APPENDIX

A PU LOSS DERIVATION

PU losses are derived from the canonical binary classification framework. In the standard supervised binary classification (or Positive-Negative classification, abbreviated as PN), let $\pi := p(Y = +1) = \frac{n_P}{n_P+n_N}$ be the prior probability of the positive class, $g : \mathbb{R}^d \to \mathbb{R}$ be an arbitrary decision function (in our case, the detector model) and L be the loss function. The risk of g is defined as the expectation of loss:

$$\begin{aligned} R(g) :=& \mathbb{E}_{(X,Y)\sim p(x,y)}[L(g(X), Y)] \\ =& \pi \mathbb{E}_p[L(g(X), +1)] + (1-\pi)\mathbb{E}_n[L(g(X), -1)] \\ =& \pi R_P(g, +1) + (1-\pi)R_N(g, -1). \end{aligned} \tag{11}$$

In canonical PN learning, $R(g)$ can be approximated directly by losses calculated from training data as follows:

$$\hat{R}_{PN}(g) = \pi \hat{R}_P(g, +1) + (1-\pi)\hat{R}_N(g, -1), \tag{12}$$

where $\hat{R}_P(g, +1) := \frac{1}{n_P}\sum_{i=1}^{n_P} L(g(x_i^P), +1)$ and $\hat{R}_N(g, -1) := \frac{1}{n_N}\sum_{i=1}^{n_N} L(g(x_i^N), -1)$ are estimations of the positive and negative risk, respectively.

In the PU framework, $\hat{R}_N(g, -1)$ cannot be approximated directly via negtive samples. Alternatively, some works (Du Plessis et al., 2014; Plessis et al., 2015) perform indirect approximation as follows: defining $p_P(x) := p(x|Y = +1)$ and $p_N(x) := p(x|Y = -1)$, since

$$(1-\pi)p_N(x) = p(x) - \pi p_P(x), \tag{13}$$

the negative risk part (which is an expectation) is obtained as

$$(1-\pi)R_N(g, -1) = R_U(g, -1) - \pi R_P(g, -1), \tag{14}$$

and $R(g)$ can be approximated indirectly as

$$\hat{R}_{uPU}(g) = \pi \hat{R}_P(g, +1) - \pi \hat{R}_P(g, -1) + \hat{R}_U(g, -1), \tag{15}$$

where $\hat{R}_P(g, -1) := \frac{1}{n_P}\sum_{i=1}^{n_P} L(g(x_i^P), -1)$ and $\hat{R}_U(g, -1) := \frac{1}{n_U}\sum_{i=1}^{n_U} L(g(x_i^U), -1)$ are estimations calculated from positive and unlabeled training samples. Eq. 15 is defined as the unbiased PU (uPU) loss (Du Plessis et al., 2014).

B ESTIMATION DETAILS OF CONFIDENCE EXPECTATION

The transition matrix Given positive probability p of a single token, we express state transition as a band matrix $\mathbf{P}$. An example matrix form of $\mathbf{P}$ is listed as follows:

$$\begin{bmatrix} 1-p & p & 0 & 0 & ... & 0 & 0 & 0 \\ 1-p & 0 & p & 0 & ... & 0 & 0 & 0 \\ 0 & 1-p & 0 & p & ... & 0 & 0 & 0 \\ & & & & ... & & & \\ & & & & ... & & & \\ & & & & ... & & & \\ 0 & 0 & 0 & 0 & ... & 1-p & 0 & p \\ 0 & 0 & 0 & 0 & ... & 0 & 1-p & p \end{bmatrix}$$

Demonstration of $\tilde{\pi}$ increment with respect to lengths We try to mathematically demonstrate that prior $\tilde{\pi}$ increases with length l. The initial state σ_0 is one-hot, so the prior $\tilde{\pi}(l)$ with respect to l could be written as:

$$\tilde{\pi}(l) = E\left[\Delta(S_l)\right] = \sigma_0 \mathbf{P}^l \alpha^T = \mathbf{P}[n, :]\mathbf{P}^{l-1}\alpha^T, \tag{16}$$

where $\mathbf{P}[n,:]$ represents the last row of transition matrix $\mathbf{P}$. To demonstrate $\tilde{\pi}$ increases with l, we alternatively demonstrate $\tilde{\pi}(l+1) - \tilde{\pi}(l) = E\left[\Delta(S_{l+1})\right] - E\left[\Delta(S_l)\right]$ is positive.

However, sizes of states and transition matrices are different for corpora of different lengths. We use a subscript to indicate this difference. For instance, sequence vector $\alpha_l := [i/l]_{i=0}^{l}$ indicates all possible confidences in a sorted sequence; $\mathbf{P}_l$ indicates the transition matrix $\mathbf{P}$ of size $(l+1)\times(l+1)$. Then:

$$E\left[\Delta(S_{l+1})\right] - E\left[\Delta(S_l)\right] = \mathbf{P}_{l+1}[n,:]\mathbf{P}_{l+1}^{l-1}\mathbf{P}_{l+1}\alpha_{l+1}^T - \mathbf{P}_l[n,:]\mathbf{P}_l^{l-1}\alpha_l^T \tag{17}$$

Interestingly, we could leverage unique features of the sparse band matrix $\mathbf{P}$. First, obviously $\mathbf{P}_{l+1}[n,:] = [0; \mathbf{P}_l[n,:]]$. Further, if we compare

$$M := \mathbf{P}_{l+1}[n,:]\mathbf{P}_{l+1}^{l-1} \in \mathbb{R}^{l+2} \quad \text{and} \quad K := \mathbf{P}_l[n,:]\mathbf{P}_l^{l-1} \in \mathbb{R}^{l+1},$$

we would discover that $M = [0; K]$, namely, array M is array K prepended by a zero. (The physical meaning of M and K is the last line of matrix $\mathbf{P}_{l+1}^l$ and $\mathbf{P}_l^l$, respectively.) Based on this discovery, we could simplify Eq. 17:

$$E\left[\Delta(S_{l+1})\right] - E\left[\Delta(S_l)\right] = [0; K]\mathbf{P}_{l+1}\alpha_{l+1}^T - K\alpha_l^T \tag{18}$$

Then we look at the concrete form of $[0; K]\mathbf{P}_{l+1}$. For simplicity, we denote the n^{th} element of K as k_n:

Count	0	1	2	...	n	$n+1$
$[0; K]$	0	k_0	k_1	...	k_{n-1}	k_n
$[0; K]\mathbf{P}_{l+1}$	$(1-p)k_0$	$(1-p)k_1$	$pk_0 + (1-p)k_2$	...	$pk_{n-2} + (1-p)k_n$	$pk_{n-1} + pk_n$

Based on the table above, we could derive the relations between $E\left[\Delta(S_{l+1})\right]$ and $E\left[\Delta(S_l)\right]$:

$$\begin{aligned} E\left[\Delta(S_{l+1})\right] - E\left[\Delta(S_l)\right] &= \frac{\sum_{n=0}^{l} nk_n}{l+1} + \frac{2p}{l+1} - \frac{k_l p}{l+1} - \frac{\sum_{n=0}^{l} nk_n}{l} \\ &= -\frac{\sum_{n=0}^{l} nk_n}{(l+1)l} + \frac{2p - k_l p}{l+1} \\ &= -\frac{E\left[\Delta(S_l)\right]}{l+1} + \frac{2p - k_l p}{l+1}, \end{aligned} \tag{19}$$

which means that

$$E\left[\Delta(S_{l+1})\right] = \frac{l}{l+1} E\left[\Delta(S_l)\right] + \frac{2p - k_l p}{l+1}, \tag{20}$$

As long as we view $\{l \times E\left[\Delta(S_l)\right]\}$ as a sequence of corpus length l starting from $1 \times E\left[\Delta(S_1)\right] = p$, we could solve $E\left[\Delta(S_l)\right]$ for $l > 1$:

$$E\left[\Delta(S_l)\right] = \frac{(2l-1)p - p\sum_{n=1}^{l-1} k_{n,n}}{l} = 2p - \frac{p}{l}(1 + \sum_{n=1}^{l-1} k_{n,n}), \tag{21}$$

where $k_{n,n}$ is the probability of the abstract recurrent model outputting positive confidence 1 for a corpus of length n. However, we encounter the difficulty that the analytic solution to $k_{n,n}$ is not easily solvable; we only know that $k_{n,n}$ is a probability bounded in $(0, 1)$. We inspect $k_{n,n}$ for relatively small p and found that $k_{n,n}$ quickly converges to 0. This process is demonstrated by Figure 1, where $k_{n,n}$ decays in an approximately exponential manner to infinitesimally small values (which decays much faster than reciprocals, i.e. $1/l$). As a result, prior $\tilde{\pi}$ keeps increasing

as l increases, and converges to $2p$. Figure 1 (Right) confirms the convergence derived in Eq. 21.

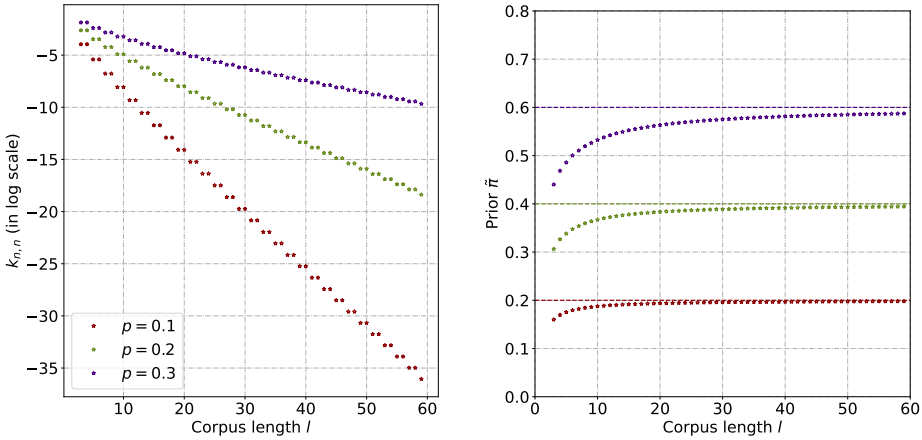

Figure 1: **Left:** $k_{n,n}$ (in log scale) with respect to corpus length l. **Right:** $\tilde{\pi}$ with respect to corpus length l.

C Proposal of Imposing Space Cleaning on the HC3-English Benchmark

We use the HC3 (Guo et al., 2023) benchmark for ChatGPT corpus detection experiments. However, we inspected HC3 corpora and discovered that the corpora are flawed: human corpora have additional spaces before punctuations, while corpora from AI do not have this feature. The extra spacing could directly impact the input to detectors. We list several examples below, demonstrating the obvious difference between Human and ChatGPT corpora in the HC3 benchmark (Guo et al., 2023):

```
# labeled as Human
corpus = 'Basically there are many categories of " Best Seller " .'
input_ids = [0, 34480, 89, 32, 171, 6363, 9, 22, 2700, 44795, 22, 479, 2]

corpus = 'Same thing for best sellers .'
input_ids = [0, 42271, 631, 13, 275, 12649, 479, 2]

corpus = 'Also , IIRC the rankings change every week or something like
    that .'
input_ids = [0, 22412, 2156, 3082, 5199, 5, 8359, 464, 358, 186, 50, 402,
     101, 14, 479, 2]

# labeled as ChatGPT
corpus = 'It is generally not acceptable or ethical to advocate for or
    condone the assassination of any individual , regardless of their
    actions or beliefs.'
input_ids = [0, 243, 16, 3489, 45, 9796, 50, 13557, 7, 7156, 13, 50,
    35005, 5, 16351, 9, 143, 1736, 6, 6069, 9, 49, 2163, 50, 9734, 4, 2]

corpus = 'There are also practical considerations at play in this
    situation.'
input_ids = [0, 970, 32, 67, 7708, 19199, 23, 310, 11, 42, 1068, 4, 2]

corpus = 'It can also lead to further conflict and instability in the
    region.'
input_ids = [0, 243, 64, 67, 483, 7, 617, 3050, 8, 16826, 11, 5, 976, 4,
    2]
```

In the examples, we show original corpus as well as their token ids after being processed by the RoBERTa-base tokenizer. Most human corpora have an unexpected 479 token (standing for " .", i.e. a space and a period), while ChatGPT corpora does not manifest this feature.

Hence, the detector could judge the attribution of a certain corpus simply by detecting these spacing mistakes. Embarrasingly, if we use the logical judgement of whether token id 479 is contained in the sequence to detect human corpora, the F1 score would reach 82.12% on sentence-level test corpora of the HC3 benchmark. The performance of such a simple logic is even better than the officially reported performance (81.89%) of finetuned RoBERTa-base (Guo et al., 2023). Above all, we strongly recommend later works that involve the HC3 benchmark to remove unnecessary spaces before punctuations. We will opensource the code simple cleaning helper function that removes unnecessary spaces.

D BASELINE REPLICATIONS

D.1 DETECTGPT

DetectGPT (Mitchell et al., 2023) is a latest open-sourced AI corpus detection baseline, but the original paper did not report its performance on latest LLM texts. Hence, we replicate DetectGPT on the HC3-English (Guo et al., 2023) ChatGPT corpus dataset, and compare it with our MPU method. The experiment results are shown in Table 4, where our MPU method outcompetes DetectGPT by large margins. There is still a visible gap between latest training-agnostic methods (e.g. DetectGPT) and finetuned language models on ChatGPT corpora.

We also provide some detailed procedures to tailor DetectGPT for the HC3 benchmark: 1. Full-scale HC3 corpora are always too long to perturb. Therefore, we truncate corpora as long as they raise perturbation errors, following recommendations from authors of DetectGPT. 2. We use 100 perturbations for full-scale HC3 corpora (following DetectGPT (Mitchell et al., 2023)), but we use 10 perturbations for sentence-level HC3 because there are too many corpora. It also reflects that DetectGPT is not very efficient for large-scale corpora compared to language model detectors, because it requires tens of model runs for a single corpus. 3. DetectGPT uses AUROC as the classification metric; however, this metric is not applicable to finetuned language models that output probabilities for respective classes. Hence, given confidences of all corpora outputted from DetectGPT, we choose 1000 equally-spaced threshold between max and min values, and maintain the threshold with the largest F1 score. Notably, this will provide an upperbound for the performance of DetectGPT, as in real applications the threshold is pre-set; scanning for the best threshold on test sets is strictly prohibited.

D.2 GLTR, PPL, & OPENAI

These methods have already been open-sourced on HuggingFace. We directly input all texts in the testset to these baseline methods and measure their performances.

We have found an inconsistency in comparison to reported values while replicating GLTR (Gehrmann et al., 2019) and RoBERTa-Finetuned (Cui et al., 2020) on the HC3-Chinese (Guo et al., 2023) benchmark, shown in Table 11. This inconsistency is tolerable and won't affect our final conclusion.

Method	Full	Sent
GLTR (Reported by Guo et al. (2023))	89.61	44.02
GLTR (Replicated)	87.40	49.94
RoBERTa-Finetuned (Reported by Guo et al. (2023))	98.79	83.64
RoBERTa (Replicated)	96.28±3.42	83.07±6.85

Table 11: Our replication of HC3-Chinese Guo et al. (2023) baselines compared with reported values.

E REPLICATION DETAILS

Following the training setting of Kumarage et al. (2023), we use batchsize 16, learning rate $1e-5$ for TweepFake; following the setting of Guo et al. (2023), we use batchsize 32, learning rate $5e-5$ for HC3. AdamW optimizors are adopted. Selected benchmarks are publicly accessible online.

We use a single Nvidia Tesla V100 as the device for experiments. A single epoch of training costs around 30 minutes. We replicate all experiments three times to avoid fluctuation, using seed=0,1,2. The codes are opensourced at GitHub and Gitee.