Efficient Clustering with Provable Guardrails for LLM Inference at Scale

Longshaokan Wang LONGSHA@AMAZON.COM
Wai Tsang Keung WKEUNG@AMAZON.COM
Punit Ghodasara GHOPUNIT@AMAZON.COM
Roman Wang WANGRZ@AMAZON.COM
Ali Dashti DASHTIA@AMAZON.COM
Francesc Moreno-Noguer CESCMORE@AMAZON.ES
Amazon

Abstract

Scaling LLM-based applications to millions of users is bottlenecked by the inference cost and latency of modern foundation models. A natural fix is to cluster the inputs and call the LLM only on cluster representatives, letting other members inherit the output — but this is only safe if each member is measurably close to its representative. Existing clustering methods do not offer such per-sample quality control at scale: none jointly guarantee a minimal within-cluster similarity, exact matching of categorical attributes, and scalability to tens of millions of samples. We propose a two-stage algorithm that generates initial clusters with Mini-batch K-Means, then greedily selects representatives within each initial cluster — a step equivalent to the Johnson–Chvátal heuristic for Set Cover over α-balls in embedding space. The algorithm enforces the similarity and attribute guardrails exactly by construction; and runs in $O(nd + n^2d/K)$ time and $O(nd + n^2/K^2)$ memory for n samples, feature dimension d, and K initial clusters — linear in n when K grows proportionally with n. We provide benchmarks against common clustering methods on internal and public datasets — our method not only delivers per-sample guardrails but also runs 10–1000$\times$ faster and scales to data sizes where most standard methods become intractable. Deployed on 38 million customers for a persona-based recommender, the clustering method cut downstream cost and latency by 50-fold while preserving personalization and unblocked the production launch.

1. Introduction

Large Language Models (LLMs) are increasingly deployed in eCommerce applications [6, 11, 13, 16, 18, 20, 23, 32], but their inference cost and latency scale poorly to user bases of tens of millions [4, 10, 17, 19, 24, 26, 34]. A natural way to reduce this cost is to cluster inputs and invoke the downstream LLM only on cluster representatives, at the price of approximation error: all members of a cluster inherit the output computed for the representative. For customer-facing applications, this error must be explicitly controlled—irrelevant recommendations damage trust and, in the worst case, pose safety concerns (e.g., choking-hazard toys surfaced to households with young children).

We consider a production personalized-recommendation pipeline (Figure 1) in which customer shopping personas are mapped to personalized queries via an LLM, products are retrieved, and an LLM-based Marketing Critic filters results. For 38M customers, the LLM stages alone cost \$1.13M and 508 days of wall-clock time under our allocated throughput. Clustering the input personas before the pipeline is therefore essential, but must satisfy four requirements jointly:

1. semantic similarity between each sample and its cluster representative exceeds a user-specified threshold α as a quality guardrail;

© L. Wang, W.T. Keung, P. Ghodasara, R. Wang, A. Dashti & F. Moreno-Noguer.

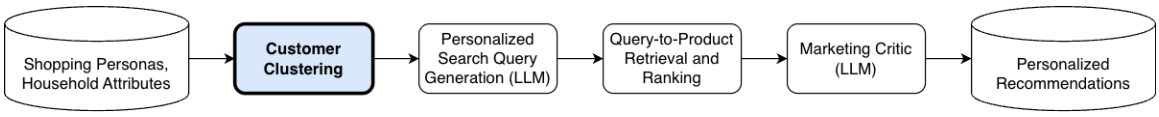

Figure 1: High-level overview of the personalized recommendation pipeline.

2. all samples in a cluster share identical user-specified categorical attributes (e.g., household composition) as another quality guardrail;
3. the number of clusters is substantially smaller than the initial sample size n (target $\geq 10\times$ reduction) for downstream computation reduction;
4. clustering runtime, memory, and cost scale to $n \sim 10^7$ and remain small relative to the downstream LLM cost saved.

To our knowledge no existing clustering method satisfies all four jointly. We propose a two-stage clustering algorithm that satisfies all four desired requirements: Stage (i) Mini-batch K-Means [27] produces initial clusters, and (ii) within each initial cluster we iteratively select the point covering the largest number of remaining samples in its α-ball under the match relation (similarity $\geq \alpha$ *and* attribute equality). Stage 2 is exactly the Johnson–Chvátal greedy heuristic for SET COVER restricted to each initial partition, which yields an exact guardrail guarantee and a provable $(1 + \ln |C_k|)$-approximation on cluster count per initial cluster C_k. The resulting cluster-size distribution is heavily skewed by design, enabling aggressive tail-trimming for additional data reduction.

Contributions. (1) A simple two-stage clustering algorithm with exact similarity and attribute guardrails, and a set-cover approximation bound on data-reduction efficiency (Appendix D). (2) Complexity analysis showing $O(nd + n^2 d/K)$ time and $O(nd + n^2/K^2)$ memory, linear in n when $K = \Theta(n)$ (Appendix E). (3) Benchmarks against six standard clustering methods on internal and public data showing $10\times$–$1000\times$ speedups at matched cluster counts (Section 4). (4) A production deployment clustering 38M customers for a 50-fold reduction in downstream compute (Section 5).

2. Related Work

Standard clustering methods (K-Means [21], Gaussian Mixture [9], BIRCH [33], Spectral [29, 30]) do not accept minimal-similarity or attribute-equality constraints out of the box. Threshold-based methods such as Agglomerative Clustering [31], Community Detection [25], and SimClus [2] support a similarity threshold but require $O(n^2)$ pairwise computation and do not scale to $n \sim 10^7$. Lexical near-deduplication (MinHash [5]) cannot handle paraphrases. Closest to our work is SemDedup [1], which uses a two-stage embedding-plus-clustering design for data-efficient pretraining; we differ in (i) enforcing attribute-equality jointly with similarity; and (ii) using greedy set-cover representative selection with a reassignment step, which improves average within-cluster similarity (Table 1) and produces the skewed cluster-size distribution exploited for tail-trimming (Appendix F.1).

3. Method

3.1. Problem Formulation

Let $\mathcal{D} = \{(P_i, \mathbf{A}_i)\}_{i=1}^n$ with textual content P_i and categorical attributes $\mathbf{A}_i$. An embedding model f_{emb} maps $P_i \mapsto \mathbf{E}_i \in \mathbb{R}^d$, and f_{sim} denotes cosine similarity. We seek a cluster assignment $\widehat{f}_{\text{cluster}} : \mathcal{D} \to \{1, \ldots, n\}$ whose image is the set of representative indices, subject to the *guardrail property*

$$\widehat{f}_{\text{cluster}}(P_i, \mathbf{A}_i) = j \implies f_{\text{sim}}(\mathbf{E}_i, \mathbf{E}_j) \geq \alpha \text{ and } \mathbf{A}_i = \mathbf{A}_j. \tag{1}$$

Input : Dataset $\mathcal{D} = \{(P_i, \mathbf{A}_i)\}_{i=1}^n$, embedding function f_{emb}, similarity function f_{sim}, similarity threshold α, number of initial clusters K.

Output : Cluster assignment $\hat{f}_{\text{cluster}} : \mathcal{D} \to \{1, 2, \ldots, n\}$; data reduction $|\hat{f}_{\text{cluster}}(\mathcal{D})| < |\mathcal{D}|$; minimal similarity $\widehat{f}_{\text{cluster}}(P_i, \mathbf{A}_i) = j \Rightarrow f_{\text{sim}}(f_{\text{emb}}(P_i), f_{\text{emb}}(P_j)) \geq \alpha$; attribute matching $\widehat{f}_{\text{cluster}}(P_i, \mathbf{A}_i) = j \Rightarrow \mathbf{A}_i = \mathbf{A}_j$.

```
/* Stage 1: Initial Clustering                                          */
```
$\mathbf{E}_i \leftarrow f_{\text{emb}}(P_i)\ \forall\, i \in \{1, 2, \ldots, n\}$ `// Convert personas to embeddings`

$\mathcal{C}_{\text{init}} \leftarrow \text{MiniBatchKMeans}(\{\mathbf{E}_i\}_{i=1}^n, K)$ `// Generate initial clusters`

```
/* Stage 2: Representative Customer Selection                           */
```
for each *initial cluster* $C_k \in \mathcal{C}_{init}$ **do**

- $S_{ij} \leftarrow f_{\text{sim}}(\mathbf{E}_i, \mathbf{E}_j)\ \forall\, i, j \in C_k$ `// Compute pairwise similarities within` C_k
- $M_{ij} \leftarrow \mathbf{I}\{S_{ij} \geq \alpha \ \wedge\ \mathbf{A}_i = \mathbf{A}_j\}\ \forall\, i, j \in C_k$ `// Compute pairwise matches within` C_k
- $\mathcal{U} \leftarrow C_k$ `// Set of unmatched customers`
- $\mathcal{R} \leftarrow \emptyset$ `// Set of representative customers`
- **while** $\mathcal{U} \neq \emptyset$ **do**
  - $\mathcal{M}_i \leftarrow \{j \in \mathcal{U} : M_{ij} = 1\}\ \forall\, i \in C_k$ `// Find matches from the unmatched customers`
  - $r^* \leftarrow \arg\max_{i \in C_k} |\mathcal{M}_i|$ `// Customer with the most matches`
  - $\mathcal{R} \leftarrow \mathcal{R} \cup \{r^*\}$ `// Add to representatives`
  - $\hat{f}_{\text{cluster}}(P_i, \mathbf{A}_i) \leftarrow r^*\ \forall\, i \in \mathcal{M}_{r^*}$ `// Assign matched customers to the representative`
  - $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{M}_{r^*}$ `// Remove matched customers`
- **end**

end

return $\hat{f}_{\text{cluster}}$

Algorithm 1: Two-Stage Clustering with Greedy Representative Selection

Equivalently, defining the *match relation* $i \sim_\alpha j \iff f_{\text{sim}}(\mathbf{E}_i, \mathbf{E}_j) \geq \alpha \wedge \mathbf{A}_i = \mathbf{A}_j$ and its α-ball $\mathcal{B}_\alpha(i) = \{j : i \sim_\alpha j\}$, every sample must be assigned to a representative whose α-ball covers it. Note $\sim_\alpha$ is reflexive and symmetric but *not transitive*, which is what makes the problem non-trivial. The formulation extends to any feature vector and any similarity/distance function.

3.2. Two-Stage Algorithm

Algorithm 1 proceeds in two stages. **Stage 1** runs Mini-batch K-Means on $\{\mathbf{E}_i\}$ to produce K initial clusters $\{C_k\}$. **Stage 2** operates independently within each C_k: it computes the pairwise similarity matrix S_{ij} and match matrix $M_{ij} = \mathbb{I}\{S_{ij} \geq \alpha \wedge \mathbf{A}_i = \mathbf{A}_j\}$, then iteratively selects the point r^* with the largest number of still-unmatched matches, designates it a representative, and assigns all its matched points to it. The process repeats until every point is covered. Figure 2 (Appendix) illustrates the procedure on a toy example.

Set-cover view and guarantees. Within each C_k, Stage 2 is exactly the Johnson–Chvátal greedy heuristic for SET COVER over the candidate family $\{\mathcal{B}_\alpha(i) \cap C_k : i \in C_k\}$. Reflexivity of $\sim_\alpha$ ensures a valid cover always exists. We prove in Appendix D that Algorithm 1 satisfies the guardrail (1) exactly, and that the number of selected representatives within each initial cluster obeys $|\mathcal{R}_k| \leq \text{OPT}_k \cdot (1 + \ln |C_k|)$. Because greedy set-cover prioritizes the largest uncovered set at each step, the resulting cluster-size distribution is heavily right-skewed—an intended feature that enables tail-cluster trimming for further data reduction (Appendix F.1).

Table 1: Within-cluster similarity and runtime at matched cluster count. Min. Sim. is the worst-case sample-to-representative similarity—baselines violate the guardrail ($< \alpha$) for 3–21% of samples, while Algorithm 1 is exact by construction.

Method	Shopping Personas				AG News				Cosmopedia			
	Avg. Sim.	Min. Sim.	Perc. Below Thresh.	Time (sec.)	Avg. Sim.	Min. Sim.	Perc. Below Thresh.	Time (sec.)	Avg. Sim.	Min. Sim.	Perc. Below Thresh.	Time (sec.)
Proposed	0.802	**0.750**	**0.0%**	**2.7**	0.413	**0.300**	**0.0%**	**4.2**	0.512	**0.400**	**0.0%**	**6.8**
Proposed w/ Reassign.	0.825	**0.750**	**0.0%**	**2.7**	0.490	**0.300**	**0.0%**	**4.2**	0.588	**0.400**	**0.0%**	**6.8**
Mini-batch K-Means	0.819	0.554	6.1%	44	0.553	0.006	5.7%	33	0.619	-0.005	4.3%	91
K-Means	0.828	0.600	2.9%	154	0.561	-0.038	4.9%	136	**0.630**	0.133	2.9%	296
Agglomerative	0.812	0.542	10.3%	1778	0.543	-0.108	9.6%	2854	0.608	0.007	7.0%	2851
BIRCH	0.799	0.558	14.0%	28	0.536	-0.068	9.2%	878	0.603	-0.001	7.0%	1337
Spectral	0.807	0.445	11.3%	6281	0.505	-0.186	21.2%	1221	0.592	-0.077	13.9%	1804
Gaussian Mixture	**0.846**	0.618	0.8%	5548	**0.562**	0.006	4.8%	1898	**0.630**	0.090	3.0%	4254

Complexity. Stage 1 is $O(T_1 bKd)$ time and $O(nd)$ memory. Stage 2 is $O(d \sum_k n_k^2)$ time and $O(\max_k n_k^2)$ memory when processed one initial cluster at a time. For balanced $n_k \approx n/K$, total complexity is $O(nd + n^2 d/K)$ time and $O(nd + n^2/K^2)$ memory—linear in n when $K = \Theta(n)$. Appendix E shows the full derivation and comparison. The role of K is a scalability knob: larger K reduces per-cluster quadratic cost but introduces partition noise that mildly inflates $|\mathcal{R}|$ (Table 3).

Optional reassignment. Greedy selection is order-dependent: a point may be covered early by a representative less similar than one chosen later. A post-hoc reassignment step (Algorithm 2, Appendix) reassigns each point to its most-similar feasible representative, strictly increasing average within-cluster similarity while preserving $|\mathcal{R}|$ and the guardrail (1) (Appendix D). The trade-off is a flatter cluster-size distribution, which slightly reduces tail-trimming efficiency (Appendix F.1).

4. Experiments

We benchmark on 100K-sample subsets of (i) internal customer shopping personas, (ii) AG News [8, 12], and (iii) Cosmopedia [3]. The 100K size is chosen because several baselines (Agglomerative, Spectral) become intractable beyond this size. Full setup in Appendix G.

4.1. Within-Cluster Similarity and Runtime

To compare fairly against baselines that do not accept an α constraint, we fix the *number* of clusters: we first run our algorithm with α chosen to yield $\sim 50\times$ data reduction (thresholds 0.75/0.3/0.4 for the three datasets), then run each baseline with the matching cluster count. Table 1 reports the average and minimum within-cluster similarity, fraction of samples below α, and runtime on a single `r7i.12xlarge` without parallelization.

Two observations: (i) every baseline violates the guardrail on a non-trivial fraction of samples (Min. Sim. well below α), while our method holds it exactly; (ii) our method is $10\times$–$1000\times$ faster because the $O(n^2)$ similarity computation is restricted to within each initial cluster of size $\approx n/K$, not to the full dataset. Notably, running Mini-batch K-Means alone to produce all final clusters is $> 10\times$ slower than our two-stage approach at the same cluster count, since running K-Means with $K = |\mathcal{R}|$ is dominated by centroid updates across many clusters.

4.2. Cluster-Size Skew and Tail Trimming

The set-cover greedy selects the largest uncovered set first, producing a heavily right-skewed cluster-size distribution (Figure 3, Appendix F). This is a practical advantage: the top 4% of clusters on the internal data cover 90% of customers, so trimming the tail yields an additional $25\times$ data reduction at the cost of dropping a small fraction of users (who can be compensated for by oversampling upstream). Baselines, which target balanced clusters, achieve at most $\sim 30\%$ coverage at the same 4% threshold. The reassignment variant (Section 3.2) partially flattens this distribution, trading tail-trimming efficiency for higher average similarity.

4.3. Similarity as a Proxy for Recommendation Relevance

The guardrail (1) is only useful if within-cluster similarity correlates with downstream recommendation relevance. We stratified-sample 7,000 member–representative pairs across 7 similarity buckets (0.2–0.9), generate 15 recommendations per representative, and have an LLM-based Marketing Critic rate each (member, product) pair on a 1–5 relevance scale. The Pearson/Spearman correlations between pair similarity and mean relevance are $0.781/0.797$ ($p < 0.001$; scatter in Figure 5, Appendix), empirically validating embedding similarity as a proxy for recommendation relevance.

5. Production Deployment

We deployed Algorithm 1 in a June 2025 A/B test of the recommendation pipeline in Figure 1, targeting 38M customers with $\alpha = 0.77$ and exact matching on adult gender, adult count, child presence, child gender, and child age. The threshold $\alpha = 0.77$ was chosen via offline validation to balance recommendation relevance against the throughput ceiling imposed by our allocated LLM budget; the attribute-matching constraints reflect product-safety considerations (e.g., age-appropriateness) not captured by embedding similarity alone. Clustering achieves $\sim 50\times$ data reduction, shrinking the LLM query-generation step from \$114,000/22.8 days (cost/time) to \$2,100/0.4 days and the Marketing Critic step from \$1,018,500/485 days to \$20,370/9.7 days—an aggregate $\sim$50-fold reduction in downstream LLM compute and wall-clock time.

To validate that clustering preserves recommendation quality at scale, we applied the Marketing Critic to 5,000 representative–member pairs: the product-to-member relevance rate is only 0.7% below the product-to-representative rate, confirming that the α-guardrail translates to minimal end-quality loss. This gap is small relative to the $\sim 50\times$ compute and latency savings downstream. The A/B test showed statistically significant positive impact on business metrics and enabled subsequent launches built on the same clustering infrastructure.

6. Conclusion

We presented a two-stage clustering algorithm that reframes the LLM-inference-scaling problem as greedy set cover over α-balls in embedding space, yielding exact within-cluster similarity and attribute guardrails, a $(1 + \ln |C_k|)$ approximation on cluster count, and linear-in-n runtime for $K = \Theta(n)$. Empirically it is 10–$1000\times$ faster than standard baselines at matched cluster counts, and in production it enabled a 50-fold downstream computation reduction for a 38M-customer recommender system. Future directions include richer similarity metrics that better predict LLM output agreement, and learned representatives that minimize cluster count under the guardrail.

Acknowledgements

The paper focuses on the novel clustering method and its application, while the persona-based recommender system (Section 5) involves many contributors and multi-team collaborations including but not limited to our engineering partners Nicholas Rahardja, Sri Venkata Vivek Dhulipala, Minh Le, and Daniel Di Pasquale.

References

[1] Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S. Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication, 2023. URL https://arxiv.org/abs/2303.09540.

[2] Mohammad Al Hasan, Saeed Salem, and Mohammed J. Zaki. SimClus: an effective algorithm for clustering with a lower bound on similarity. *Knowledge and Information Systems*, 28(3): 665–685, 2011. doi: 10.1007/s10115-010-0360-6.

[3] Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Cosmopedia, February 2024. URL https://huggingface.co/datasets/HuggingFaceTB/cosmopedia.

[4] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022. URL https://arxiv.org/abs/2108.07258.

[5] A.Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29, 1997. doi: 10.1109/SEQUEN.1997.666900.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[7] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. doi: 10.1287/moor.4.3.233.

[8] Gianna M. Del Corso, Antonio Gulli, and Francesco Romani. Ranking a stream of news. In *Proceedings of the 14th International World Wide Web Conference*, pages 97–106, Chiba, Japan, 2005.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1): 1–38, 1977. ISSN 00359246. URL http://www.jstor.org/stable/2984875.

[10] Ege Erdil. Inference economics of language models, 2025. URL https://arxiv.org/abs/2506.04645.

[11] Chenhao Fang, Xiaohan Li, Zezhong Fan, Jianpeng Xu, Kaushiki Nag, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Llm-ensemble: Optimal large language model ensemble method for e-commerce product attribute value extraction. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pages 2910–2914, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3661357. URL https://doi.org/10.1145/3626772.3661357.

[12] Antonio Gulli. The anatomy of a news search engine. In *Proceedings of the 14th International World Wide Web Conference*, pages 880–881, Chiba, Japan, 2005.

[13] Christian Herold, Michael Kozielski, Leonid Ekimov, Pavel Petrushkov, Pierre-Yves Vandenbussche, and Shahram Khadivi. Lilium: ebay's large language models for e-commerce, 2024. URL https://arxiv.org/abs/2406.12023.

[14] Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[15] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974. doi: 10.1016/S0022-0000(74)80044-9.

[16] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference*

on Neural Information Processing Systems, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

[17] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pages 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

[18] Yangning Li, Shirong Ma, Xiaobin Wang, Shen Huang, Chengyue Jiang, Hai-Tao Zheng, Pengjun Xie, Fei Huang, and Yong Jiang. Ecomgpt: instruction-tuning large language models with chain-of-task tasks for e-commerce. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29820. URL https://doi.org/10.1609/aaai.v38i17.29820.

[19] Yueying Li, Jim Dai, and Tianyi Peng. Throughput-optimal scheduling algorithms for llm inference and ai agents, 2025. URL https://arxiv.org/abs/2504.07347.

[20] Chen Luo, Dimitri Papadimitriou, Hariharan Muralidharan, Dhineshkumar Ramasubbu, Aakash Kolekar, Wenju Xu, Cong Xu, Anirudh Srinivasan, Mukesh Jain, and Qi He. Language model alignment for conversational shopping at amazon. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pages 4314–4318, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715921. doi: 10.1145/3726302.3731955. URL https://doi.org/10.1145/3726302.3731955.

[21] J MacQueen. Multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. Large language model based long-tail query rewriting in taobao search. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, pages 20–28, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701726. doi: 10.1145/3589335.3648298. URL https://doi.org/10.1145/3589335.3648298.

[24] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *ArXiv*, abs/2211.05102, 2022. URL https://api.semanticscholar.org/CorpusID:253420623.

[25] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

[26] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE, 2023.

[27] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1177–1178, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605587998. doi: 10.1145/1772690.1772862. URL https://doi.org/10.1145/1772690.1772862.

[28] Matthias Seeger, David Salinas, and Valentin Flunkert. Bayesian intermittent demand forecasting for large inventories. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4653–4661, 2016.

[29] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[30] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[31] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963. URL https://api.semanticscholar.org/CorpusID:32863022.

[32] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

[33] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 103–114. ACM, 1996. doi: 10.1145/233269.233324.

[34] Kan Zhu, Yufei Gao, Yilong Zhao, Liangyu Zhao, Gefei Zuo, Yile Gu, Dedong Xie, Tian Tang, Qinyu Xu, Zihao Ye, Keisuke Kamahori, Chien-Yu Lin, Ziren Wang, Stephanie Wang, Arvind Krishnamurthy, and Baris Kasikci. Nanoflow: Towards optimal large language model serving throughput, 2025. URL https://arxiv.org/abs/2408.12757.

Appendix A. Design Considerations and Technical Details

By design, the algorithm proposed in Section 3.2 satisfies all the desired requirements listed in Section 1: Steps 4-5 ensure that any customer and their matched representative customer will have a

persona similarity above the user-specified threshold and share the same user-specified attributes. The initial clustering allows the more expensive pairwise similarity computation to be performed within an initial cluster instead of for all the data. The embedding model, the initial clustering with Mini-batch K-Means, and the representative customer selection are all computationally cheap, especially compared to LLMs. Hence the clustering algorithm can effectively scale to large datasets (38 million samples in our recommendation use case described in Section 5). Section 4.1 shows the speed advantage of the proposed algorithm against various benchmark methods. Section F.3 shows that even with a relatively high minimal similarity requirement of 0.75, we can still achieve 186-fold reduction in data size.

We perform clustering on the embeddings of the customer personas to capture their semantic similarity, a better approach than near-deduplication approaches based on lexical similarity like MinHash [5], as lexical similarity cannot handle paraphrases and does not always align with semantic similarity. SentenceTransformer [25] provides a variety of embedding models and their performance and latency benchmarks to help with model selection. Our first selection criterion is for the output embeddings to support both Euclidean distance (used during initial clustering with Mini-batch K-means) and Cosine similarity (used during representative customer selection). The next criterion is for the embedding model to perform well across multiple benchmark datasets while having a reasonable latency. Based on these criteria, we select "all-MiniLM-L6-v2" as the embedding model for the recommendation use case, but other embedding models that satisfy these criteria can be used as well.

The initial clustering does not need to be very precise, as the subsequent representative customer selection will further partition the initial clusters. Hence we choose a cheap and fast clustering method like Mini-batch K-Means for the initial clustering. Mini-batch K-Means is significantly faster and more memory-efficient than standard K-Means while achieving similar results [27]. Similarly, due to the subsequent further partition, our clustering algorithm is relatively robust to the hyperparameters of Mini-batch K-Means or the number of initial clusters as shown in Appendix F.4.

Appendix B. Clustering Algorithm Illustration

See Figure 2 for an illustration of the proposed clustering algorithm.

Appendix C. Pseudo-Code of the Reassignment Step

Refer to Algorithm 2 for the optional reassignment step that improves average within-cluster similarity while preserving the minimal similarity and attribute matching guardrails.

Appendix D. Formal Guarantees of the Proposed Algorithm

In this section we prove that both Algorithm 1 (the base two-stage algorithm) and Algorithm 2 (the optional reassignment variant) satisfy the minimal similarity and attribute matching requirements stated in Section 1 (Requirements 1 and 2). We formalize the argument through the lens of *set cover*, which naturally captures the representative-selection step.

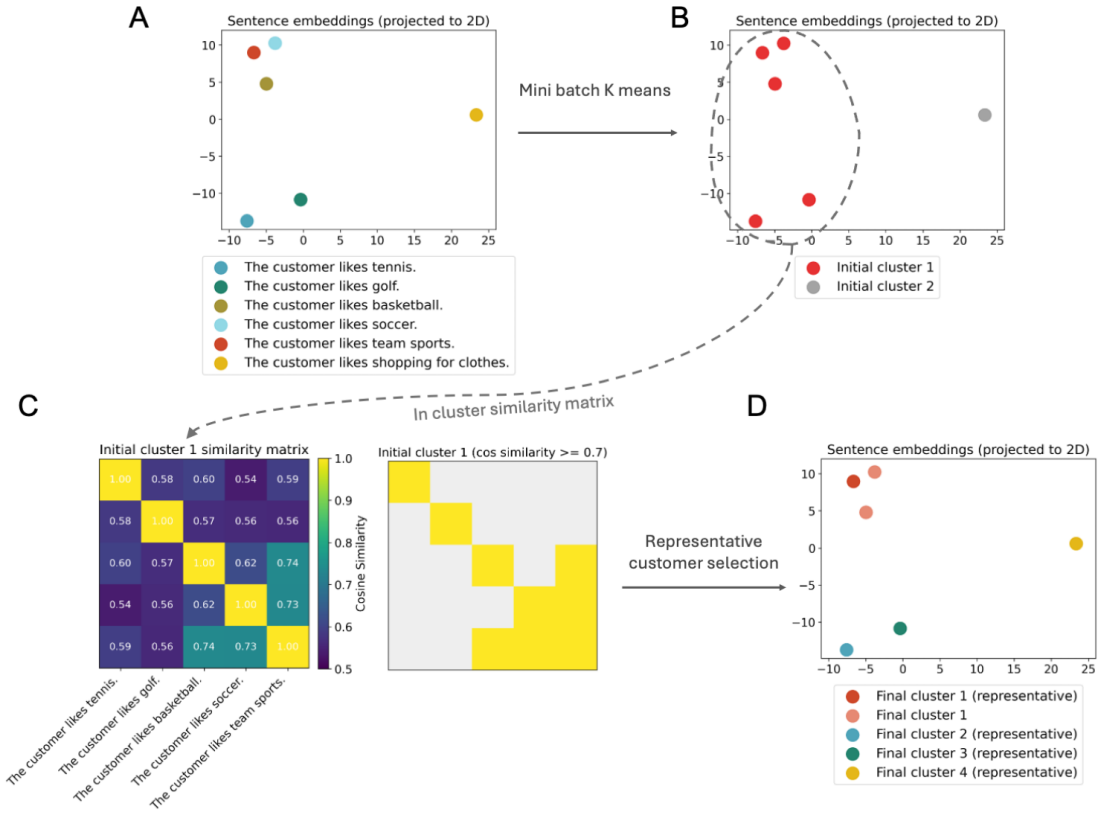

Figure 2: A. Visualization of the embeddings of six sentences projected to a 2D plot. B. The six embeddings are clustered into two initial clusters by Mini-batch K-means. C. Cosine similarity matrix of initial cluster 1 (left) and after filtering with a similarity threshold of 0.7 (right). "The customer likes team sports" has the highest number of matches and is selected as the representative. D. The final clusters.

Input : Initial cluster C_k, representative set $\mathcal{R} \subseteq C_k$, similarity matrix $\{S_{ij}\}_{i,j \in C_k}$, match matrix $\{M_{ij}\}_{i,j \in C_k}$, cluster assignment $\hat{f}_{\text{cluster}}$ from Algorithm 1.

Output : Updated cluster assignment $\hat{f}_{\text{cluster}}$ with improved average within-cluster similarity; minimal similarity and attribute matching guarantees preserved.

for each *customer* $i \in C_k$ **do**
  $r^* \leftarrow \arg\max_{r \in \mathcal{R}: M_{ir}=1} S_{ir}$ `// Find the best-matching representative`
  if r^* *exists* **then**
    $\hat{f}_{\text{cluster}}(P_i, \mathbf{A}_i) \leftarrow r^*$ `// Reassign customer to the most similar representative`
  end
end
return $\hat{f}_{\text{cluster}}$

Algorithm 2: Reassignment Step (optionally applied within Algorithm 1, Stage 2, for each initial cluster)

D.1. Preliminaries and Notation

Let $\mathcal{D} = \{(P_i, A_i)\}_{i=1}^n$ be the input dataset, with embeddings $E_i = f_{\text{emb}}(P_i)$. For a user-specified similarity threshold $\alpha \in [-1, 1]$ and attribute space, define the *match relation* $\sim_\alpha$ on $\mathcal{D}$ by

$$i \sim_\alpha j \iff f_{\text{sim}}(E_i, E_j) \geq \alpha \text{ and } A_i = A_j.$$

For any $i \in \{1, \ldots, n\}$, let the α-*ball* (or *admissible set*) of i be

$$\mathcal{B}_\alpha(i) = \{ j \in \{1, \ldots, n\} : i \sim_\alpha j \}.$$

Note that $\sim_\alpha$ is reflexive (since $f_{\text{sim}}(E_i, E_i) = 1 \geq \alpha$ for any reasonable $\alpha \leq 1$, and trivially $A_i = A_i$) and symmetric (since f_{sim} is symmetric and equality of attributes is symmetric), so $i \in \mathcal{B}_\alpha(i)$ and $j \in \mathcal{B}_\alpha(i) \iff i \in \mathcal{B}_\alpha(j)$. It is *not* transitive in general, which is why clustering under this relation is non-trivial.

A cluster assignment $\hat{f}_{\text{cluster}} : \mathcal{D} \to \{1, \ldots, n\}$ with representative set $\mathcal{R} \subseteq \{1, \ldots, n\}$ is said to satisfy the *guardrail property* if

$$\hat{f}_{\text{cluster}}(P_i, A_i) = r \implies r \in \mathcal{R} \text{ and } i \in \mathcal{B}_\alpha(r). \tag{2}$$

Equivalently, (2) says that every customer is assigned to a representative whose α-ball *covers* them. This is precisely the condition required by Requirements 1 and 2.

D.2. Set-Cover Interpretation

Within any initial cluster $C_k \subseteq \{1, \ldots, n\}$ produced in Stage 1, consider the collection of candidate cover sets

$$\mathcal{F}_k = \{ \mathcal{B}_\alpha(i) \cap C_k : i \in C_k \}.$$

Stage 2 of Algorithm 1 is exactly the classical greedy algorithm for SET COVER applied to the universe C_k with candidate sets $\mathcal{F}_k$: at each iteration it picks the set covering the largest number of still-uncovered elements, removes those elements, and repeats until the universe is exhausted. Because $\sim_\alpha$ is reflexive, every element $i \in C_k$ is contained in at least one candidate set (namely $\mathcal{B}_\alpha(i) \cap C_k$), so a valid cover always exists and the greedy procedure terminates.

D.3. Correctness of Algorithm 1

Theorem 1 (Guardrail guarantee for Algorithm 1) *For any input $\mathcal{D}$, any embedding function f_{emb}, any symmetric similarity function f_{sim} with $f_{sim}(x, x) \geq \alpha$, any threshold α, and any number of initial clusters $K \geq 1$, the assignment $\hat{f}_{cluster}$ returned by Algorithm 1 satisfies the guardrail property (2). In particular, for every $i \in \{1, \ldots, n\}$, letting $r = \hat{f}_{cluster}(P_i, A_i)$,*

$$f_{sim}\big(f_{emb}(P_i), f_{emb}(P_r)\big) \geq \alpha \qquad \textit{and} \qquad A_i = A_r.$$

Proof Fix an initial cluster C_k produced in Stage 1, and consider Stage 2 applied to C_k. Let $\mathcal{U}^{(t)}$ denote the set of unmatched customers at the start of iteration t, with $\mathcal{U}^{(1)} = C_k$.

At iteration t, the algorithm computes, for each $i \in C_k$,

$$\mathcal{M}_i^{(t)} = \{ j \in \mathcal{U}^{(t)} : M_{ij} = 1 \} = \mathcal{B}_\alpha(i) \cap \mathcal{U}^{(t)},$$

where the second equality follows from the definition $M_{ij} = \mathbf{1}\{S_{ij} \geq \alpha \text{ and } A_i = A_j\}$ in line 7 of Algorithm 1. It then selects $r^{*(t)} = \arg\max_{i \in C_k} |\mathcal{M}_i^{(t)}|$ and assigns every $j \in \mathcal{M}_{r^{*(t)}}^{(t)}$ to representative $r^{*(t)}$ (line 14). By construction,

$$j \in \mathcal{M}_{r^{*(t)}}^{(t)} \implies j \in \mathcal{B}_\alpha(r^{*(t)}),$$

so $f_{\text{sim}}(E_j, E_{r^{*(t)}}) \geq \alpha$ and $A_j = A_{r^{*(t)}}$. The set $\mathcal{M}_{r^{*(t)}}^{(t)}$ is then removed from $\mathcal{U}^{(t+1)}$ (line 15), so no customer is ever reassigned within Stage 2, and every customer assigned in iteration t satisfies (2).

Since $\sim_\alpha$ is reflexive, $r^{*(t)} \in \mathcal{M}_{r^{*(t)}}^{(t)}$, so at least one element is removed per iteration, guaranteeing termination in at most $|C_k|$ iterations with $\mathcal{U}^{(T+1)} = \emptyset$. Hence every $i \in C_k$ is assigned to some representative $r \in C_k$ with $i \in \mathcal{B}_\alpha(r)$. Because Stage 1 partitions $\{1, \ldots, n\}$ into $\{C_k\}_{k=1}^K$ and Stage 2 is applied independently within each C_k, the guarantee extends to every $i \in \{1, \ldots, n\}$. ■

Corollary 2 (Cover interpretation) *Let $\mathcal{R}_k \subseteq C_k$ be the representatives selected by Stage 2 within initial cluster C_k. Then $\{\mathcal{B}_\alpha(r) \cap C_k\}_{r \in \mathcal{R}_k}$ is a valid cover of C_k, and $\mathcal{R} = \bigcup_{k=1}^K \mathcal{R}_k$ induces the final cluster assignment. The final number of clusters is exactly $|\mathcal{R}| = \sum_{k=1}^K |\mathcal{R}_k|$.*

Remark 3 (Role of the initial clustering) *Theorem 1 holds for* any *partition $\{C_k\}_{k=1}^K$ used in Stage 1, including the trivial partition $K = 1$. The choice of initial clustering affects only the number of final clusters and the runtime/memory (Appendix B in the main paper), not the correctness of the guardrails. This is consistent with the empirical robustness reported in Table 3.*

D.4. Correctness of Algorithm 2 (Reassignment Variant)

Theorem 4 (Guardrail guarantee for Algorithm 2) *Under the assumptions of Theorem 1, the assignment $\hat{f}_{cluster}$ returned after applying Algorithm 2 to the output of Algorithm 1 also satisfies the guardrail property (2). Moreover, for every customer i, the post-reassignment similarity is at least as large as the original:*

$$f_{sim}\Big(E_i,\, E_{\hat{f}^{\text{new}}_{\text{cluster}}(P_i, A_i)}\Big) \;\geq\; f_{sim}\Big(E_i,\, E_{\hat{f}^{\text{old}}_{\text{cluster}}(P_i, A_i)}\Big).$$

where $\hat{f}^{\text{old}}_{cluster}$ and $\hat{f}^{\text{new}}_{cluster}$ denote the assignments before and after reassignment, respectively.

Proof Fix an initial cluster C_k and let $\mathcal{R}_k \subseteq C_k$ be the representatives produced by Algorithm 1. For each $i \in C_k$, Algorithm 2 (line 2) defines

$$\begin{aligned} r^* &= \text{argmax}_{r \in \mathcal{R}_k \,:\, M_{ir} = 1} S_{ir} \\ &= \text{argmax}_{r \in \mathcal{R}_k \cap \mathcal{B}_\alpha(i)} f_{\text{sim}}(E_i, E_r). \end{aligned}$$

The reassignment is performed only if the constraint set $\mathcal{R}_k \cap \mathcal{B}_\alpha(i)$ is nonempty (line 3), in which case $r^* \in \mathcal{B}_\alpha(i)$ by definition, so the guardrail property (2) is preserved.

If $\mathcal{R}_k \cap \mathcal{B}_\alpha(i)$ is empty, the assignment is left unchanged; by Theorem 1 the original assignment already satisfied the guardrail, so it still does.

However, by Theorem 1 the original representative $r^{\text{old}} = \hat{f}^{\text{old}}_{\text{cluster}}(P_i, A_i)$ belongs to $\mathcal{R}_k$ and satisfies $i \in \mathcal{B}_\alpha(r^{\text{old}})$, equivalently $r^{\text{old}} \in \mathcal{B}_\alpha(i)$ by symmetry of $\sim_\alpha$. Hence $\mathcal{R}_k \cap \mathcal{B}_\alpha(i)$ is *always*

nonempty, and the reassignment always executes. Moreover, since r^{old} is a feasible candidate in the $\arg\max$,

$$f_{\text{sim}}(E_i, E_{r^*}) \;\geq\; f_{\text{sim}}(E_i, E_{r^{\text{old}}}),$$

establishing the monotonicity claim. ■

Remark 5 (Representative set is preserved) *Algorithm 2 does not modify $\mathcal{R}_k$; it only redistributes the non-representative customers within C_k among the existing representatives. Therefore the final number of clusters $|\mathcal{R}|$ is identical for both variants, while the within-cluster similarities can only weakly increase. This explains the empirical observation in Table 1 that the "Proposed w/ Reassign." variant achieves higher average similarity at the same minimal similarity α and the same number of clusters as "Proposed Clustering".*

Remark 6 (Trade-off with tail-cluster trimming) *While reassignment preserves the guardrail and improves average similarity, it redistributes mass from the largest (earliest-selected) clusters to smaller ones, flattening the cluster-size distribution. Tail-cluster trimming (Section 4.2) therefore covers fewer customers per retained cluster under Algorithm 2, consistent with Figures 3 and 4.*

D.5. Relation to the Minimum Set Cover Problem

The greedy step in Stage 2 is the classical Johnson–Chvátal greedy heuristic for SET COVER. Let OPT_k denote the minimum number of α-balls needed to cover C_k. Then the greedy procedure yields

$$|\mathcal{R}_k| \;\leq\; \text{OPT}_k \cdot \big(1 + \ln |C_k|\big),$$

so the total number of final clusters satisfies

$$|\mathcal{R}| \;=\; \sum_{k=1}^{K} |\mathcal{R}_k| \;\leq\; \big(1 + \ln \max_k |C_k|\big) \sum_{k=1}^{K} \text{OPT}_k.$$

This provides a formal ceiling on the data reduction loss incurred by the greedy representative selection, relative to the (NP-hard) optimum cover within each initial cluster. We emphasize that this bound concerns data-reduction efficiency, not correctness: the guardrail property (2) holds exactly, not approximately.

Remark 7 (Partition-constrained vs. global optimum) *The bound above compares $|\mathcal{R}|$ to the optimum $\mathcal{R}^*_{\textit{partition}}$ subject to the Stage 1 partition $\{C_k\}$. The unconstrained optimum $\mathcal{R}^*_{\textit{global}}$ (ignoring partition boundaries) can be strictly smaller, since two similar customers split across different C_k cannot share a representative in our algorithm. The gap $|\mathcal{R}^*_{\textit{partition}}| - |\mathcal{R}^*_{\textit{global}}|$ is controlled by the quality of the initial clustering; in our setting Mini-batch K-Means is empirically sufficient to keep this gap small (Table 3 in Appendix F.4).*

Appendix E. Computational Complexity Analysis

We analyze the time and memory complexity of Algorithm 1 and Algorithm 2 as a function of the dataset size n, the embedding dimension d, the number of initial clusters K, and the attribute-vector dimension a. Let $n_k = |C_k|$ denote the size of the k-th initial cluster, with $\sum_{k=1}^{K} n_k = n$.

E.1. Stage 1: Initial Clustering with Mini-Batch K-Means

Let b be the Mini-batch K-Means batch size and T_1 the number of iterations. A standard Mini-batch K-Means update evaluates b points against K centroids in $\mathbb{R}^d$ and performs b centroid updates per iteration, yielding

$$T_{\text{Stage 1}} \;=\; O\big(T_1\, b\, K\, d\big), \qquad M_{\text{Stage 1}} \;=\; O\big((n+K)\, d\big),$$

where the memory term accounts for storing the n embeddings and K centroids. Since b and T_1 are user-specified constants independent of n, Stage 1 is effectively linear in n: $O(n\, d)$ memory and (amortized) $O(K\, d)$ time per mini-batch update. This is consistent with Sculley [27].

E.2. Stage 2: Pairwise Similarity, Match Matrix, and Greedy Selection

Stage 2 is executed independently within each initial cluster C_k. Within C_k we perform the following substeps.

(a) Pairwise similarity matrix. Computing $S_{ij} = f_{\text{sim}}(E_i, E_j)$ for all $i, j \in C_k$ costs $O(n_k^2\, d)$ time and $O(n_k^2)$ memory.

(b) Attribute match matrix. Computing $M_{ij} = \mathbf{1}\{S_{ij} \geq \alpha \text{ and } A_i = A_j\}$ costs $O(n_k^2\, a)$ time and $O(n_k^2)$ memory (assuming a-dimensional categorical attributes compared in $O(a)$). In practice one can precompute attribute group IDs in $O(n_k\, a)$ and reduce the pairwise attribute check to an $O(1)$ equality test, giving $O(n_k^2)$ total.

(c) Iterative greedy representative selection. Let T_k be the number of greedy iterations within C_k (i.e., $T_k = |\mathcal{R}_k|$). In each iteration the algorithm must find $r^* = \arg\max_{i \in C_k} |\mathcal{M}_i^{(t)}|$, where $\mathcal{M}_i^{(t)} = \{j \in \mathcal{U}^{(t)} : M_{ij} = 1\}$. A straightforward implementation maintains a row-sum vector of M restricted to $\mathcal{U}^{(t)}$:

- Initialization of row sums: $O(n_k^2)$.
- Per iteration: select r^* in $O(n_k)$, then update row sums by subtracting the columns of M indexed by $\mathcal{M}_{r^*}^{(t)}$, costing $O(|\mathcal{M}_{r^*}^{(t)}| \cdot n_k)$.

Since the sets $\mathcal{M}_{r^{*(t)}}^{(t)}$ partition C_k, $\sum_t |\mathcal{M}_{r^{*(t)}}^{(t)}| = n_k$, and the total cost of all updates across all T_k iterations is $O(n_k^2)$. Hence the greedy loop runs in $O(n_k^2)$ time.

Per-cluster Stage 2 complexity. Combining (a)–(c):

$$\begin{aligned} T_{\text{Stage 2}}(C_k) \;&=\; O\big(n_k^2\, d + n_k^2\big) \;=\; O(n_k^2\, d), \\ M_{\text{Stage 2}}(C_k) \;&=\; O(n_k^2). \end{aligned}$$

Aggregate Stage 2 complexity. Summing over initial clusters,

$$T_{\text{Stage 2}} \;=\; O\left(d \sum_{k=1}^{K} n_k^2\right), \quad M_{\text{Stage 2}} \;=\; O\left(\max_k n_k^2\right),$$

assuming Stage 2 is processed one initial cluster at a time (so that only one pairwise matrix is held in memory). If initial clusters are roughly balanced with $n_k \approx n/K$,

$$T_{\text{Stage 2}} \;=\; O\left(\tfrac{n^2 d}{K}\right), \qquad M_{\text{Stage 2}} \;=\; O\left(\tfrac{n^2}{K^2}\right).$$

This formalizes the role of K: increasing K decreases Stage 2 time *linearly* and peak memory *quadratically*, at the cost of slightly reduced data-reduction efficiency (Appendix F.4, Table 3).

E.3. Stage 2b: Reassignment Step (Algorithm 2)

Within initial cluster C_k with representatives $\mathcal{R}_k$, the reassignment loop (Algorithm 2) examines, for each $i \in C_k$, only the columns of S indexed by $\mathcal{R}_k$. The cost is $O(n_k\,|\mathcal{R}_k|)$ per cluster and

$$T_{\text{Reassign}} \;=\; O\left(\sum_{k=1}^{K} n_k\,|\mathcal{R}_k|\right) \;\le\; O\left(\sum_{k=1}^{K} n_k^2\right),$$

which is dominated by the pairwise similarity computation and does not increase the asymptotic complexity. No additional pairwise storage is needed because S and M have already been materialized in Stage 2. This matches the empirical observation that the reassignment variant has essentially identical runtime to the base algorithm (Table 1, "Time" column).

E.4. Overall Complexity

Combining Stage 1 and Stage 2,

$$T_{\text{total}} \;=\; O\left(T_1\, b\, K\, d \;+\; d\sum_{k=1}^{K} n_k^2\right),$$
$$M_{\text{total}} \;=\; O\left(n\, d \;+\; \max_k n_k^2\right).$$

For roughly balanced initial clusters with $n_k \approx n/K$,

$$T_{\text{total}} \;=\; O\Big(nd + \tfrac{n^2 d}{K}\Big), \qquad M_{\text{total}} \;=\; O\Big(nd + \tfrac{n^2}{K^2}\Big).$$

In the regime $K = \Theta(n)$ (i.e., $n_k = O(1)$), Stage 2 becomes $O(n\,d)$, matching Stage 1, and the overall algorithm is linear in n.

E.5. Comparison with Baselines

For reference, the baseline methods compared in Section 4.1 have the following standard complexities on n points in $\mathbb{R}^d$:

- **K-Means** (Lloyd): $O(nKdT)$ time, $O(nd)$ memory.
- **Mini-batch K-Means**: $O(bKdT)$ time, $O(nd)$ memory.
- **Agglomerative (Ward)**: $O(n^2 d)$ time, $O(n^2)$ memory (storing the full distance matrix or heap).
- **BIRCH**: $O(nd)$ amortized time, but with large constants.
- **Spectral Clustering**: $O(n^3)$ time in the dense case, with $O(n^2)$ memory for the affinity matrix.
- **Gaussian Mixture (EM)**: $O(nKd^2T)$ time, $O(nd + Kd^2)$ memory.

None of these baselines admit a user-specified minimal within-cluster similarity constraint, and the $O(n^2)$ memory requirement of Agglomerative and Spectral clustering is prohibitive at the $n = 38$M scale (as noted in Section 4: ~ 90 TB for $n = 5$M). Algorithm 1 circumvents this by restricting the quadratic computation to *within* each initial cluster, yielding $O(\max_k n_k^2) \ll O(n^2)$ memory for any $K \gg 1$. This is the underlying reason for the $10\times$–$1000\times$ runtime advantage observed in Table 1.

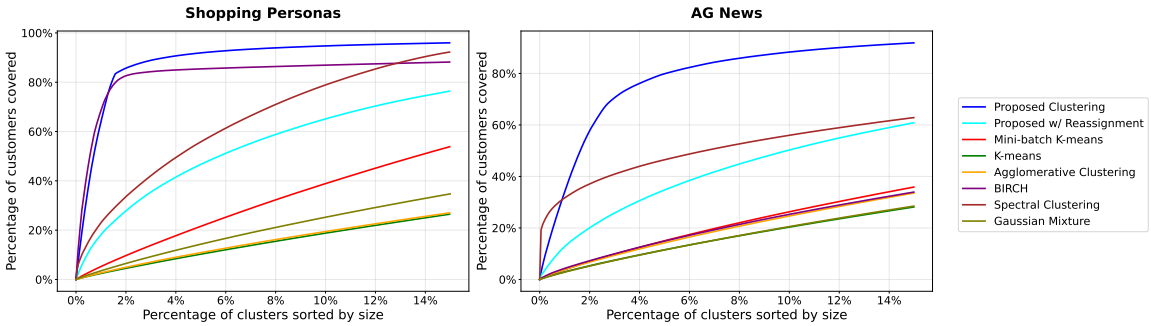

Figure 3: Sample coverage by sorted clusters.

Appendix F. Additional Experiment Results

F.1. Cluster Size Distributions

As detailed in Section 3, the iterative greedy selection of representative samples in the proposed algorithm by design leads to highly skewed cluster size distribution, with practical advantages: We can remove a large percentage of tail clusters at the cost of removing a small percentage of samples, improving data size reduction; in the case of personalized recommendation, if quality assurance is expensive, we can focus on the top clusters by size. We examine how many overall samples the top clusters by size can cover for different clustering methods in Figure 3 and Figure 4 based on the experiment output from Section 4.1.

We see that as the percentage of top clusters increases, the proposed clustering algorithm can cover significantly more samples than other clustering methods. The practical implication, for example, is that we can keep only 4% of clusters to cover 90% of customers when generating personalized recommendations for a further 25-fold reduction in downstream computation cost and latency. The customer removal can be compensated for by oversampling customers in the beginning.

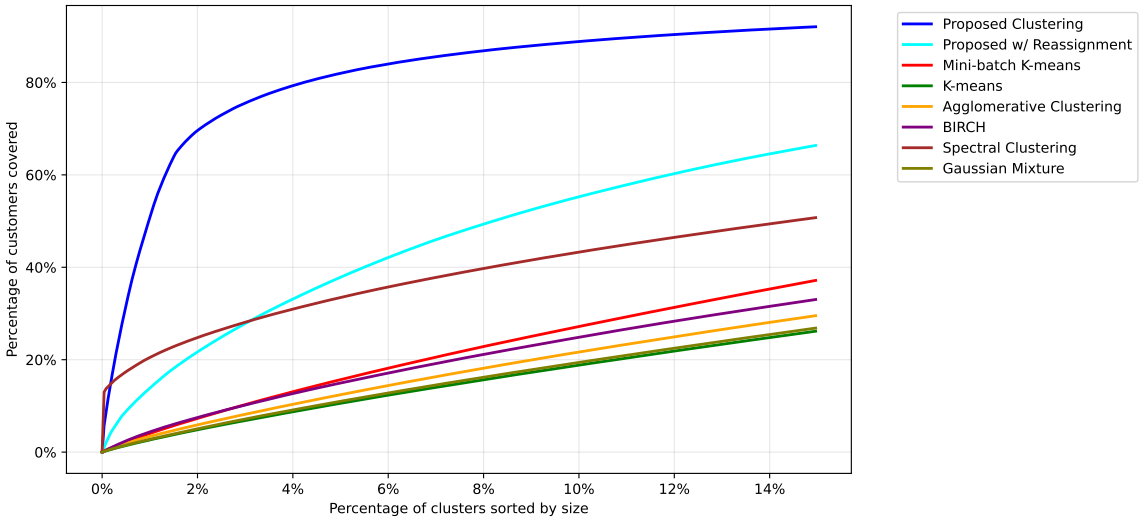

Figure 4: Sample coverage by sorted clusters on Cosmopedia.

F.2. Relevance Rating v.s. Similarity Figure

We show the scatterplot of relevance rating vs the member-representative similarity in Figure 5, empirically validating using the minimal within-cluster similarity as quality guardrails of product recommendations.

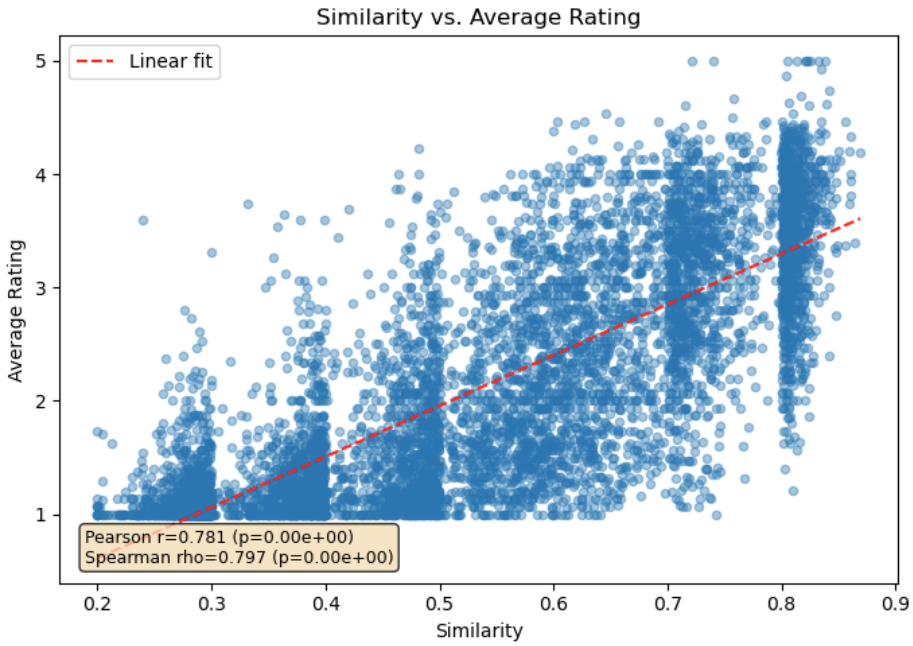

Figure 5: Recommendation relevance rating by member-representative similarity.

F.3. Impact of Minimal Similarity and Household Attributes

There is a trade-off between within-cluster similarity and the number of clusters. In the case of personalized recommendations, the higher the similarity, the more likely we recommend the relevant products to all customers in a cluster, at the cost of increasing the number of clusters and thus downstream computations. Similarly, the number of clusters will increase if we require matching household attributes in addition to the persona similarity, such as adult gender, adult count, child presence, child gender, and child age. These household attributes are important for personalized recommendations and not always captured in the customer shopping personas. We show how the number of clusters is affected by the minimal similarity and matching attributes required between each sample and the cluster representative in Table 2 on randomly selected 5 million customers of the 38 million customer dataset.

F.4. Impact of Hyperparameters and Robustness

We perform the proposed clustering with reassignment step on the 100K Shopping Personas for different hyperparameters of Mini-batch K-Means used for the initial clustering before representative selection. Because the greedy selection and optional reassignment in the second stage help further partition the initial clusters based on desired safety/quality constraints, the proposed clustering

Table 2: Clustering performance metrics with different matching attributes and similarity thresholds on 5 million customers.

Matching Attributes Required	Minimal Similarity Required	Number of Clusters	Average Similarity	Fraction of Data Retained	Data Reduction Fold
None	0.65	2344	0.794	0.05%	2133×
	0.7	5012	0.812	0.1%	998×
	0.75	26868	0.824	0.5%	186×
	0.8	230262	0.840	4.6%	22×
	0.85	1694033	0.904	33.9%	3×
	0.9	4695169	0.994	93.9%	1×
adult gender, adult count, child presence, child gender, child age	0.65	44285	0.792	0.9%	113×
	0.7	49804	0.809	1.0%	100×
	0.75	97298	0.824	1.9%	51×
	0.8	435727	0.846	8.7%	11×
	0.85	2146887	0.916	42.9%	2×
	0.9	4787855	0.996	95.8%	1×

method is robust to the hyperparameters of Mini-batch K-Means or even the choice of the clustering method used in the first stage (Table 3).

There are two main tradeoffs in our clustering application. The first main tradeoff is between cluster quality and data reduction fold, which impacts the recommendation quality and downstream computation savings. This tradeoff is controlled by the user specified minimal similarity threshold and matching attributes shown in Table 2. The second main tradeoff is between scalability (in terms of latency and memory usage) and data reduction fold. This tradeoff is controlled by the initial number of clusters, K. Recall that in the second stage of the proposed clustering we compute pairwise embedding similarity within each initial cluster for the greedy representative selection. This pairwise computation is $O(n^2)$ w.r.t. sample size n, in this case the size of each initial cluster. Thus as K increases, the latency and memory requirement will both decrease, increasing scalability. In exchange, the initial clustering also introduces noise so that samples that pass the final cluster criteria might end up separated during initial clustering, reducing the efficiency of data size reduction, which worsens as K increases (Table 3). In practice we recommend choosing a K as small as latency and memory requirements allow to maximize data reduction.

Appendix G. Experimental Setup Details

In the experiment on within-cluster similarities from Section 4.1, we specify the minimal similarity thresholds (0.75 on internal shopping personas, 0.3 on AG News, and 0.4 on Cosmopedia) to obtain $\sim$50 fold reduction in data size based on our target use case in personalized recommendation and for making results across 3 datasets more comparable. The conclusions are consistent for other values of minimal similarity thresholds based on further experiments. We set the number of initial clusters to be 0.04% of the sample size. Increasing this ratio will reduce the memory requirement of pair-wise similarity computation within each initial cluster at the potential cost of increasing the number of final clusters.

Table 3: Robustness analysis of the proposed clustering algorithm with reassignment on internal Customer Shopping Personas across different Mini-Batch K-Means (MBKM) hyperparameters: batch size, init size, and number of initial clusters. We report average within-cluster similarity (Avg. Sim.), minimal within-cluster similarity (Min. Sim.), number of final clusters (# Clusters), data reduction fold (Reduc. Fold), Stage 1 time (MBKM), and Stage 2 time (Representative Selection and Reassignment) in seconds.

MBKM Batch Size	MBKM Init Size	Num Initial Clusters (K)	Avg. Sim.	Min. Sim.	# Final Clusters	Reduc. Fold	Stage 1 Time (sec.)	Stage 2 Time (sec.)
1024	1024	10	0.826	0.750	2055	48×	0.12	12.97
1024	1024	50	0.825	0.750	2539	39×	0.22	1.48
1024	1024	250	0.824	0.750	2920	34×	0.46	1.02
1024	10240	10	0.825	0.750	2029	49×	0.18	14.59
1024	10240	50	0.825	0.750	2575	38×	0.46	1.41
1024	10240	250	0.824	0.750	3042	32×	1.57	0.90
10240	1024	10	0.826	0.750	2034	49×	0.29	13.54
10240	1024	50	0.825	0.750	2579	38×	0.29	1.44
10240	1024	250	0.824	0.750	3054	32×	0.54	0.89
10240	10240	10	0.826	0.750	2048	48×	0.21	14.89
10240	10240	50	0.825	0.750	2602	38×	0.58	1.46
10240	10240	250	0.823	0.750	3104	32×	1.69	0.86

We use the default hyperparameters of the clustering methods benchmarked against as implemented in Scikit-learn [22] 1.7.2, except for Spectral Clustering whose hyperparameters are slightly adjusted for memory efficiency. The number of clusters is set to match the final number of clusters produced by the proposed clustering algorithm on each dataset to achieve the same data size reduction. Other key hyperparameters are as follows:

1. **Mini-batch K-Means**: 'max_iter': 100, 'batch_size': 10240, 'init_size': 30720, 'random_state': 123.
2. **K-Means**: 'init': 'k-means++', 'n_init': 'auto', 'max_iter': 300, 'algorithm': 'lloyd'.
3. **Agglomerative Clustering**: 'metric': 'euclidean', 'linkage': 'ward', 'compute_full_tree': 'auto'.
4. **BIRCH**: 'threshold': 0.5, 'branching_factor': 50.
5. **Spectral Clustering**: 'affinity': 'nearest_neighbors', 'n_neighbors': 10, 'eigen_solver': 'lobpcg'.
6. **Gaussian Mixture**: 'covariance_type': 'full', 'init_params': 'kmeans', 'max_iter': 100.