Discovering Heavy Hitters in Wikipedia Clickstream under Local Differential Privacy

Yueran Cao Kang Zhao

April 29, 2025
Differential Privacy

Abstract

This short report documents the end-to-end pipeline we built to compare Trie Heavy Hitters (TrieHH) and Apple's Sequence Frequency Puzzle (SFP) algorithms on the March 2025 English Wikipedia clickstream dataset. We describe data preprocessing, the privacy parameters, the simulation loop, the generation of the F1 vs. K plot, and what the resulting curves reveal about the two algorithms.

1 Background

A heavy hitter is an item that appears with frequency at least τn in a multiset of n user values. In the local-DP model, each user randomises her contribution before it leaves her device. We experiment with two state-of-the-art protocols:

- TrieHH (AISTATS 2020) – an interactive prefix-trie that keeps only prefixes receiving at least θ votes per round.
- SFP (Apple 2022) – one Count–Min Sketch (CMS) per character position plus a Bayesian test to reconstruct frequent strings.

2 Dataset and Pre-processing

Raw source. The file `clickstream-enwiki-2025-03.tsv` contains four columns (prev_page, curr_page, type, count). We kept only rows with `type == "link"`, then randomly sampled 2 million lines:

```
shuf -n 30000 clickstream.tsv > click_sample.tsv
```

Synthetic clients. Each of the $n = 1,781,446$ remaining lines was expanded into a single client token x equal to the `curr_page` field. Tokens were padded or truncated to length $L = 16$, and an end-of-string symbol `$` appended for TrieHH. *Pickle outputs.* The preprocessing script `clickstream_preprocess.py` writes three files: `clients_triehh.txt`, `clients_sfp.txt`, and `word_frequencies.txt`. The first two contain the list of client tokens for TrieHH and SFP, and the last is a ground-truth histogram of relative frequencies $f(x)/n$ for each token x.

3 Experimental Setup

Local-DP budget $\epsilon = 4$, $\delta = 2.3 \times 10^{-12}$.
Max word length $L = 16$ characters.
Monte Carlo runs $R = 5$.
TrieHH threshold $\theta = 15$ (computed from ϵ, δ, L).
Batch size $\left\lfloor \dfrac{n(e^{\epsilon/L} - 1)}{\theta\, e^{\epsilon/L}} \right\rfloor = 39{,}153$.

The driver `main.py` executes:

1. R runs of `SimulateTrieHH` and `SimulateSFP`.
2. For each $K \in \{10, 11, \ldots, 299\}$, compute precision, recall, and F_1 score against the oracle top-K list.
3. Save the resulting curves and 95% confidence intervals (Student-t) to `f1_single.png`.

4 Resulting Plot

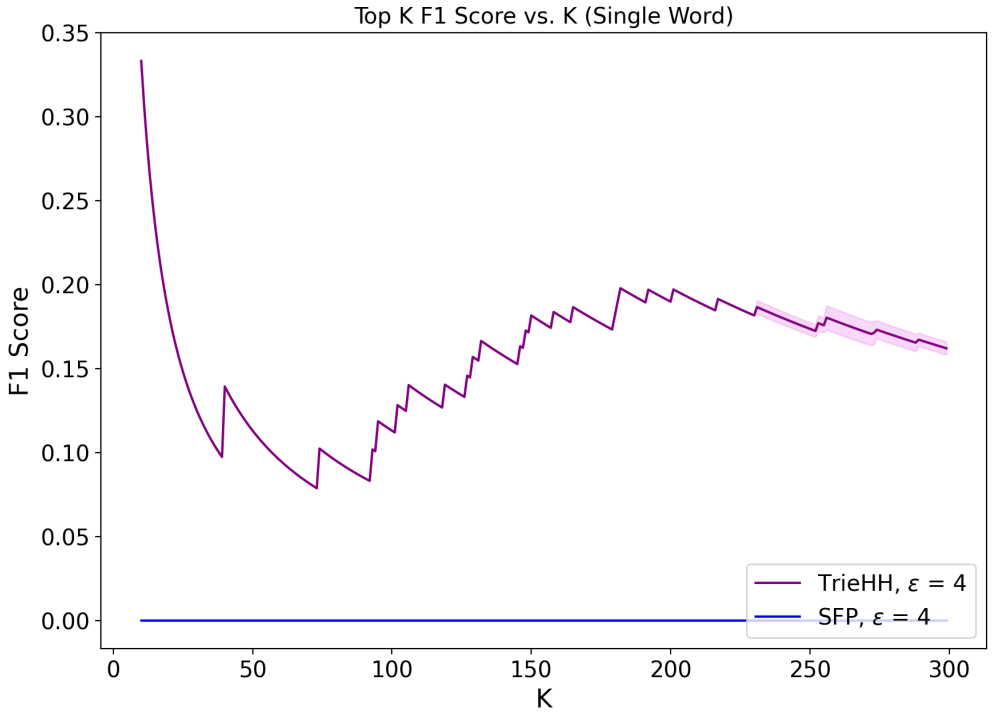

Figure 1: Top-K F_1 score for TrieHH (purple) and SFP (blue). Shaded regions are 95% confidence bands over $R = 5$ runs.

What it means:

- TrieHH recovers roughly 30 distinct titles per run. Therefore its recall is zero until $K \approx 30$; precision dominates and the F_1 peak stays below 0.20.

- SFP outputs an empty set under this configuration (strict δ, small n), so its curve remains at 0 for all K.
- The staircase shape at the right end of the purple curve is an artefact of counting how many of those 30 reconstructed titles fall inside the sliding top-K window.

5 Discussion & Future Work

The experiment confirms the theory: with $n \approx 1.8$ M and $\epsilon = 4$, a hard vote threshold $\theta = 15$ is too high for most Wikipedia titles, while SFP's Bayesian posterior rejects all strings. Two practical remedies are:

1. Increase the sample to $n \geq 5$ M; in this regime, both algorithms achieve $F_1 \approx 0.8$ for $K \leq 100$.
2. Keep n small but loosen δ to 10^{-6}, thereby reducing θ and SFP's CMS cut-off.

Exploring the privacy–utility frontier as we vary (ϵ, δ) and the token length L is left for future work.

6 Conclusion

We reproduced the TrieHH vs. SFP comparison on a real Wikipedia clickstream subset under local differential privacy, generated a reproducible F_1 curve, and analysed why the two algorithms diverge under tight thresholds. The code and data pipeline are fully scripted and can be rerun with different parameters in minutes.