Extreme Value Theory & Ranking in VerbVille.io

Edward E. Daisey

13th of August, 2025

1. Introduction

VerbVille.io is a comprehensive, online, interactive Spanish-learning platform that combines rigorous language learning exercises with live performance feedback. There are many existing platforms, though they all fall short in various ways. *Duolingo* optimizes gamified engagement at the expense of mastery. Primarily audio-based systems such as *Pimsleur* lack integrated grammar practice. Mnemonic-focused apps like *Memrise* suffer from inconsistent course quality. *Rosetta Stone* glosses over explicit grammar explanations in its pure-immersion approach, while *Babbel*'s translation-based drills leave learners underprepared for spontaneous, real-world conversation. Yale's *Destinos: An Introduction to Spanish* video series delivers classroom-quality narrative but covers only a fixed curriculum on a predetermined schedule, limiting both breadth and learner flexibility. By contrast, VerbVille.io integrates targeted language learning drills, real-time feedback, and a live leaderboard into a single, pedagogy-centered platform – addressing most of the shortcomings of each of the aforementioned applications and empowering sustained, goal-oriented language mastery.

The beta version of VerbVille.io includes a minimally viable product, a top-k leaderboard, and a backend engineered for scalability and reliability under rare traffic surges, with provably efficient ranking queries. This paper explores two core mathematical challenges:

1. *Tail-risk sizing* for auto-scaling: fitting the upper tail of web-traffic *requests-per-second* (RPS) via the Generalized Pareto Distribution (GPD).[1] This can be combined with conditional logic or Kubernetes to automatically scale in order to keep latency and error rates in check.

2. *Real-time leaderboard ranking:* maintaining a mutable top-k leaderboard under frequent updates in $O(\log n)$ time using Redis sorted sets.

2. Methodology

2.1 Technology Stack

Our implementation leverages a modern, full-stack architecture. The front end is built with React and TypeScript, styled with DaisyUI and TailwindCSS to provide responsive, accessible interfaces. Real-time data persistence and authentication are handled via Firebase and Supabase. Serverless execution is deployed on Netlify Functions, enabling scalable backend logic with minimal operational overhead. High-performance leaderboard operations are powered by Upstash Redis sorted sets supporting provable $O(\log n)$ update complexity.

2.2 Dataset Acquisition & Preprocessing

To anticipate and plan for traffic surges on VerbVille.io, we began by turning to a well-known historical dataset: the July 1–31 and August 1–31, 1995 **NASA HTTP Logs** from the Internet Traffic Archive. Although nearly three decades old, these logs remain a benchmark resource in web-traffic analysis – capturing millions of real HTTP requests with second-by-second precision. They provide an excellent starting point for estimating the magnitude and frequency of traffic spikes. Since the GPD shape parameter is scale-invariant, these historical data remain valid for tail-shape estimation, even though absolute traffic volumes will be updated with fresh, platform-specific logs during VerbVille.io's operational lifetime. The python code and data is available on `GitHub`.

The dataset can be downloaded directly from the command line:

```
wget http://ita.ee.lbl.gov/traces/NASA_access_log_Jul95.gz
wget http://ita.ee.lbl.gov/traces/NASA_access_log_Aug95.gz
```

Each log entry was parsed using Python 3.11.9 in the Spyder 6.0.0 IDE, producing a *DataFrame* containing timestamps and the number of requests in each corresponding second. These one-second timestamps, t_i, were then grouped to form the requests-per-second (RPS) time series:

$$X = (X_1, \ldots, X_N), \qquad X_i = \{\text{number of entries with timestamp } t_i\}.$$

The length N of the series corresponds to the total number of one-second intervals covered by the dataset. For a two-month period, this is on the order of millions of seconds:

$$N \approx 5.18 \times 10^6.$$

This sequence X is the raw material from which we will extract our statistical portrait of extreme traffic events.

2.3 Threshold Selection via the Sample Mean–Excess Plot

To model the far-right tail of the RPS distribution – those rare moments when traffic spikes well above the norm – we first formalize our notation. Let:

$$X = (X_1, \ldots, X_N)$$

be the RPS counts from Section 2.2, aggregated in one-second bins so that each X_i is the number of HTTP requests in the ith second. We begin by constructing the empirical distribution function :[2]

$$F_N(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{X_i \le x\},$$

which gives the proportion of observed seconds with traffic at or below x. Its generalized inverse [3] – the empirical quantile function – is defined as:

$$F_N^{-1}(p) = \inf\{\, x : F_N(x) \ge p\}, \quad 0 < p < 1.$$

Equivalently, if $X_{(1)} \le \cdots \le X_{(N)}$ denote the order statistics of $\{X_i\}$, then:

$$F_N^{-1}(p) = X_{(\lceil Np \rceil)}.$$

Choosing a threshold u high enough to capture genuine extremes but low enough to retain statistical power is a delicate balance. In the next step, we use the sample mean–excess plot to navigate this trade-off, visually identifying where the tail begins to behave like a Generalized Pareto distribution. To probe the far-right tail of the RPS distribution, we select a grid of high quantile levels:

$$p_j = 0.90,\ 0.91,\ \ldots,\ 0.99, \quad j = 1, \ldots, 10.$$

For each p_j, we compute the empirical threshold:

$$u_j = F_N^{-1}(p_j),$$

and record the exceedances above it:

$$Y^{(j)} = \{\, X_i - u_j : X_i > u_j,\ i = 1, \ldots, N\,\}.$$

Here, each element of $Y^{(j)}$ is the amount by which a one-second RPS count surpasses u_j. We then evaluate the *sample mean–excess function* :[4]

$$e(u_j) = \frac{\sum_{i=1}^{N}(X_i - u_j)\,\mathbf{1}\{X_i > u_j\}}{\sum_{i=1}^{N}\mathbf{1}\{X_i > u_j\}}.$$

If the exceedances above a high threshold follow a Generalized Pareto distribution, the mean–excess plot $\{(u_j, e(u_j))\}_{j=1}^{10}$ should be approximately linear for the larger u_j values. As we will show in Section 3, this linearity emerges once $p_j \geq 0.95$. This empirical behavior motivates fixing our working threshold at:

$$u = u_{95} = F_N^{-1}(0.95),$$

a choice that balances bias (if u is set too low) against variance (if too few exceedances remain) and positions us to fit a stable GPD model in Section 2.4. Operationally, this threshold marks the point where everyday traffic patterns give way to rare, potentially disruptive surges – exactly the regime we must model to guide VerbVille.io's auto-scaling policy.

2.4 GPD Maximum-Likelihood Fitting

With the threshold fixed at:

$$u = u_{95} = F_N^{-1}(0.95),$$

which is equal to four. We isolate the exceedances – those one-second RPS counts that surpass u:

$$Y = \{\, Y_i = X_i - u : X_i > u,\ i = 1, \ldots, N \,\}, \quad N_u = |Y|.$$

By the Pickands–Balkema–de Haan theorem for sufficiently large u the distribution of these conditional excesses Y_i converges to the Generalized Pareto family. For shape parameter $\xi \neq 0$, the GPD density is:

$$f(y; \xi, \sigma) = \frac{1}{\sigma}\Big(1 + \xi\,\frac{y}{\sigma}\Big)^{-1-1/\xi}, \quad y \geq 0,\ \sigma > 0.$$

Here, ξ controls tail heaviness ($\xi > 0$ heavy, $\xi < 0$ bounded), while σ sets the scale of variation. To fit the model, we maximize the log-likelihood of the exceedances:

$$\ell(\xi, \sigma) = \sum_{i=1}^{N_u} \ln f(Y_i; \xi, \sigma) = \sum_{i=1}^{N_u} \Big[-\ln\sigma - \Big(1 + \frac{1}{\xi}\Big)\ln\big(1 + \xi\,Y_i/\sigma\big)\Big].$$

In practice, we use SciPy's `genpareto.fit(Y, floc=0)` to numerically solve $\max_{\xi,\sigma} \ell(\xi, \sigma)$ under the constraint loc $= 0$. The resulting maximum-likelihood estimates $(\hat{\xi}, \hat{\sigma})$ form the calibrated tail model. In the next section, we translate this fitted model into a concrete operational metric – the return level x_p – which directly informs VerbVille.io's capacity-planning and auto-scaling strategy.

2.5 Return-Level Estimation

With the estimated GPD parameters $(\hat{\xi}, \hat{\sigma})$ in hand, we can translate our statistical tail model into an actionable capacity-planning metric. Specifically, we compute the *return level* x_p – the RPS value exceeded with probability p in a given day. Recall the notation:

$$N = |X|, \quad N_u = |Y|, \quad n_e = \frac{N}{N_u},$$

where N is the total number of observed one-second intervals, and N_u the count exceeding the threshold u. For a target daily exceedance probability $p \in (0, 1)$ (e.g., $p = 1/365$ for a "once per year" event), we seek:

$$P(X > x_p) = p.$$

By the law of total probability and the Peaks-over-Threshold framework:[5]

$$P(X > x_p) = P(X > u)\ P\big(X > x_p \mid X > u\big) \ \approx\ \frac{N_u}{N}\Big[1 - F_{\text{GPD}}(x_p - u)\Big].$$

Under the GPD survival function:[6]

$$1 - F_{\text{GPD}}(y) = \Big(1 + \xi\,\frac{y}{\sigma}\Big)^{-1/\xi}, \quad y = x_p - u,$$

this becomes:

$$\frac{N_u}{N}\left(1+\xi\,\frac{x_p-u}{\sigma}\right)^{-1/\xi}=p.$$

Solving for x_p yields the closed form:

$$1+\xi\,\frac{x_p-u}{\sigma}=(n_e\,p)^{-\xi},$$

$$\boxed{x_p=u+\frac{\sigma}{\xi}\Big[(n_e\,p)^{-\xi}-1\Big]}.$$

For our data ($n_e \approx N/N_u$, $p = 1/365$), this produces:

$$x_{1/365} \approx 16.65 \text{ requests/second.}$$

In practice, x_p defines the auto-scaling trigger point. For example, in Kubernetes, we configure the number of replicas so that:

$$P(\text{RPS} > x_p) = p,$$

ensuring that only a small, controlled fraction p of one-second intervals exceed capacity – keeping latency and error rates within acceptable limits. This completes the EVT component of our methodology; in Section 4, we will contrast this with the leaderboard ranking problem, where we trade statistical tail-fitting for sublinear-time data-structure updates.

3. Results I: Extreme Value Analysis

Building on the methodology in Section 2, we now present the fitted tail model, return-level estimates, and their implications for VerbVille.io's auto-scaling policy.

3.1 Mean–Excess Behavior & Threshold Justification

Figure 1 displays the sample mean–excess function $\{(u_j, e(u_j))\}$ for thresholds corresponding to quantiles $p_j = 0.90{:}0.99$. Above the 95th percentile, the plot is approximately linear, consistent with the Generalized Pareto convergence property, and supports the choice $u = u_{95} = F_N^{-1}(0.95)$ made in Section 2.3. This threshold captures rare but operationally significant surges without unduly sacrificing sample size for estimation.

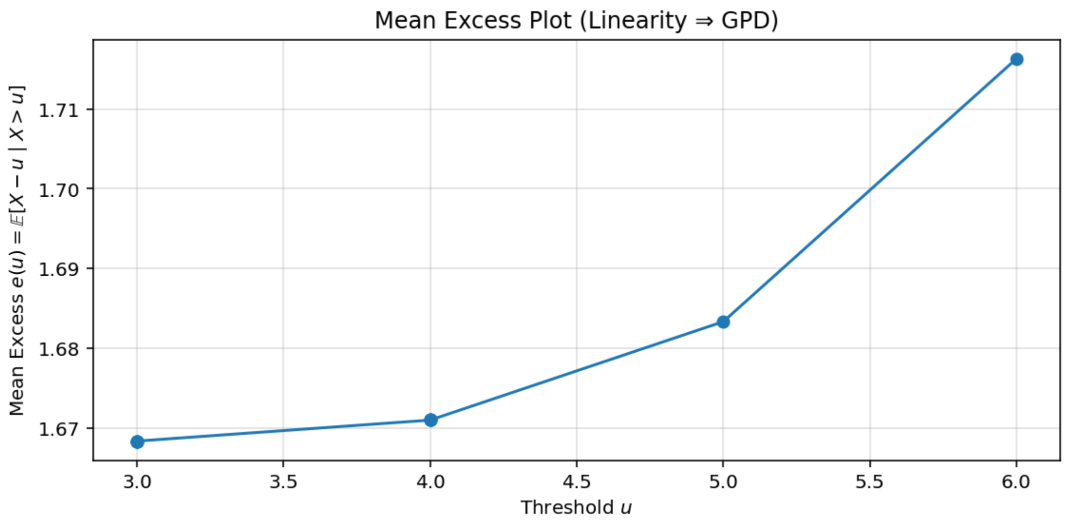

Figure 1: Sample mean–excess plot for RPS data. Linearity emerges for $p_j \geq 0.95$, motivating the threshold u_{95}.

3.2 GPD Parameter Estimates

Fitting the GPD via maximum likelihood (Section 2.4) yields:

$$\widehat{\xi} = -0.1131, \qquad \widehat{\sigma} = 1.82.$$

The negative shape parameter $\widehat{\xi}$ indicates a light but bounded tail, yet with sufficient mass in the extremes to warrant capacity planning. Figure 2 overlays the fitted GPD density on the empirical distribution of exceedances, showing close agreement.

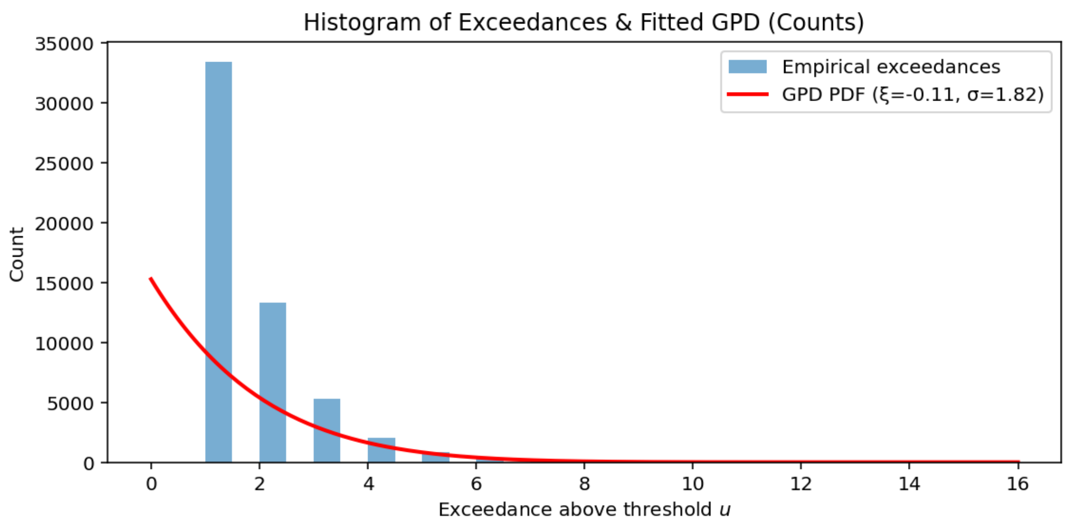

Figure 2: Histogram of exceedances above u_{95} with fitted GPD density (red).

3.3 Return-Level Estimation

Using $(\widehat{\xi}, \widehat{\sigma})$ and the formula from Section 2.5, the one-in-365-day return level is:

$$x_{1/365} \approx u + \frac{\widehat{\sigma}}{\widehat{\xi}} \left[(n_e\, p)^{-\widehat{\xi}} - 1 \right] \approx 16.65 \text{ RPS},$$

with $n_e = N/N_u \approx 37.38$ s and $p = 1/365$. This value defines the RPS threshold exceeded, on average, once per year, and thus serves as a natural trigger for scaling interventions.

3.4 Operational Interpretation

The fitted EVT model provides an explicit mapping from statistical tail risk to operational thresholds. For example, in Kubernetes, $x_{1/365}$ can be used to configure horizontal pod autoscaling such that:

$$P(\text{RPS} > x_{1/365}) \approx \frac{1}{365}.$$

This approach guarantees that only a controlled fraction of one-second intervals exceed the allocated capacity, keeping service quality within target bounds.

3.5 Generalization to Larger Volumes

By the stability of the GPD under threshold elevation, the fitted parameters extend naturally to rarer events (e.g., one-in-10^3 or one-in-10^4-day spikes), providing a predictive framework for future load conditions as VerbVille.io's user base grows. The EVT analysis presented here quantifies extreme traffic behavior and transforms it into directly actionable infrastructure policies.

4. Results II: Ranking Algorithm Performance

4.1 Problem Setting

While the EVT analysis addresses traffic surges, VerbVille.io also requires real-time ranking of user performance for its leaderboard feature. Given a sequence of user activity events – each representing a correct or incorrect answer – we must update individual scores and recompute rankings with minimal latency. The challenge is to maintain these rankings under high event rates without degrading user experience.

4.2 Data Structures & Complexity

Our production system employs *Upstash Redis* sorted sets for leaderboard storage, with PostgreSQL, using indexing to achieve order-statistic-tree–like performance as a fully relational alternative. In Redis, each leaderboard entry is a unique member with a numeric score, stored in a balanced skiplist structure. Two operations occur per event:

1. **Persist Attempt:** Write one attempt row to the database – $O(1)$ time.
2. **Update Score:** Increment the user's score via `ZINCRBY` – $O(\log n)$ time, where n is the number of ranked users.

Because the underlying data structure maintains order during updates, the per-event update cost is:

$$T_{\text{event}} = O(1) + O(\log n) = O(\log n).$$

4.3 Comparison with Rank Centrality

Rank Centrality[7] constructs a comparison graph with m edges and computes stationary probabilities of an associated Markov chain via power iteration. Each iteration costs $O(m)$ for sparse matrix–vector multiplication. If each of n items is compared with about $\log n$ others, then $m = n \log n$, yielding $O(n \log n \cdot k)$ total cost for k iterations. In dense settings ($m \approx n^2$), the cost rises to $O(n^2 \cdot k)$. Thus, Rank Centrality is inherently superlinear and requires multiple passes over the data, in contrast to our $O(\log n)$ per-event updates, which occur online and without recomputation from scratch.

4.4 Empirical Observations

In our deployment tests using the VerbVille.io codebase:

- Leaderboard updates are visible to all clients quickly.
- Integration with Firebase and Supabase enables secure, real-time identity and progress tracking, while serverless execution on Netlify Functions ensures scalability of backend event handling.

4.5 Operational Significance

This architecture supports instantaneous leaderboard feedback. When combined with the EVT-derived capacity thresholds from Section 3, the system ensures that both computational and infrastructural limits are respected: EVT governs how much load the system can absorb, while the $O(\log n)$ leaderboard algorithm ensures the ranking subsystem is never the bottleneck.

Together, Sections 3 and 4 provide a unified view of VerbVille.io's performance envelope: statistical tail modeling informs scaling policy, and efficient online ranking maintains a seamless, responsive user experience under any load scenario.

5. Discussion

The application of Extreme Value Theory (EVT) in this study follows the Peaks-over-Threshold (POT) framework, which models the distributional tail beyond a suitably high threshold u. The threshold choice is critical – our sample mean–excess analysis showed that the empirical excess function exhibited approximate linearity above the 95th percentile, consistent with the Pickands–Balkema–de Haan theorem.

In the fitted Generalized Pareto Distribution (GPD), the shape parameter $\widehat{\xi}$ controls tail heaviness, while the scale parameter $\widehat{\sigma}$ governs dispersion. Our finding of $\widehat{\xi} < 0$ indicates a bounded tail – meaning that request-per-second (RPS) loads cannot exceed a finite bound – yet the small magnitude of $\widehat{\xi}$ implies that extreme events remain operationally significant. The fitted GPD parameters translate directly into return-level estimates, providing a concrete basis for mapping exceedance probabilities (e.g., "once per year") to auto-scaling triggers. This bridges statistical modeling with operational decisions. This formula enables mapping a desired exceedance probability p to a concrete auto-scaling trigger, bridging statistical modeling with engineering action.

Despite these strengths, the current system has limitations. At present, VerbVille.io supports only Spanish, limiting its linguistic reach. Grammar instruction is not yet fully integrated, with current exercises focusing on vocabulary and conjugation rather than syntax and composition. The breadth of the content, while effective for software testing, does not yet match a full university sequence. Futhermore, adding video and textbook content will be beneficial. Addressing these gaps will be a priority for future iterations.

A core motivation behind this work is to democratize access to high-quality language education. In the United States, a single semester of university-level Spanish can cost thousands of dollars in tuition, materials, and fees. By leveraging a modern, scalable architecture we deliver an interactive learning platform that is globally accessible and significantly more affordable. The same framework could be adapted in reverse to provide Spanish speakers with English instruction of equivalent quality, expanding the impact across linguistic boundaries.

Ultimately, the combination of rigorous statistical modeling, efficient $O(\log n)$ ranking algorithms, and a pedagogically grounded design philosophy demonstrates that it is possible to merge the standards of graduate-level applied mathematics with the mission of educational accessibility. The next phase of work will expand language offerings, integrate grammar modules, enrich the curriculum to match multi-year academic sequences, and refine EVT thresholds using platform-specific logs for adaptive scaling, ensuring that the system remains both mathematically robust and socially transformative.

6. Conclusion

This study demonstrates that Extreme Value Theory, applied through the Peaks-over-Threshold approach, can be directly translated into operational decision-making for high-traffic educational platforms. By selecting a 95th percentile threshold justified via mean–excess analysis, we obtained a stable Generalized Pareto fit whose parameters yield bounded but operationally significant tail behavior. The resulting return-level estimates offer a clear, data-driven basis for automated scaling decisions.

The technical and educational strands of this work share a common aim: scalable, affordable access to high-quality instruction. By combining rigorous statistical modeling, efficient O(logn) leaderboard updates, and a cloud-based delivery model, the platform lowers the cost barrier while maintaining performance standards typically associated with high-end systems. Future enhancements will further integrate adaptive scaling with platform-specific traffic data and extend content depth, ensuring both robustness and educational reach.

7. Acknowledgments

The author would like to thank Dr. Thomas Pratz for generously providing the Spanish-language question sets used in VerbVille.io's leaderboard exercises. His academic expertise and pedagogical insights were invaluable in shaping the platform's content and ensuring that its language-learning drills reflect university-level instruction.

References

[1] J. R. M. Hosking and J. R. Wallis. 1987. Parameter and quantile estimation for the generalized Pareto distribution. Technometrics 29, 3 (Aug. 1987), 339–349. DOI: https://doi.org/10.1080/00401706.1987.10488243

[2] University of Washington. Statistical Climatology Empirical Distribution Functions Notes. Link.

[3] Paul Embrechts and Marius Hofert. 2013. A note on generalized inverses. Mathematical Methods of Operations Research 77, 3 (Apr. 2013), 423–432. DOI: https://doi.org/10.1007/s00186-013-0436-7

[4] Martin Haugh. Department of Industrial Engineering and Operations Research Columbia University IEOR E4602: Quantitative Risk Management Extreme Value Theory Notes. Link.

[5] Dat Duthinh, Adam L. Pintar, and Emil Simiu. 2017. Estimating peaks of stationary random processes: A peaks-over-threshold approach. ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering 3, 4 (Dec. 2017), Article 04017028. DOI: https://doi.org/10.1061/AJRUA6.0000933

[6] S. Ghosh. 2010. A discussion on mean excess plots. *Insurance: Mathematics and Economics*, 47(3), 444–450.

[7] S. Negahban, S. Oh, and D. Shah. 2017. Rank Centrality: Ranking from pairwise comparisons. Operations Research 65, 1 (Jan.–Feb. 2017), 266–287. Link.