden states, etc.) to recover the grammar. Your submission will be evaluated based on how well your recovered PCFG matches the true PCFG in terms of likelihood on a held-out test set of strings.

2 Provided Material

For each subtask, you will receive:

- A pretrained Transformer language model checkpoint.

Additionally, we also include

- Starter code to (1) load the model and the vocabulary, (2) compute next-token probabilities, (3) sample sequences. Example execution command: `python sampling.py checkpoints/pcfg3.pt --num-samples 10`;
- A Python 3.12 requirement file `requirements.txt` that specifies the dependencies for the starter code, where you may create a virtual environment by `python3.12 -m venv .venv`, activate it by `source .venv/bin/activate` (tested on Linux), and install the dependencies by `pip install -r requirements.txt`;
- A .csv file containing an example PCFG (PCFG 1) for Task 1 assigning uniform probabilities, which you can use as a reference of format;
- A script `trainer.py` that we use to train Transformers for your reference.

3 Submission Format

Your final submission should be made on LEARN, which includes:

- Grammar files named `pcfg1.csv, pcfg2.csv, pcfg3.csv`, respectively for each subtask, containing the production rules and their probabilities, with detailed formats specified in the starter code. **Note**: Please make sure the grammar files are in the root of your zip file (not inside a subfolder); otherwise we may not be able to grade your submission. The required columns are

  - ID: index of the rule, you can fill in any unique identifier for each rule (e.g., 1, 2, 3, ...).
  - LHS: the left-hand side of a rule, must be a nonterminal or preterminal symbol.
  - LHS Type: the type of the LHS symbol, either "nonterminal" or "preterminal".
  - RHS: the right-hand side of a rule, must be either two nonterminals (for nonterminal expansion rules) or one terminal (for preterminal rules).
  - Probability: the probability of the rule, which should be a number between 0 and 1, and for every nonterminal A, the probabilities of all rules with A as the LHS should sum to 1.

  Please make sure your grammar files contain **all** rules in your PCFG, including the ones provided in the task description.

- Your code that is used to generate the grammar files from the provided Transformer checkpoints under `code/`. It is okay to have separate code for different subtasks, or have subfolders in `code/` for different subtasks. While you are encouraged to make sure the code executes and directly output the grammar files, you are allowed to propose a solution that requires some manual steps (e.g., inspecting model outputs and modifying rules by hand), as long as you provide clear instructions on how to reproduce your final grammar files from the checkpoints.

- A `README.md` file describing your method and solution choices.

Meanwhile, we will offer a Kaggle leaderboard (released on February 16 on the course page) for each task for you to offer some real-time feedback on your submission's performance on 5% of the hidden test set.

4 Backgrounds

4.1 Probabilistic Context-Free Grammars (PCFGs)

A PCFG consists of a set of nonterminals $\mathcal{N}$,[1] terminals Σ, a start symbol $S \in \mathcal{N}$, and a set of production rules $\mathcal{R}$. Each rule $r \in \mathcal{R}$ has the form $A \to \alpha$ with probability $p(r)$, where $A \in \mathcal{N}$ and $\alpha \in (\mathcal{N} \cup \Sigma)^*$. For every nonterminal A, the probabilities of rules expanding A sum to 1.

The probability of a parse tree t is the product of its rule probabilities, and the probability of a string x is the sum over all parse trees that yield x:

$$P(x) = \sum_{t \in \mathcal{T}(x)} \prod_{r \in t} p(r)$$

where $\mathcal{T}(x)$ is the set of parse trees that produce x. In this assignment, you will build a PCFG that places high probability mass on the same strings as the teacher Transformer.

[1] Preterminals can be viewed as a special type of nonterminals.

4.2 Transformer Training

We train a small Transformer language model (using the standard GPT2 architecture) on a large synthetic corpus generated by the PCFG. We sample 100K strings from the PCFG, and train the Transformer for 10 epochs with a batch size of 64 using the standard AdamW optimizer and the negative log probability–based next-token prediction objective. The learning rate is set to 5e-4 with a linear "warmup"[2] over the first 5% of training steps, and a weight decay of 0.01.[3] You are encouraged to explore the training script and understand the training process, although you are not asked to train any models yourself for this assignment. More detailed hyperparameter settings can be found by loading the provided checkpoints, and inspect the loaded model's `config` attribute.

5 Tasks

In all tasks, the PCFG will be in its Chomsky Normal Form (CNF)—all nonterminals expands to either two nonterminals or one terminal—and the start symbol is always S. Throughout this assignment, there is a set of preterminal nonterminals that only expand to one terminal, and all other nonterminals only expand to two nonterminals. **Your submission needs to be in the Chomsky Normal Form as well**, that is, every rule in your grammar file should be either of the form $A \to B\ C$ or $A \to w$, where A, B, C are nonterminals and w is a terminal.

For simplicity, all PCFG rules will have a probability of at least 0.02.

5.1 Task 1 (20 points): Getting Started

Let's first work with a simple PCFG (PCFG 1) that only contains ten rules (probabilities omitted here):

$$\begin{aligned}
\text{S} &\to \text{NP VP} \\
\text{S} &\to \text{NNS VP} \\
\text{NP} &\to \text{DT NNS} \\
\text{VP} &\to \text{V NNS} \\
\text{DT} &\to \text{the} \\
\text{NNS} &\to \text{dogs} \\
\text{NNS} &\to \text{cats} \\
\text{NNS} &\to \text{mice} \\
\text{V} &\to \text{chase} \\
\text{V} &\to \text{like}
\end{aligned}$$

Your goal is to reconstruct this PCFG from the provided Transformer checkpoint (`pcfg1.pt`), by estimating the probabilities of the above rules.

[2]Warmup is a common neural network training strategy that gradually increases the learning rate from 0 to the target value over the beginning steps; see `https://neurips.cc/virtual/2024/poster/95431`

[3]For more about weight decay, this recent paper offers an interesting view: `https://proceedings.neurips.cc/paper_files/paper/2024/file/29496c942ed6e08ecc469f4521ebfff0-Paper-Conference.pdf`

5.2 Task 2 (50 points): A "Realistic" PCFG

Now let's move on to a more realistic PCFG (PCFG 2) that contains 220 rules in total, with 20 nonterminal expansion rules (in the form of $A \to B\ C$) and 200 preterminal rules (in the form of $A \to w$).

In this PCFG, the terminals are English words, and the nonterminals are phrase categories (e.g., noun phrases, determiners, etc.). You may use your knowledge of English syntax to help you recover the grammar, for example, "cats" and "dogs" are likely to be nouns (and they are, therefore, of the same category). You may assign whatever symbol you like to the nonterminals, as long as the internal rules and their probabilities are correctly represented, and the preterminal rules correctly map the nonterminals to the terminals. As an example, the following two PCFGs are equivalent and will get the same score, as they only differ in the choice of nonterminal symbols:

(PCFG a)

$$
\begin{aligned}
\text{S} &\to \text{NNS V}(p = 1)\\
\text{NNS} &\to \text{dogs}(p = 0.5)\\
\text{NNS} &\to \text{cats}(p = 0.5)\\
\text{V} &\to \text{meows}(p = 0.3)\\
\text{V} &\to \text{barks}(p = 0.7)
\end{aligned}
$$

(PCFG b)

$$
\begin{aligned}
\text{S} &\to \text{A B}(p = 1)\\
\text{A} &\to \text{dogs}(p = 0.5)\\
\text{A} &\to \text{cats}(p = 0.5)\\
\text{B} &\to \text{meows}(p = 0.3)\\
\text{B} &\to \text{barks}(p = 0.7)
\end{aligned}
$$

Please note that the sentences generated by this PCFG are not necessarily grammatical English sentences, for example, we do not distinguish plural noun phrases (e.g., "the dogs") from singular noun phrases (e.g., "the dog") in the nonterminals (both falling into the category NP), so there might be some awkward sentences (e.g., "the dogs meows") in the training data.

5.3 Task 3 (30 points): A Purely Synthetic PCFG

Now let's move on to a purely synthetic PCFG (PCFG 3) that contains 26 preterminal rules with probability 1. Without loss of generality, we can assume these rules are $A \to a$, $B \to b$, ..., $Z \to z$, where $A, B, ..., Z$ are the preterminal nonterminals and $a, b, ..., z$ are the terminals. There are additionally 50 nonterminal expansion rules (in the form of $NT_1 \to NT_2\ \ NT_3$) with various probabilities. Your task is to recover the structure and probabilities of the PCFG, without any prior knowledge about the syntax.

6 Evaluation Metrics

Your submission will be primarily evaluated based on the total variation distance (TVD) between the distribution of your PCFG and the ground-truth PCFG on a hidden test set of strings, which will be described in detail below.

In addition, for Tasks 2 onwards, where you need to derive your own PCFG structure, we will also evaluate the number of rules in your PCFG as a secondary metric, where fewer rules is considered better. If number of rules is considered as a metric, it will compose 10% of the task marks.

Total Variation Distance (TVD)

The hidden test set consists of up to 10K unique strings from the ground-truth PCFG (note: they may or may not overlap with the training set), and your score will be based on the estimated total variation distance (TVD) between the distribution of your PCFG and the ground-truth PCFG on the test samples:

$$\text{TVD}(P, Q) = \frac{1}{2} \sum_{x} |P(\mathbf{x}) - Q(\mathbf{x})|$$

where P is the distribution of the ground-truth PCFG, Q is the distribution of your PCFG, and $\mathbf{x}$ is a string in the test set. You might have recognized that TVD is equivalent to the L1 distance between two distributions, and is always between 0 and 1. Each $P(\mathbf{x})$ and $Q(\mathbf{x})$ will be calculated by summing over the probabilities of all parse trees that yield $\mathbf{x}$ with the inside algorithm.
Your final score will be compared against two baselines:

- **Uniform PCFG** (0-mark baseline for tasks with ground-truth CFG provided, 25% mark baseline for other tasks): a PCFG with the same structure as the ground-truth PCFG but with uniform probabilities for all rules expanding the same nonterminal (e.g., for PCFG 1, $P(\text{NP} \to \text{NNS}) = P(\text{NP} \to \text{DT NNS}) = 0.5$).
  - When uniform PCFG is serving as the 25%-mark baseline, the 0-mark PCFG will be a degenerate PCFG that only contains the start symbol S and a single rule $S \to \epsilon$ where ϵ is an empty string, which assigns zero probability to all non-empty strings.
- **Empirical PCFG** (100%-mark baseline): we implement a simple, sampling-and-counting-based method to fit a PCFG from the Transformer.

Your mark will be the interpolation between the baselines based on your TVD score. Here are three examples:

- **Example 1**: Suppose the TVD between your PCFG and the ground-truth is 0.1, the TVD of the uniform PCFG is 0.3 (as a 0-mark baseline), and the TVD of our empirical PCFG is 0.05, then your mark will be $\frac{0.3-0.1}{0.3-0.05} = 80\%$.
- **Example 2**: Suppose the TVD between your PCFG and the ground-truth is 0.2, the TVD of the uniform PCFG is 0.3 (as a 25%-mark baseline), and the TVD of our empirical PCFG is 0.05, then your mark will be $\frac{0.3-0.2}{0.3-0.05} \times 75\% + 25\% = 58.3\%$.
- **Example 3**: Suppose the TVD between your PCFG and the ground-truth is 0.4, the TVD of the uniform PCFG is 0.3 (as a 25%-mark baseline), and the TVD of the zero-mark PCFG is defined as 1, then your mark will be $\frac{1-0.4}{1-0.3} \times 25\% = 21.4\%$.

Bonus Marks

For each of Tasks 2 and 3, if your TVD is better than our empirical PCFG baseline, you will get a bonus mark of 1 point for every 0.01 improvement in TVD, up to a maximum of 8 bonus points per task (i.e., 16 possible bonus points in total for Tasks 2 and 3).

Number of Rules

W only count the nonterminal expansion rules (in the form of $A \rightarrow B\ C$) for this metric. If your number of PCFG rules is no more than 150% of the number of rules in the ground-truth PCFG, you will get the full score for this metric; otherwise, your score will be linearly scaled down to zero when your number of rules reaches 200% of the number of rules in the ground-truth PCFG.

7 Tips

- Throughout this assignment, a terminal symbol can only be generated by one preterminal.
- Start by sampling from the teacher model and looking for obvious structure (e.g., brackets, separators, recurring patterns).
- Training models on your own PCFG might be helpful.