Discovering Heavy Hitters in Wikipedia Clickstream under Local Differential Privacy

Yueran Cao yc1306

Kang Zhao kz262

Date: 4/30/2025

GitHub Repository: github.com/YueranCao2001/DP_Final_Project

17 89

GEORGETOWN UNIVERSITY

Outline

- Motivation & Problem Statement
- Initial Plan and Approach
- Algorithms Implemented (LDP Heavy Hitters)
- Difficulties & Plan Deviations
- Results – Top-K F1 Score vs. K (ε=4)
- Result Observations
- Lessons Learned
- Conclusion

Motivation & Problem Statement

- **Heavy Hitters:**
  most frequent items in a dataset (e.g. popular search queries or visited pages)

- **Challenge:**
  Finding global heavy hitters *without* violating individual user privacy

- **Local Differential Privacy (LDP)**
  ensures each user's data is randomized **before** sharing (protecting personal info)

- **Goal**:
  Explore an LDP solution for heavy hitters – identify top trends **privately**

Initial Plan and Approach

- Simulate the **client-server pipeline** in one program (no actual network needed)
- Generate **synthetic user data** to fine-tune conditions (control domain size, distribution shape, known heavy hitters)
- Planned to evaluate under various scenarios:
  - Different data distributions (uniform vs. skewed frequency)
  - Different domain sizes (number of unique items)
  - Different privacy levels (varying ε values)
- **Why**: Understand how privacy settings and data characteristics affect heavy-hitter detection accuracy

Algorithms Implemented (LDP Heavy Hitters)

- **TrieHH (AISTATS 2020)** – interactive, prefix-trie based algorithm
  – Builds strings character by character, keeping only prefixes with enough votes (post-noise) each round

- **SFP – Sequence Frequency Puzzle (Apple 2022)** – batch (non-interactive) algorithm
  – Each user reports to a Count-Min Sketch per character position + a Bayesian inference to reconstruct frequent strings

- Implemented both protocols to compare their performance on the same data under LDP

Difficulties & Plan Deviations

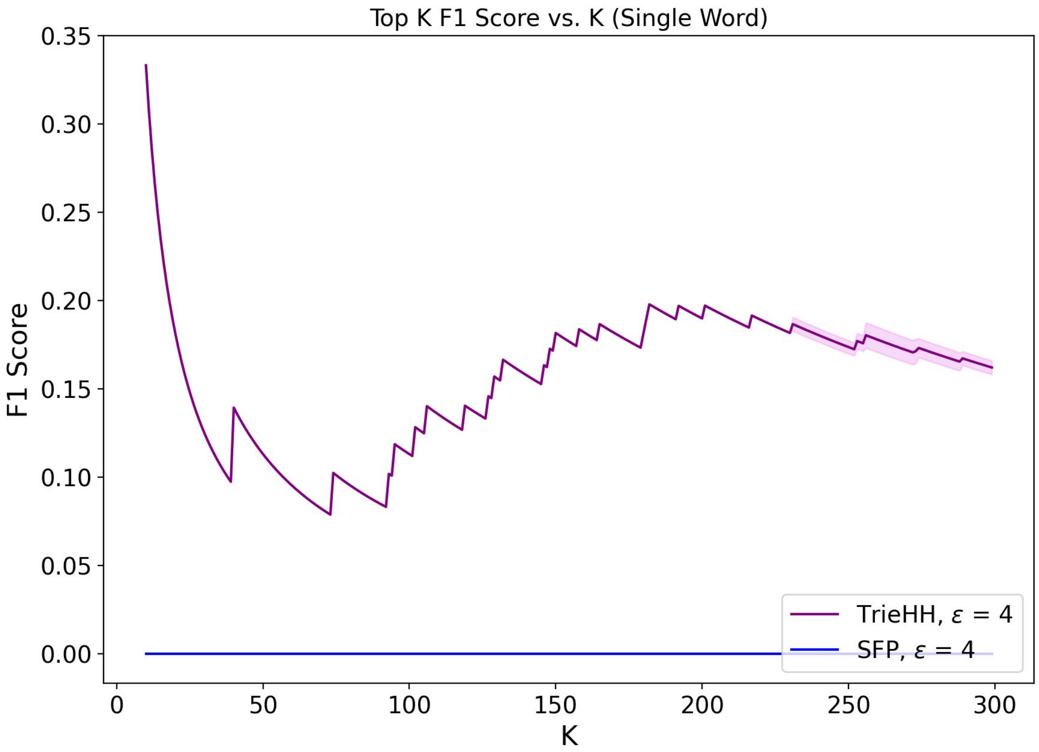
- **Data pivot** – Used a real dataset (Wikipedia clickstream) instead of purely synthetic data
  ↳ *More realistic evaluation, but reduced time for trying diverse synthetic scenarios*

- **Scope adjustment** – Focused on one privacy setting ε = 4, δ ≈ 10^(−12) due to time
  ↳ *Did not exhaustively test multiple ε values as initially intended*

- **Technical challenges** – SFP algorithm was complex to implement & debug
  ↳ *Required extra debugging (unexpected edge cases), causing slight delays*

- **Timeline constraints** – Midterm exams and other coursework slowed early progress
  ↳ *Had to catch up in later weeks; prioritized core features over "nice-to-haves"*
  (Despite changes, main goal – comparing TrieHH vs SFP under LDP – stayed on track.)

Results – Top-K F1 Score vs. K (ε=4)



Description:

- TrieHH (purple) shows a **peak** around **K = 20–30**.
- F1 score rises initially, peaks ~0.34, and **gradually decreases** as K increases.
- SFP (blue) remains **flat at F1 = 0** across all K (no heavy hitters detected).

Result Observations

TrieHH Performance:

- Achieves a **peak F1 ≈ 0.34** when K is small (~10–20 heavy hitters).
- After K ≈ 30, F1 slowly declines as more noise and false positives accumulate.
- Overall **better robustness** under ε = 4 compared to expected.

SFP Performance:

- **Consistently 0** across all K.
- Did **not recover any heavy hitters** under strict (ε=4, δ≈10^-12) settings.

Key Takeaways:

- **Interactive approach (TrieHH)** succeeds modestly despite noise.
- **Non-interactive method (SFP)** too conservative to detect true signals.

Lessons Learned

- **Privacy–Utility Trade-off** – Stronger privacy (low δ, moderate ε) **drastically** reduced accuracy
  ↳ *Our experiment highlights how challenging it is to get useful results under strict LDP*

- **Algorithm Design Matters** – Interactive methods (TrieHH) can extract signal where one-shot methods (SFP) may fail
  ↳ *Different approaches have different strengths; context (data size, noise) is key*

- **Implementing Research** – Gained hands-on experience turning complex DP algorithms from papers into working code
  ↳ *Improved our understanding of each step (and the pitfalls) of TrieHH and SFP*

- **Adaptability** – Learned to adjust our plan and scope on the fly
  ↳ *Chose realistic data over idealized tests, focused on core comparison when time was limited, and ensured we met the main objectives*

Conclusion

- Implemented two LDP heavy-hitter algorithms
- Faced and overcame practical challenges
- Found that TrieHH outperformed SFP on real data under strict privacy

Thank You !