Fair Document Valuation in LLM Summaries via Shapley Values

Zikun Ye *
University of Washington

Hema Yoganarasimhan
University of Washington

November 23, 2025

Abstract

Large Language Models (LLMs) are increasingly used in systems that retrieve and summarize content from multiple sources, such as search engines and AI assistants. While these systems enhance user experience through coherent summaries, they obscure the individual contributions of original content creators, raising concerns about credit attribution and compensation. We address the challenge of valuing individual documents used in LLM-generated summaries by proposing a Shapley value–based framework for fair document valuation. Although theoretically appealing, exact Shapley value computation is prohibitively expensive at scale. To improve efficiency, we develop Cluster Shapley, a simple approximation algorithm that leverages semantic similarity among documents to reduce computation while maintaining attribution accuracy. Using Amazon product review data, we empirically show that off-the-shelf Shapley approximations, such as Monte Carlo sampling and Kernel SHAP, perform suboptimally in LLM settings, whereas Cluster Shapley substantially improves the efficiency–accuracy frontier. Moreover, simple attribution rules (e.g., equal or relevance-based allocation), though computationally cheap, lead to highly unfair outcomes. Together, our findings highlight the potential of structure-aware Shapley approximations tailored to LLM summarization and offer guidance for platforms seeking scalable and fair content attribution mechanisms.

Keywords: LLMs, Shapley Value, Digital Marketing, Search System, Retrieval Augmented Generation, Revenue Management.

*We thank Yizhuo Chang and Lei Wang for outstanding research assistance. We are also grateful to the participants of the WUSTL Junior Faculty Forum in Marketing 2025, the AIM Conference 2025, the 2025 INFORMS Marketing Science Conference, the UNC marketing seminar, and the MIT DSL seminar for feedback and comments, which have significantly improved this paper. We also thank Xiao Liu and Dennis Zhang for their detailed and thoughtful comments. Please address all correspondence to: zikunye@uw.edu and hemay@uw.edu.

1 Introduction

The advent of Large Language Models (LLMs) has revolutionized how users search, process, and consume information. Today's LLM-based search and summarization engines combine the strengths of LLMs with those of traditional search engines. First, unlike traditional search engines that return a list of links in response to a query, LLM-powered search provides a concise summary with references to relevant documents/websites. This frees users from the cognitive load of manually navigating and aggregating information from multiple sources (OpenAI, 2024). Second, unlike regular LLMs that rely solely on static training data, LLM-based search engines augment their generative process with real-time retrieval and source grounding using a framework known as Retrieval-Augmented Generation (RAG) (Fan et al., 2024). In the RAG framework, the system first retrieves a set of documents/articles that are most relevant to the query and then uses them as context in the generative process. This ensures that the LLM's responses are grounded in up-to-date, relevant content and do not suffer from limitations such as non-factual hallucinations and outdated knowledge that plague off-the-shelf LLMs. Thus, LLM-based search and summarization platforms combine the generative aspects of LLMs with the retrieval aspects of search engines by augmenting generation with context/information from documents most relevant to a query.

Over the last few years, all major search engines have adopted these developments and integrated LLMs into their search infrastructures, e.g., Microsoft's Bing AI (Microsoft, 2025) and Google's AI Overview (Google, 2025). Both these tools provide a summary response to a user's search query with references or links to relevant websites for informational queries.[1] Further, new LLM-based search engines such as OpenAI's ChatGPT Search (OpenAI, 2024) and Perplexity AI (Perplexity AI, 2025) have emerged and grown rapidly in the last few years. The impact of this transformation goes beyond web search and extends to Q&A and informational websites (e.g., Reddit) and e-commerce websites (e.g., Amazon, Best Buy). For example, Amazon now shows an LLM-based summary of all the reviews for a product on the main product page. Customers can further access the source reviews the summary is built on, if needed; see Figure 1 for an example. In addition to broad overviews of the reviews, consumers can query Amazon's AI-based search tool Rufus for specific information related to the product, and it provides a response based on the information in the reviews/product page (Schermerhorn, 2023; Mehta and Chilimbi, 2024). Examples of all these applications are illustrated in Web Appendix §A.

The main advantage of LLM-based search and summarization tools is that they simplify the search process for users, who no longer have to click on multiple links and aggregate information from individual sources. This can increase customer satisfaction and lead to higher platform usage (Xu et al., 2023). For example, Zhu (2025) empirically shows that LLM-based search and summarization can significantly boost consumer purchases on an e-commerce platform through better articulation of consumer needs and product attributes. Furthermore, these approaches keep customers on the platform's own website or interface; that is, consumers don't need to leave the search engine and visit other websites to gather information. This, in turn,

[1] Search queries are typically categorized into three groups – (1) informational, where the user is seeking information on a topic, (2) navigational, where the user is simply using the search engine to navigate to a website, and (3) transactional, where the user seeks to perform some transaction, e.g., a purchase. LLM-based summaries are mainly shown for informational queries.

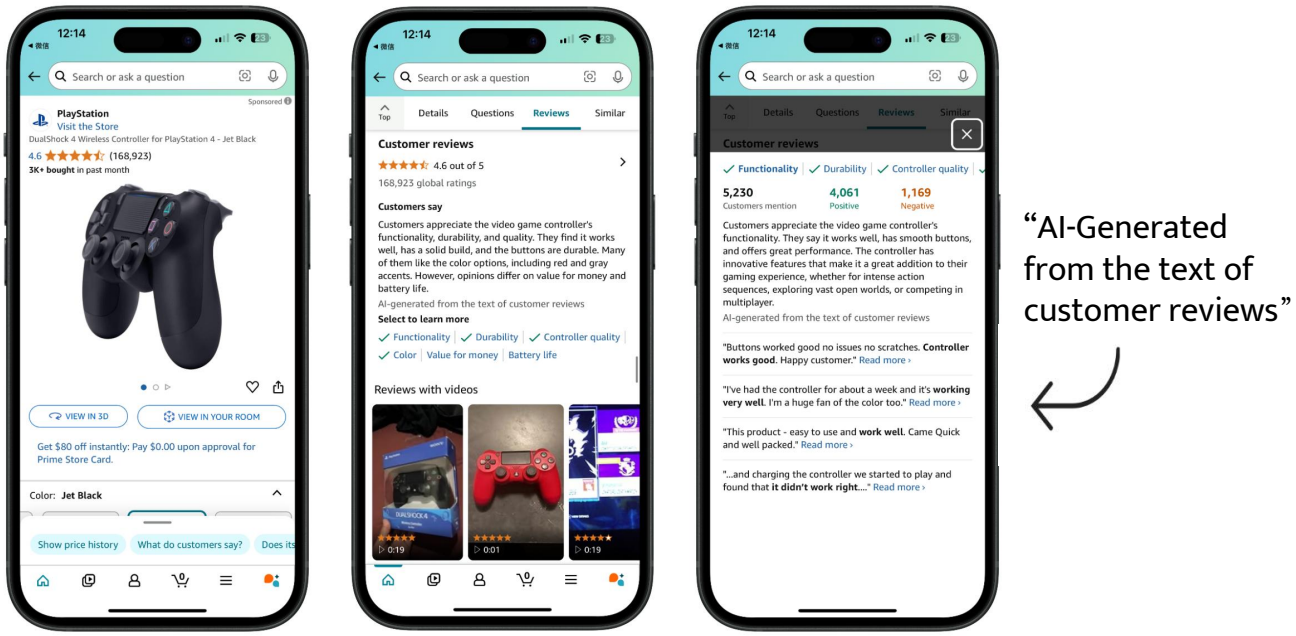


Figure 1: Amazon's AI-generated customer review for a wireless controller product (snapshot taken on Dec 12, 2024): The left image shows the wireless controller product page on Amazon. The center image displays an Amazon AI-generated summary review of this product. Users can click "Select to learn more" to focus on specific aspects of interest. The right image shows AI-generated summaries for the selected aspect, displaying the source customer reviews with key information highlighted in bold.

can lead to increased time spent on the platform, resulting in more eyeballs and potentially higher revenues (Goodwin, 2024). For these reasons, the rise of LLM-based search and summarization has generally been seen as beneficial to both platforms and consumers, at least in the short term.

However, the rise of LLM-based automated summarization tools has led to significant concerns among content creators and information providers. In a traditional search model, the source documents (and their corresponding producers) stand to gain traffic and reputation by being shown in response to search queries in organic listings.[2] In contrast, LLM-powered search delivers a synthesized answer directly in the search interface, allowing users to obtain information without visiting the source websites. For example, after the introduction of Google's AI overview, website owners have expressed reservations over the loss of traffic and revenues (Carroll, 2025; Sherrer, 2025; Chee, 2025). Similarly, on e-commerce and Q&A websites, individual users typically seek to develop a reputation for expertise and knowledge by providing meaningful answers or reviews and are often rewarded with badges, elite status, or free products in return.[3] However, if

[2]Both higher traffic and better reputation can usually be monetized through advertising and/or subscription revenues.

[3]For example, sites like Reddit use a "Karma" system which captures how helpful/unhelpful users found a user's responses (Reddit Help, 2024). Similarly, Yelp recognizes high-quality reviewers by awarding them an "Elite" badge shown publicly on their profile, which in turn gives them access to attend exclusive events in the local community as well as other perks (Chang, 2023).

consumers stop reading individual reviews/answers and rating them, the reputation systems on these platforms would cease to function (Resnick et al., 2000), and reviewers/content generators on these sites will lose the opportunity to monetize their content. Thus, the shift to LLM-based summaries threatens the revenue streams of content creators and raises concerns about the uncompensated use of their data.[4] If content creators are not fairly compensated for their content, then they will have no incentive to provide content to platforms and search engines. This can lead to degradation in the quality of AI-powered summaries and failure of platforms' reputation systems over time. Thus, while platforms may see short-term benefits from avoiding attribution or compensation to content creators and websites, this is not a sustainable equilibrium in the long term.

As a first step to address this problem, many AI firms have started making licensing deals with large content creators; e.g., recently OpenAI and Murdoch-owned empire of publications like *The Wall Street Journal* and *The New York Post* made a deal to allow OpenAI to use the content from these publications (Robertson, 2024). Other industry approaches, such as Cloudflare's recently launched marketplace (Zeff, 2025), allow websites to charge AI bots for scraping but determine payments solely by the number of scrape requests rather than the contribution of the scraped content. However, there are several challenges associated with such arrangements. First, it is often unclear how to value the body of content from different publishers and arrive at an equitable price for each publisher/contributor. Quantifying a document or publisher's contribution to a given summary is a non-trivial task – the value that a specific document adds to a summary (in response to a query) depends not only on the document's own relevance, quality, and reliability but also on the extent to which the information it contains is unique, i.e., the marginal contribution of the document in comparison to other documents. Further, it is also unclear how to aggregate a document's contributions across queries. For example, one document/publisher may be very valuable for a niche query while another may be moderately valuable for a very popular query. Thus, we need a principled way to correctly value a document's contribution to both *within* and *across* queries. Second, while licensing deals are a potential solution when there are a few large publishers, they are less likely to work when there are a large number of small niche contributors since it is too costly to contract individually with each of them.

Further, for a document valuation approach to be broadly applicable and practical, it needs to satisfy three properties: (1) be summarization procedure agnostic, (2) be evaluation process agnostic, and (3) be scalable and cost-effective. The first property is important because LLM-based summarization procedures vary across platforms (e.g., Google uses Gemini-based systems, whereas Perplexity allows the users to choose their base LLM for summarization). Further summarization techniques and LLMs continue to evolve. Thus, it is essential that any framework we develop is not specific to a single LLM or summarization technique. The second property implies that the framework should be agnostic to the procedure used to score the usefulness of the summary because evaluation methods vary across platforms. For example, search engines typically use implicit feedback, such as how long the user spends on the summary. In contrast, question-answering websites or review aggregators often have explicit feedback on the helpfulness of the summaries. Finally, the third property implies that the document valuation procedure should scale to real platforms and not be

[4]Indeed, several major publishers such as *The New York Times* have started to restrict or revoke AI access to their content, citing concerns that their contributions are being undervalued and misused without fair compensation (Grynbaum and Mac, 2023).

unreasonably costly in terms of time or money. Together, these properties help ensure that any document valuation procedure is general and scalable, allowing it to be applied across multiple business contexts.[5]

In this paper, we present a framework for *fair document valuation* in the context of LLM summaries. Our framework addresses the challenges discussed above and satisfies the three properties noted earlier. Our solution concept builds on the Shapley value framework from cooperative game theory (Shapley, 1953). First, given a query q and an LLM summary based on a specific set of documents, we use the Shapley value framework to distribute the total value generated by the summary (measured by its quality or usefulness) across this set of documents. Shapley values quantify how much each document contributes to the overall summary while accounting for interactions between documents, such as redundancy and overlapping information. Formally, Shapley values ensure fair attribution through four properties: Efficiency (the total value is fully distributed across all documents), Symmetry (treat equally contributing documents identically), Null Document (assign zero value to non-contributing documents), and Linearity (additive consistency across multiple evaluations). Note that this framework supports value attribution at the query level, and the linearity property enables aggregation of values across queries. This flexibility is crucial for revenue attribution, as it accommodates both per-query revenues (e.g., advertising or API calls) and aggregated subscription-based revenues. As such, this framework addresses both the challenges discussed above. Moreover, because Shapley value computation treats the summarization and evaluation procedures as black boxes, it remains compatible with any LLM summarization method or evaluation metric, satisfying the first two design requirements introduced earlier.

However, a major hurdle in applying Shapley values to LLM summaries is its exponential computational complexity – if we have S_q relevant documents for query q, the number of summarizations and evaluations needed to calculate Shapley values is $2^{|S_q|} - 1$. Even for modest-scale platforms that process millions of queries, this becomes computationally infeasible in practice. To address this problem, many approximation algorithms have been proposed, such as Monte Carlo (Mann and Shapley, 1960), Truncated Monte Carlo (Ghorbani and Zou, 2019), and Kernel SHAP (Lundberg and Lee, 2017). However, all these algorithms treat each document as independent entities and do not leverage the textual information of documents. As a result, their empirical performance in LLM summarization settings is often suboptimal.

In this paper, we develop a simple yet effective approximation algorithm, Cluster Shapley, that leverages textual information to improve computational efficiency. The intuition is straightforward: documents with similar content are likely to make comparable contributions to the final summary and therefore should receive similar Shapley values. Instead of treating documents as independent, the algorithm uses LLM-generated embeddings to group semantically similar documents into clusters. Each cluster is then treated as a single meta-document in the Shapley calculation, and the resulting cluster-level values are distributed evenly across the documents within each cluster. This reduces the number of combinations the LLM must summarize and

[5]Note that many simplistic approaches for document valuation can be used, e.g., using the word count of the document, valuation based on a document's presence/absence in a summary, copyright or link-based attribution, or leave-one-out method suffer from significant disadvantages (Ghorbani and Zou, 2019). However, these methods are unable to properly account for the uniqueness of information, marginal/incremental value, and actual contribution of a document to the summary.

evaluate, providing efficiency gains with limited loss in accuracy. A tunable hyperparameter-the clustering diameter ϵ-controls the balance between accuracy and cost: smaller values of ϵ yield more accurate but slower computations, whereas larger values offer faster, coarser approximations.

We establish a set of theoretical guarantees for our proposed Cluster Shapley algorithm that balance accuracy and computational efficiency. Under a mild Lipschitz continuity assumption in the embedding space, we prove that the approximation error for each document's Shapley value is bounded by $L\epsilon$, where L is the Lipschitz constant, and the error converges to zero as clusters become finer, i.e., the clustering diameter ϵ goes to zero. The algorithm reduces the exponential complexity of exact Shapley computation-from $O(2^n)$ to $O(n^2 + 2^m)$, where $n = |S_q|$ is the number of relevant documents and m is the number of clusters-yielding substantial computational savings when documents exhibit semantic redundancy. Next, we consider settings where m remains large even after the clustering process – usually because the number of relevant documents n is too large to begin with and/or because the set of relevant documents is disparate, and cannot be represented by a small number of clusters. For such settings, we extend our method by incorporating approximation algorithms (e.g., Monte Carlo) at the cluster level. We derive high-probability error bounds that decompose the total approximation error into clustering and cluster-level approximation error. For example, using Monte Carlo sampling for cluster-level Shapley estimation further reduces the total complexity further reduces the computation complexity to $O(n^2 + m\epsilon^{-2})$ while maintaining the same-order approximation error. Together, these results demonstrate that Cluster Shapley is a principled and scalable solution for fair document valuation in LLM-generated summaries, enabling efficient deployment even in large-scale settings.

To empirically demonstrate the performance of our algorithm, we apply it to a dataset of products drawn from the popular Amazon product review dataset (Hou et al., 2024). This dataset closely aligns with real-world applications, as Amazon's AI-generated review summaries are built on the same repository. We choose 24 products from diverse categories and construct two queries for each product based on frequently mentioned attributes on Amazon, mimicking the information needs of real consumers. For each query, the system summarizes relevant reviews using the RAG framework: the top eight most relevant reviews are retrieved using embedding similarity (between the query and the review), and the LLM generates a query-specific summary based on this retrieved set.

We benchmark the proposed Cluster Shapley algorithm against two simple attribution rules-equal attribution and relevance-weighted attribution-and three widely used approximation methods: Monte Carlo (Mann and Shapley, 1960), Truncated Monte Carlo (Ghorbani and Zou, 2019), and Kernel SHAP (Lundberg and Lee, 2017). The results yield two main insights. (1) Off-the-shelf approximation algorithms are empirically far from optimal in the context of LLM-based summarization. These algorithms, originally designed for settings with independent contributors, fail to exploit the rich semantic structure of textual data and therefore deliver poor efficiency–accuracy trade-offs. In contrast, Cluster Shapley substantially improves the efficient frontier: for instance, our method requires only 20–40 unique permutations (out of 255 total) to reach an MAE of 0.20, whereas other approximation methods need more than 110 permutations to achieve comparable accuracy. This finding underscores the importance of developing Shapley approximation algorithms specifically tailored

for LLM summarization-leveraging embedding-based similarity and other contextual structures-and points to a promising direction for future research. (2) Simple attribution rules are highly biased, raising fairness concerns in practical applications. Equal attribution, despite its negligible computational cost, results in an MAPE exceeding 150%, and even relevance-weighted attribution remains above 30%. These large deviations from the fair Shapley baseline demonstrate that simplistic heuristics, though computationally cheap, can substantially distort individual document contributions. Together, these findings highlight that more effective, structure-aware Shapley approximations are both feasible and necessary: they enable fairer value attribution with affordable computational cost.

We also conduct a series of robustness checks and explore extensions to demonstrate the stability and versatility of our algorithm. Our main numerical results remain consistent under a variety of conditions, including alternative evaluation LLMs (e.g., Claude instead of GPT), different error metrics, and different sample splits. We further show that explicitly enforcing intra-cluster similarity leads to improved performance compared to off-the-shelf clustering methods. Finally, we show that our approach easily scales to settings where the platform uses a larger number of documents in the summarization. While standard Shapley is computationally infeasible for large n settings, the version of our Cluster Shapley that incorporates Monte Carlo sampling is not only feasible but is also able to recover sufficiently accurate valuations. Overall, our framework is able to capture the efficiency–accuracy trade-off in Shapley calculations and yields a favorable efficient frontier across a variety of use cases.

In summary, our paper makes four key contributions. First, we address the important and largely unsolved problem of source document valuation in the context of LLM-generated summaries. By leveraging the Shapley value framework, we propose a principled approach for fair document valuation, to the best of our knowledge, the first study to apply this concept to LLM-based summarization systems. Second, we introduce the Cluster Shapley algorithm, which improves the computational efficiency of Shapley value approximation by exploiting semantic similarity among documents. We provide theoretical guarantees on its performance and evaluate its empirical effectiveness using real-world data, showing that it substantially improves the efficiency–accuracy frontier. Third, our results open up a new research direction by demonstrating that Shapley approximation algorithms can be significantly improved in LLM applications when additional structure, such as embedding-based similarity, is available. This highlights the potential for developing a broader class of structure-aware Shapley approximation algorithms for LLM applications. Fourth, we show that simple attribution rules, though computationally cheap, can lead to substantial unfair outcomes. This insight provides practical guidance for decision-makers: platforms relying on simplistic heuristics may systematically undervalue or overvalue individual contributors, underscoring the importance of adopting fair and theoretically grounded attribution frameworks.

2 Related Literature

Our research relates to and contributes to multiple streams of work, including game theory, LLMs, and business applications of AI systems. First, our work relates to the growing literature on LLM-based summarization. Recent research on summarization has shown that summaries produced by LLMs like GPT-4

achieve comparable or superior factual accuracy, coherence, and overall quality compared to human annotators in news summarization tasks (Pu et al., 2023). Indeed, text summarization has evolved from traditional extractive methods like TextRank (Mihalcea and Tarau, 2004) to more sophisticated abstractive approaches powered by LLMs. RAG frameworks (Lewis et al., 2020), which combine information retrieval with LLM generation, have shown significant improvements in factual accuracy and information currency (Gao et al., 2023; Jiang et al., 2023). Recent developments include GraphRAG (Edge et al., 2024), which enhances retrieval performance using graph-based representations. While summarization technologies continue to evolve rapidly, our work addresses the fundamental challenge of document valuation that persists across summarization approaches. The document valuation framework we develop is designed to be agnostic to specific summarization techniques, ensuring its applicability even as LLM and RAG technologies advance.

Next, our work builds on the game-theoretic concept of Shapley values (Shapley, 1953), which has gained significant traction in machine learning for quantifying feature importance (Lundberg and Lee, 2017) and data valuation (Jia et al., 2019; Ghorbani and Zou, 2019).[6] While prior studies have applied Shapley values primarily to supervised learning tasks and feature attribution, our work represents, to our knowledge, the first application of Shapley values to document valuation in LLM-based summarization systems. Our work also relates to the work on approximation algorithms for computing Shapley values since it is well-known that computing exact Shapley values is computationally challenging. Without additional structural assumptions on the value function beyond bounded range, any unbiased estimator must rely on sampling permutations or coalitions; simple Monte Carlo (random-permutation) estimators are unbiased and achieve concentration rates that match information-theoretic lower bounds up to constants (Castro et al., 2009; Maleki et al., 2013). Specifically, with value range bounded by $[a, b]$, Hoeffding-style arguments yield a sample complexity of $O\left(\frac{(b-a)^2}{\varepsilon^2} \log \frac{1}{\delta}\right)$ to obtain an ε-accurate estimate with probability $1-\delta$, and Maleki et al. (2013) give matching lower bounds. Alternative speedups such as linear regression surrogates, kernel/coalitional weighting, or grouping assumptions can reduce computation but introduce modeling assumptions that may bias attribution when violated (Fatima et al., 2008; Bachrach et al., 2010; Ghorbani and Zou, 2019).

In our proposed Cluster Shapley algorithm, we explicitly account for the semantic similarity between documents, which allows us to achieve high accuracy at relatively low computation costs. We expect that our general idea of using a clustering approach to capture the similarity between documents/observations

[6] Apart from the Shapley value approach, alternative approaches for calculating the value of contributors in cooperative tasks have also been proposed. Some of the well-known approaches include the leave-one-out approach (Cook, 1977) and the influence function approach (Barshan et al., 2020; Han et al., 2020; Guo et al., 2020). However, these methods, neglecting high-order interactions, may result in undesirable data valuation. For example, when there are two or more similar contributors, and tend to assign nearly zero value to similar documents. For example, consider a scenario where we have three documents, A, B, and C, where A and B have the same high-quality content and C contributes nothing to the overall value. Intuitively, we should expect A and B to have the same valuation and C to have zero value. However, the leave-one-out approach will assign zero value to both A and B – if we leave A out and calculate the value generated by B and C, the value would almost be the same as that with A (and vice-versa for B). Thus, if we believe this approach and assume that both A and B have no incremental value and drop them both from the set of contributors, the overall value generated by the set will go to zero! In sum, these approaches are unable to correctly account for the marginal contributions of each player. This is the reason why Shapley has emerged as the dominant paradigm for valuing contributors in cooperative settings.

in Shapley calculations can be extended beyond LLM summarization settings and applied more broadly to other cases where similarity measures between observations exist. While the notion of computing Shapley values for groups of data points has been previously mentioned – e.g., Ghorbani and Zou (2019) briefly discuss group-level Shapley values in their Section 3.2, and Corder and Decker (2019) explore a hierarchical clustering approach combined with exact Shapley calculations-we make two contributions. First, we formally analyze the trade-off between approximation error and computational complexity, whereas Corder and Decker (2019) only provide theoretical results under restrictive assumptions (e.g., K-means with two clusters). Second, to our knowledge, we are the first to use LLM-generated text embeddings to guide the clustering, which better captures semantic relationships and is especially suited to our task.

More broadly, recent works such as Wang et al. (2024) have used the notion of Shapley value to quantify the value of large corpora to pretraining by approximating Shapley values through repeated fine-tuning (an expensive and often infeasible process). In contrast, our method targets post-training applications – specifically, LLM-driven summarization tasks, such as AI search or QA, where the model is fixed and attribution occurs at inference time. Because our approach approximates Shapley values based on LLM-generated embeddings, it avoids retraining and scales more efficiently with the number of input documents. This distinction is crucial: Wang et al. (2024) deals with corpus-level attribution during LLM training; we focus on attribution over documents presented as context to an already-trained LLM, making our method more practical for interactive, high-frequency applications.

3 Problem Definition

We define the problem from the perspective of a platform that has access to D original documents generated by different producers. We do not make any distributional assumptions on D. These documents are not necessarily i.i.d. and may contain overlapping information and vary in the quantity and quality of content. Users arrive at the platform and query for some information from the platform using queries q, drawn from a distribution $g(q)$. The platform generates a response to each query q based on the D documents using an LLM-based summarization model $A(q, D)$. We can view $A(q, D)$ as a black box that takes a dataset D of any size from 0 to ∞ to generate a summary in response to query q. Note that in practice, the platform may choose to only use a subset of documents ($S_q \subseteq D$) that are most relevant to the query for the summarization process; that is, we allow for cases where all documents are not relevant to all queries. In such cases, the summarization process is denoted by $A(q, S_q)$.

The quality or performance of a summarization is denoted by $v(q, A(q, S_q))$. Intuitively, this score captures the extent to which the user finds the summary useful or valuable. The performance score v can be treated as a black-box that takes the query and summary as input and returns a score. In practice, $v(q, A(q, S_q))$ can be obtained in a multitude of ways. It could be actual scores collected from user surveys on how helpful they find a given summary to be (e.g., rating of helpfulness, fraction of upvotes). Alternatively, it could be helpfulness scores based on an LLM model, where an independent LLM agent does the scoring instead of human agents. This can be a viable option in settings where collecting user responses is costly and/or slow; indeed, recent research has shown that LLM ratings tend to align with user ratings in many

situations (Kang et al., 2023; Cheng et al., 2025). It could also represent implicit helpfulness scores based on user behavior, which are commonly used in the information retrieval and search literature to measure the relevance of a given document/link, e.g., whether the user clicked on the summary, the time spent reading the summary (Liu et al., 2009; Yoganarasimhan, 2020).

The platform's goal is to determine how much each document $i \in D$ contributes to the quality of summaries produced for user queries. Because both the summarization model $A(\cdot)$ and the evaluation method $v(\cdot)$ are fully controlled by the platform, content providers cannot directly influence the generation or evaluation of outputs. This ensures that the valuation mechanism is robust to manipulation and reflects the intrinsic contribution of each document.

We define $\phi_i(q, S_q, A, v)$ as the value of document i for a given query q. This value function depends on the subset of documents retrieved for query q (i.e., $S_q \subseteq D$), the summary function of documents $A(q, S)$, and the resulting performance score function $v(q, A(q, S))$. Throughout the discussion, we use the shorthand $\phi_i(q) = \phi_i(q, S_q, A, v)$ whenever the context is clear. This setup allows the platform to aggregate document values across queries while respecting both the relative importance of documents and the frequency of the queries they support. For instance, a document that is essential for answering a rare query may be less valuable overall than a document that provides moderate value across many common queries.

We argue that it is important to conduct value attribution at the query level, especially for small publishers and niche queries. First, the economic impact of LLM-generated answers varies substantially across queries and sources. Second, large publishers can negotiate direct licensing agreements with LLM providers, but small or niche sites, whose value lies in specific, long-tail queries, cannot feasibly do so at scale. A query-level approach also ensures that marginal contributions are measured in the exact contexts where the content is used, leading to a more accurate and fair valuation. Thus, our goal is to develop a query-level document valuation approach that satisfies two properties:

- Summarization Procedure Agnostic: The approach should be agnostic to the specifics of the summarization process, $A(\cdot)$, used by the platform. That is, it should generalize across RAG systems. While we present a standard RAG implementation in §6.1, numerous alternatives exist-from simpler methods to more sophisticated frameworks like TextRank, GraphRAG, and DRAG (Mihalcea and Tarau, 2004; Edge et al., 2024; Zhang et al., 2025). As LLM technologies rapidly evolve, our document valuation framework is designed to remain effective, regardless of advancements in summarization, ensuring broad applicability across current and future implementations.
- Evaluation Process Agnostic: The approach should apply to any evaluation method, $v(\cdot)$. As discussed earlier, many explicit and implicit approaches for scoring summaries exist. Different business use cases may have access to different evaluation approaches. For example, search engines (e.g., Perplexity or Google) usually only have implicit feedback/evaluation, whereas question-answering websites or review aggregators may have more explicit feedback on the helpfulness of reviews. We would like our algorithm to be agnostic to the exact approach used. In our empirical context, we use a prompt-based LLM approach for evaluation; see §6.2.

4 Solution Concept: Shapley Value for Document Valuation

This section presents a principled framework for addressing the problem of document valuation in LLM summaries, as introduced in §3. In §4.1, we introduce Shapley value, a game-theory concept for assigning value to individual documents in cooperative settings, and explain its relevance to our context. In §4.2, we discuss the business application of using Shapley value for revenue attribution. While theoretically appealing, computing exact Shapley values is not practically applicable in most settings (including ours). To address this, §4.3 introduces an efficient approximation algorithm which leverages document similarity measures to reduce the computational burden. This algorithm serves as a proof of concept within our document valuation framework; in real-world implementations, it can be adapted or extended to align with the specific architectures and operational constraints of modern LLM-based systems. Finally, in §4.4, we analyze the theoretical properties of this algorithm and provide guarantees on approximation error and computation complexity.

4.1 Shapley Value

We now introduce the Shapley value along with a concise, self-contained explanation of the framework. Following the standard Shapley literature, we adapt and present the four properties of $\phi_i(q)$ within the context of our LLM-summarization problem to guarantee **fair document valuation**:

1. **Efficiency.** Efficiency ensures that the total value generated by the summarized document is fully distributed among the documents, with no surplus or deficit. Mathematically, this is represented as:

$$\sum_{i \in D} \phi_i(q) = v(q, A(q, D)). \tag{1}$$

   If the efficiency property is not enforced, the value function ϕ is only determined up to a proportional constant (see Theorem 2.1 in Ghorbani and Zou (2019)).

2. **Symmetry.** Symmetry ensures that documents with equal contributions are valued equitably. That is, two documents i and j have the same valuation if they contribute equally to every possible coalition. Formally:

$$\phi_i(q) = \phi_j(q), \quad \text{if} \quad v(q, A(q, S \cup \{i\})) = v(q, A(q, S \cup \{j\})) \ \ \forall \ \ S \subseteq D \setminus \{i, j\}. \tag{2}$$

3. **Null Document.** A null document implies that if a document provides no marginal value to any subset of documents, its value is zero. Formally, a document i in a query q is called null if $v(q, A(q, S \cup \{i\})) = v(q, A(q, S))$ for subsets $S \subseteq D \setminus \{i\}$. If document i for query q is null, then the value $\phi_i(q) = 0$.

   In our setting, any document i not used in the summarization process for query q has zero value for that query. That is, $\phi_i(q) = 0$, if $i \notin S_q$.

4. **Linearity.** The values of document i under two separate queries q_1 and q_2, sum up to its value when

evaluated using a performance score function that combines the individual performance score functions. Formally:

$$\phi_i(q_1 + q_2) = \phi_i(q_1) + \phi_i(q_2), \tag{3}$$

where $q_1 + q_2$ represents a combination of two queries, and the performance score function for this combined query is naturally defined as $v(q_1, A(q_1, S)) + v(q_2, A(q_2, S))$, reflecting the aggregate contributions of q_1 and q_2.

Note that the combination of two queries $q_1 + q_2$ does not imply that the two queries are merged into a single new query. Instead, it represents a setting where there are two distinct queries being processed (this definition naturally extends to any finite number of queries, not just two). For example, consider two queries: one on quality (q_1) and another on price (q_2). The value of document i under the quality query is denoted by $\phi_i(q_1)$, calculated using the performance score function $v(q_1, A(q_1, S))$. Similarly, $\phi_i(q_2)$ represents the value of document i under the price query, based on the performance score function $v(q_2, A(q_2, S))$. The linearity property asserts that if we sum the values obtained from these separate queries, i.e., $\phi_i(q_1) + \phi_i(q_2)$, the result is equivalent to the value of document i calculated under a new, combined performance score function $v(q_1, A(q_1, S)) + v(q_2, A(q_2, S))$. This property ensures that value attributions are invariant to the merging of multiple queries, which has important implications for revenue attribution, as discussed in §4.2.

While other desirable properties are worth discussing, these four – Efficiency, Symmetry, Null Document, and Linearity – are sufficient to uniquely determine the document value function $\phi_i(q)$, known as Shapley value.[7] This is a foundational result in cooperative game theory. We refer interested readers to Shapley's seminal paper for a formal proof, and henceforth only focus on how these properties apply to our context and their importance in our setting.

The Shapley value $\phi_i(q)$ for a document $i \in S_q \subseteq D$ can be written as the expected marginal contribution of document i across all possible coalitions:

$$\phi_i(q) = \frac{1}{|S_q|} \sum_{S \subseteq S_q \setminus \{i\}} \frac{v(q, A(q, S \cup \{i\})) - v(q, A(q, S))}{\binom{|S_q|-1}{|S|}}. \tag{4}$$

This can be stated equivalently as $\phi_i(q) = \frac{1}{|S_q|!} \sum_{\pi \in \Pi(S_q)} [v(q, A(q, P_i^\pi \cup \{i\})) - v(q, A(q, P_i^\pi))]$ where $\pi \in \Pi(S_q)$ is a permutation of S_q, and P_i^π is the set of documents which precede document i in the permutation π.

Note that for $i \notin S_q$, the Shapley value of i is $\phi_i(q) = 0$ because only documents in S_q are used for summarization. Documents outside S_q have no marginal contribution and thus receive a Shapley value of zero. By simple calculation, the Shapley value formula can thus be equivalently written over the original

[7]These four properties (axioms) are independent of any specific summarization (A) or evaluation (v), which are laid out in our problem definition. These properties originate from Shapley's work on cooperative game theory (Shapley, 1953).

document set D as:

$$\phi_i(q) = \frac{1}{|D|} \sum_{S \subseteq D \setminus \{i\}} \frac{v(q, A(q, S \cup \{i\})) - v(q, A(q, S))}{\binom{|D|-1}{|S|}}. \tag{5}$$

The equivalence follows directly from the permutation-based definition, as all permutations of $D \setminus S_q$ do not affect the performance score.

A natural concern in attribution systems is whether document providers can strategically manipulate their content to inflate their assigned value. This issue is not new: even in traditional search engines like Google, publishers have long engaged in "mimicking the winner" strategies, which create content that resembles highly ranked pages to improve visibility. While such behavior is theoretically possible, it is difficult to execute in practice. Modern LLM-powered platforms (e.g., Perplexity, ChatGPT, Google's AI Overview) do not disclose their specific RAG systems to external document providers. Moreover, these systems do not present explicit ranked lists but instead synthesize information into summaries, making it harder for publishers to infer their position or influence outcomes. Retrieval in such systems is governed by proprietary, black-box algorithms that incorporate a range of dynamic signals, including semantic embeddings, recency, diversity, and user engagement. This level of opacity is a deliberate design choice, observed not only in LLM systems but also in recommender platforms and digital advertising ecosystems, where hiding algorithmic details is a known strategy to deter gaming (Kroll et al., 2017; Sax and Wang, 2025).

4.2 Revenue Attribution

An important application of our Shapley-based document valuation framework is enabling fair and economically grounded revenue sharing between platforms and document owners. In practice, platforms generate revenue either at the *query level*, for instance through per-query advertising or API charges, or through *subscription-based models* that aggregate revenue over a large number of queries. Our framework can accommodate both settings and provide a principled method for revenue allocation based on each document's marginal contribution to the LLM-generated output.

To illustrate how such a framework can be operationalized in practice, consider platforms running LLM-based summarization services. Such platforms can determine a revenue share rate $\beta \in [0, 1]$ for distribution among documents while retaining $1 - \beta$ as platform revenue. For example, ProRata.AI provides an attribution solution in AI search and shares revenue 50/50 with its sources (ProRata AI, 2025). However, it uses a proprietary algorithmic approach to score and determine attribution (Universal Music Group, 2024), the details of which are not publicly known.

Subscription-Based Revenue. Let queries be drawn from a distribution $g(q)$, and let the total subscription revenue over a billing cycle (e.g., a month) be denoted by R. Then, the total value of document i over the query distribution can be expressed as the expectation:

$$\mathbb{E}_{q \sim g(q)}[\phi_i(q)].$$

The platform can use this to allocate revenue proportionally among documents. Specifically, document i's payout is given by:

$$\beta \cdot R \cdot \frac{\mathbb{E}_{q\sim g(q)}[\phi_i(q)]}{\sum_{j\in D}\mathbb{E}_{q\sim g(q)}[\phi_j(q)]}.$$

This ensures that documents are compensated according to their aggregate value across the full set of queries served, aligning economic incentives with contribution.

Query-Level Revenue. In settings where revenue is generated on a per-query basis-such as through API billing or advertising impressions-the allocation is straightforward. For a given query q that yields revenue r_q, the payment to document i is:

$$\beta \cdot r_q \cdot \frac{\phi_i(q)}{\sum_{j\in S_q}\phi_j(q)}.$$

Combined Revenue. When both subscription-based and query-level revenue are present, the total expected payout to document i is:

$$\text{Total Payout}_i = \beta \cdot \left(R \cdot \frac{\mathbb{E}_{q\sim g(q)}[\phi_i(q)]}{\sum_{j\in D}\mathbb{E}_{q\sim g(q)}[\phi_j(q)]} + \mathbb{E}_{q\sim g(q)}\big[r_q \cdot \frac{\phi_i(q)}{\sum_{j\in S_q}\phi_j(q)}\big] \right).$$

Scalability to Large Query Volumes. In real-world deployments, platforms often serve millions of queries, making exact evaluation of $\mathbb{E}_{q\sim g(q)}$ computationally expensive. A practical solution is the query sampling. Drawing k queries independently from $g(q)$, we obtain an unbiased estimator for the Total Payout$_i$, enabling efficient large-scale implementation.

In sum, our method offers a transparent and theoretically grounded mechanism for sharing platform revenue with document providers. Unlike flat licensing or arbitrary heuristics-such as attributing value based solely on user clicks-our framework accounts for each document's marginal utility within the LLM summarization process. This avoids two key issues with click-based attribution: (1) many users do not click through to original documents, and (2) click rates are strongly influenced by the position or prominence of document presentation. In contrast, Shapley values capture each document's actual contribution to the generated output. This enables fair compensation aligned with true informational value and supports both real-time (query-level) and batch (aggregate) attribution.

4.3 Approximating Shapley Value: Cluster Shapley Approach

Computational Challenge: While Shapley valuation is a theoretically appealing construct, evaluating Shapley values for source attribution presents a significant computational challenge even in moderate-sized settings. The computational cost associated with Shapley calculation exhibits exponential complexity – if we have S_q relevant documents for query q, the number of summarizations and evaluations needed to compute Shapley values scales as $2^{|S_q|} - 1$, where $|S_q|$ is the number of relevant documents. This rapid growth in the number of calculations makes exact Shapley computation infeasible for large datasets, as the number of summarizations and evaluations quickly becomes overwhelming. Essentially, for each combination of documents, we need the LLM to generate a new summary and then perform an evaluation of that summary.

These constraints suggest that exact Shapley methods in large-scale applications (e.g., document valuation for large platforms using LLMs) are infeasible, and we therefore need efficient approximation algorithms.

Researchers have proposed a number of algorithms designed to address the computational challenge associated with Shapley calculations. These approaches typically adopt a variety of sampling techniques to reduce the computational cost associated with Shapley calculation. One widely used method is Monte Carlo algorithm (Mann and Shapley, 1960), which estimates Shapley values by randomly sampling permutations and computing marginal contributions across these samples. While this approach reduces computational costs compared to exact Shapley values, it still requires a large number of samples to achieve reasonable accuracy. Truncated Monte Carlo (Ghorbani and Zou, 2019) improves efficiency by stopping the calculation early when additional samples provide diminishing returns below a threshold, significantly cutting down computational overhead. Another popular approach, Kernel SHAP (Lundberg and Lee, 2017), employs a regression-based approximation to estimate Shapley values. However, none of these approaches leverage the textual content of documents when approximating Shapley values, treating them purely as independent units.
Key Idea: Motivated by this limitation, we propose the Cluster Shapley algorithm. Instead of treating documents as independent, our method utilizes LLM-generated embeddings to identify and group similar documents, reducing redundant evaluations. Our core idea is intuitive: documents with similar content should have comparable contributions to the final summary and, therefore, should receive similar Shapley values. Our approach leverages the textual content of the documents and the LLM's numerical representation of this textual content (i.e., text embedding) to help approximate and simplify Shapley calculations.

Text embedding techniques convert large chunks of text-such as sentences, paragraphs, or documents-into numerical vectors that capture semantic information. Earlier embedding methods, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), are based on shallow neural networks and co-occurrence statistics, learning word-level embeddings by predicting surrounding context words or factorizing word co-occurrence matrices. These embeddings typically represent each word with a fixed vector, independent of context. In contrast, modern LLM-based embeddings, such as those produced by OpenAI's latest text-embedding models, are generated using Transformer-based architectures and are pretrained on massive text corpora via next-token prediction objectives. These newer embeddings are contextualized-meaning the vector for a word or sentence depends on its surrounding context-and are typically high-dimensional (e.g., 3072 dimensions in OpenAI's `text-embedding-3-large` model). Unlike generative LLMs designed for tasks like chat or text generation, embedding models are optimized to produce semantically meaningful representations suitable for a wide range of downstream tasks. LLM-based embedding vectors have been successfully applied to a wide variety of discriminative tasks, including text classification, document retrieval, sentiment analysis, and predicting the attractiveness of news headlines (Patil et al., 2023; Ye et al., 2025). In our setting, we leverage these embeddings to cluster similar documents before computing Shapley values, allowing us to reduce redundant calculations.
Our Approach: We outline our proposed Cluster Shapley Algorithm in Algorithm 1. Cluster Shapley begins with a preprocessing step, where for a given query q, we first determine the set of relevant documents $S_q \subseteq D$.

This retrieval step ensures that only contextually relevant documents are considered; e.g., if the query pertains to political news, unrelated sports articles will be excluded from summarization. In §6.1, we discuss the retrieval and summarization steps, and related literature in further detail. For each document $i \in S_q$, we also obtain its embedding vector e_i, using an LLM embedding model. This step can be performed using proprietary models like OpenAI and Gemini or open-source alternatives such as Llama/Alpaca.

Because similar documents tend to have similar embeddings, we can use the text embeddings to cluster the documents into similar groups. Specifically, after getting the embeddings, we cluster the embeddings of S_q based on their distance, as outlined in Step 1. To achieve this, we first need to quantify the similarity between any two documents i and j in S_q. We employ cosine similarity, a widely used metric for measuring the closeness of embeddings, for this purpose. Cosine similarity is defined as:

$$\text{cosine_similarity}(e_i, e_j) = \frac{e_i \cdot e_j}{\|e_i\|\|e_j\|}. \tag{6}$$

This metric measures the cosine of the angle between two vectors in an inner product space, capturing how similar their directional components are. Higher cosine similarity values indicate greater textual similarity, meaning the embeddings of semantically similar documents are more aligned. To facilitate clustering, we define a corresponding distance measure, $d(e_i, e_j)$, which is bounded within the range $[0, 1]$, given by:

$$d(e_i, e_j) = 1 - \text{cosine_similarity}(e_i, e_j) = 1 - \frac{e_i \cdot e_j}{\|e_i\|\|e_j\|}. \tag{7}$$

This definition ensures that similar documents, which have high cosine similarity, are assigned a smaller distance value. Documents with lower distance values are more likely to be grouped together in the clustering process, allowing us to reduce redundancy and improve computational efficiency in Shapley value estimation.

Algorithm 1 Cluster Shapley Algorithm

Step 0: Inputs and preprocessing.
Given a query q, retrieve the set of relevant documents S_q.
For each document $i \in S_q$, obtain its embedding vector e_i.
Set the clustering diameter (similarity threshold) $\epsilon > 0$.

Step 1: Document clustering with the distance constraint.
Cluster the documents in S_q into clusters $\{G_1, G_2, \ldots, G_m\}$ such that for any $i, j \in G_k$, $k \in [m] := \{1, 2, \ldots, m\}$, we have $d(e_i, e_j) \leq \epsilon$. Each cluster G_k is a group of similar documents.

Step 2: Cluster-level Shapley value computation.
Define a cluster-level value function $v_{\mathcal{G}}(T) := v\left(\bigcup_{G_k \in T} G_k\right)$ for $T \subseteq \{G_1, \ldots, G_m\}$.
Compute the Shapley value $\hat{\phi}_{G_k}$ for each cluster G_k based on $v_{\mathcal{G}}$.

Step 3: Document-level value allocation.
For each document $i \in G_k$, $\forall k \in [m]$, assign the approximated Shapley value:

$$\hat{\phi}_i = \frac{\hat{\phi}_{G_k}}{|G_k|}. \tag{8}$$

The goal of Step 1 is to partition the documents into m non-overlapping clusters, i.e., $D = \bigcup_{k=1}^{m} G_i$ with

$G_i \cap G_j = \emptyset$ for $i \neq j$. We can employ any off-the-shelf clustering algorithms in Step 1, such as K-Means (Lloyd, 1982) or Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996). However, standard clustering algorithms do not always enforce uniform intra-cluster proximity-that is, they may allow pairs of documents within the same cluster to be far apart, particularly in the presence of high noise or uneven density. To address this limitation, we propose an adaptive DBSCAN algorithm (Algorithm 3) detailed in Web Appendix §B.1. For comparison, we also empirically evaluate the standard DBSCAN in §7.3, which performs worse in our setting.

The clustering diameter ϵ is a key hyperparameter that governs both the approximation error and the computational cost. Smaller values of ϵ lead to more accurate approximations of the Shapley value but incur higher computational cost, as formalized in §4.4. In practice, one can perform standard hyperparameter tuning: use a separate dataset to select the value of ϵ that best balances approximation accuracy and computational efficiency. This selected value can then be fixed for future use. Later in §7.1, we empirically show how ϵ governs the trade-offs between computational costs and approximation error in our application setting.

In Step 2 of Cluster Shapley (Algorithm 1), we consolidate the documents within each cluster by concatenating them into a single *meta-document*, which serves as a representative unit for that cluster. Instead of computing Shapley values for individual documents, we calculate the Shapley value $\hat{\phi}_{G_k}$ for each cluster $k \in [m]$, significantly reducing the computational complexity. These meta-documents are then used as inputs for LLM-based summarization. The generated summaries are then evaluated using predefined metrics, allowing us to compute Shapley values that quantify each cluster's contribution to the final summary.

Note that our Cluster Shapley framework is flexible with respect to the choice of Shapley computation method. In Step 2, any Shapley value algorithm can be applied, including exact calculations or approximation techniques. In case we have a very large dataset or if there are a large number of clusters, approximation methods such as Monte Carlo can be employed to further reduce computational costs while maintaining reasonable accuracy. We formally discuss this additional approximation in §4.4.3.

Finally, in Step 3, we attribute the cluster's Shapley value equally across its individual documents, assigning $\hat{\phi}_i = \hat{\phi}_{G_k}/|G_k|$ as the Shapley value for document i in cluster G_k.

4.4 Theoretical Analysis of Cluster Shapley Algorithm

We now analyze our proposed Cluster Shapley algorithm. First, in §4.4.1, we show that the approximation error of Cluster Shapley can be bounded. Second, we characterize the computational complexity of the algorithm in §4.4.2. Finally, in §4.4.3, we consider a setting where m is still large (so that 2^m complexity is burdensome). In this case, we show that it is feasible to use an approximation algorithm (e.g., Monte Carlo) in the second step of Cluster Shapley and bound the approximation error.

4.4.1 Approximation Error of Cluster Shapley

We first introduce the following mild assumption.

Assumption 1 (Lipschitz continuity in embedding space)**.** There exists a constant $L > 0$ such that for any two documents $i, j \in S_q$ and for any coalition $S \subseteq S_q$ not containing i or j, the difference in their marginal

contributions is bounded by the embedding distance:

$$\Big|\big(v(S\cup\{i\})-v(S)\big)-\big(v(S\cup\{j\})-v(S)\big)\Big| \le L\, d(e_i,e_j). \tag{9}$$

Under Assumption 1, if two documents lie in the same cluster ($d(e_i,e_j)\le\epsilon$), their marginal contributions to any coalition differ by at most $L\epsilon$. Intuitively, this assumption states that semantically similar documents should exhibit similar marginal contributions across all possible coalitions. The assumption is both theoretically reasonable and practically essential: documents with similar content naturally provide comparable value when added to any subset of documents, ensuring that our clustering approach meaningfully groups functionally equivalent items. We empirically validate this assumption for our empirical application in Web Appendix §B.2, and demonstrate that the Lipschitz upper bound $L\,d(e_i,e_j)$ holds for our setting.

However, clustering introduces an approximation error by merging distinct documents. Intuitively, as the clustering becomes finer (more clusters), the approximation error decreases. In the extreme case where each document forms its own singleton cluster, the algorithm performs an exact Shapley calculation on individual documents, hence producing no error. The following theorem establishes that under Assumption 1, the Cluster Shapley value of each document converges to its true Shapley value as the clustering granularity increases and provides a bound on the approximation error in terms of the clustering diameter ϵ.

Theorem 1 (Convergence and Approximation Error Bound)**.** Under Assumption 1, the approximated Shapley values $\hat{\phi}$, output by the Cluster Shapley algorithm, converge to the exact Shapley values ϕ as the clustering diameter ϵ approaches zero (i.e., as each cluster becomes an identical-document singleton). In particular, for any document i, the approximation error is bounded by:

$$|\hat{\phi}_i-\phi_i| \le L\,\epsilon. \tag{10}$$

See Web Appendix §B.3 for the proof. This theorem provides an approximation error bound $|\hat{\phi}_i-\phi_i|$ for each document's approximated Shapley value as a function of the clustering diameter ϵ. Intuitively, it implies that the MAE of the Cluster Shapley value of a document i ($\hat{\phi}_i$) is upper bounded by the maximum distance between any two documents in the cluster (ϵ) times a constant L. We now further interpret this result and consider the following special case.

Corollary 1 (Accuracy in homogeneous clusters)**.** If a cluster G_k is such that all member documents are nearly identical, i.e., the maximum difference $\phi_{\max}-\phi_{\min}=\delta\ll 1$, then each document's Cluster Shapley value is within δ of its true Shapley value, $|\hat{\phi}_i-\phi_i|\le\delta$ for all $i\in G_k$. In particular, if documents in G_k are symmetric (meaning $v(S\cup\{i\})=v(S\cup\{j\})$ for all S and all $i,j\in G_k$), then $\hat{\phi}_i=\phi_i$ exactly for all $i\in G_k$.

The proof can be found in Web Appendix §B.4. This corollary confirms that the algorithm is exact for clusters of truly similar documents. In realistic settings, documents in a cluster may not be perfectly identical in contribution, but as long as the within-cluster heterogeneity is small, the approximation will be accurate.

4.4.2 Computational Complexity Analysis

We now analyze the computational complexity of Cluster Shapley and how it scales with the number of documents $n = |S_q|$ and the number of clusters m. We break down the algorithm into its two main steps: clustering and Cluster Shapley value calculation.

- **Document clustering complexity:** The algorithm must compute embeddings for n documents and then cluster them based on pairwise distances. Generating embeddings (using a pre-trained LLM embedding model) takes $O(n)$ operations. Computing the pairwise distance matrix naively costs $O(n^2)$ time (since there are $\binom{n}{2}$ pairs). The clustering itself using DBSCAN will typically require examining each point's neighbors; DBSCAN in worst-case can be $O(n^2)$, though it can approach $O(n \log n)$ on average (Ester et al., 1996). In summary, the clustering step is polynomial in n. We can reasonably approximate it as $O(n^2)$ in the worst case, which is manageable for moderate n. This phase is far more efficient than an exhaustive Shapley computation (which is exponential in n).
- **Cluster-level Shapley value computation complexity:** Once m clusters are formed, we must compute the Shapley values for these m meta-players. If we perform the exact Shapley computation at the cluster level, there are 2^m subsets in total (including empty and full), but since $v(\emptyset) = 0$ is trivial, one often writes $2^m - 1$ evaluations. Each such evaluation requires generating a summary from the union of documents in those clusters and evaluating it. Thus, this step is exponential in m and the worst-case runtime for computing cluster Shapley exactly is $O(2^m)$.

Combining the two phases, the overall worst-case time complexity is $O(n^2) + O(2^m)$. It is important to note that the exponential complexity in m is unavoidable if using the exact calculation in Step 2 of the Cluster Shapley algorithm. Cluster Shapley does not magically circumvent the combinatorial explosion; rather, it reduces the problem size from n to m by leveraging redundancy in the document.

4.4.3 Large m Scenario

In some cases, the number of clusters, m, can be large. Then, computing and evaluating all 2^m subsets is burdensome. In such cases, Cluster Shapley can be combined with *any* cluster-level Shapley approximation algorithm. Specifically, instead of exactly evaluating $v_{\mathcal{G}}$ on all cluster subsets, we can apply one of the existing Shapley approximation algorithms $\mathcal{A}$ with known convergence rate, approximation error, and computational cost. We formally describe this generalized procedure in Algorithm 2, and establish the corresponding error bound and computational complexity in Theorem 2.

Theorem 2 (General Cluster Shapley Approximation)**.** Suppose *(i)* Assumption 1 holds, *(ii)* Approximation algorithm $\mathcal{A}$ ensures that each cluster Shapley value $\tilde{\phi}_{G_k}$ satisfies $|\tilde{\phi}_{G_k} - \hat{\phi}_{G_k}| \leq \epsilon_{\mathcal{A}}(\eta)$ with probability at least $1 - \eta$, and *(iii)* Approximation algorithm $\mathcal{A}$ has computational complexity $C_{\mathcal{A}}(m)$, then for any document $i \in G_k$, with probability at least $1 - \eta$:

$$|\tilde{\phi}_i - \phi_i| \leq L\epsilon + \frac{\epsilon_{\mathcal{A}}(\eta)}{|G_k|}. \tag{11}$$

Algorithm 2 Cluster Shapley with Cluster-level Approximation

Step 0: Inputs and preprocessing.
Given a query q, retrieve the set of relevant documents S_q.
For each document $i \in S_q$, obtain its embedding vector e_i.
Set the clustering diameter (similarity threshold) $\epsilon > 0$.

Step 1: Document clustering.
Cluster the documents in S_q into groups $\{G_1, G_2, \ldots, G_m\}$ such that for any $i, j \in G_k,\ k \in [m] := \{1, 2, \ldots, m\}$, we have $d(e_i, e_j) \leq \epsilon$. Each cluster G_k is a group of similar documents.

Step 2: Cluster-level Shapley value approximation.
Define a cluster-level value function $v_{\mathcal{G}}(T) := v\left(\bigcup_{G_k \in T} G_k\right)$ for $T \subseteq \{G_1, \ldots, G_m\}$.
Apply approximation algorithm $\mathcal{A}$ to compute estimated cluster Shapley values $\tilde{\phi}_{G_1}, \ldots, \tilde{\phi}_{G_m}$.

Step 3: Document-level value allocation.
For each document $i \in G_k,\ \forall k \in [m]$, assign the approximated Shapley value:

$$\tilde{\phi}_i = \frac{\tilde{\phi}_{G_k}}{|G_k|}.$$

Further, the computational complexity of this general Cluster Shapley algorithm is given by $O(n^2 + C_{\mathcal{A}}(m))$.

See Web Appendix §B.5 for the proof. $\mathcal{A}$ can denote any approximation algorithm. In the following corollary, we show an instantiation of Theorem 2 for the popular Monte Carlo approximation algorithm.

Corollary 2 (Monte Carlo Cluster Shapley Error and Complexity)**.** Suppose $\mathcal{A}$ is the Monte Carlo approximation algorithm with N random permutations. Under Assumption 1 and bounded marginal contributions $V_{\max}$, for any document $i \in G_k$ with probability $1 - \eta$, we have:

$$|\tilde{\phi}_i - \phi_i| \leq L\epsilon + \frac{V_{\max}}{|G_k|}\sqrt{\frac{\log(2/\eta)}{2N}}. \tag{12}$$

To ensure $|\tilde{\phi}_i - \phi_i| \leq \epsilon_{\text{total}}$, it suffices to choose $N \geq \frac{V_{\max}^2 \log(2/\eta)}{2|G_k|^2(\epsilon_{\text{total}} - L\epsilon)^2}$. The total computational complexity for this algorithm is $O(n^2 + Nm)$.

See Web Appendix §B.6 for the proof. Effectively, this corollary states that the overall approximation error of this generalized Cluster Shapley algorithm is bounded by the sum of two approximation errors, where the first term in Equation (12) is the error from the clustering step (Step 1) whereas the second term is the error from the Monte Carlo approximation in Step 2. This also implies that to maintain the same order of total approximation error, the computational complexity scales as $O(n^2 + m\epsilon^{-2})$.

In sum, our theoretical analysis shows that Cluster Shapley is a principled, flexible algorithm that offers provable efficiency-accuracy trade-offs. We now demonstrate the empirical performance of our algorithm in the following sections.

5 Application Setting: Amazon Review Dataset

We now present an application of our algorithm to a real setting. We use the publicly available Amazon Product Reviews dataset as the empirical context to demonstrate the performance of our document valuation

approach. This dataset was collected by Hou et al. (2024) and has been used extensively in recent research studies on a variety of topics, including sentiment analysis (Haque et al., 2018), sequential product search and recommendation (Hou et al., 2024), fine-tuning of LLMs (Zhang et al., 2024), and evaluation of LLM alignment (Shankar et al., 2024). The dataset spans from May 1996 to September 2023, featuring over 571.54 million reviews from 54.51 million users and covering 48.19 million unique items. It is organized into 33 distinct categories, including electronics, household goods, clothing, and books. This user review data set consists of textual feedback provided by users that captures their opinions, ratings, and experiences with products. A comprehensive analysis of review categories, basic statistics, and detailed data field information is available in Hou et al. (2024).

While it is well-established that review valence and content can help consumers make better decisions (Chevalier and Mayzlin, 2006), it is also well-understood that it is hard for consumers to process the large amounts of information/text in reviews. For example, the most popular products in the data have hundreds or even thousands of reviews. Thus, consumers must often sift through hundreds of reviews to extract relevant insights. This information overload makes it difficult for users to efficiently locate specific details (e.g., product quality, value for money, durability, ease of return).

To help consumers navigate this vast amount of information, online platforms typically rank reviews by helpfulness votes and allow searching for specific information. While these solutions can aid consumers in their quest for information, they nevertheless require users to sift through a large volume of irrelevant information and expend significant time and effort on the task. As such, it often leads to inefficient searches and potentially uninformed purchasing decisions. To that end, many e-commerce platforms (including Amazon) have started adopting LLMs to retrieve and summarize the most relevant information for a consumer's specific query from the available set of reviews/user-generated content (see Figure 1). Customers can either see a summary from all reviews or query the system for a specific piece of information (e.g., ease of return) through Amazon's "Rufus" AI chatbot. For our analysis, we focus on query-based summaries, though our framework is quite general and can also be applied to the general summarization settings.

For our numerical experiments, we select 24 products from different categories to ensure a diverse representation of consumer goods (Table 1). These products span a variety of domains, including video games, beauty products, and personal care items, with review counts varying widely. The number of reviews per product ranges from 323 to 15,594, with a mean of 2,075 and a standard deviation of 3,216. Even the product with the fewest reviews presents a significant information overload for consumers, making it impractical to read through all reviews manually. While our methodology can be applied to a larger set of products, our empirical findings do not fundamentally change with more products. Therefore, we focus on this smaller subset of products for expositional and computational ease.

To compute the Shapley value of each individual review (within the context of a given product), we need to first specify the distribution of queries, $g(q)$, that consumers use when requesting summaries for this product. While this query distribution is not publicly available, for each product, we observe the set of popular attributes frequently mentioned by customers. For instance, in Figure 1, for the wireless controller

product, attributes such as "Functionality" and "Controller quality" are among the most common aspects of the product that users are concerned about. This information allows us to craft a proxy distribution of user queries for each product that mimics the real distribution of queries. Specifically, for each product, we design two queries based on the top two attributes most frequently mentioned in its reviews, as shown in the last column of Table 1. Each query is assigned an equal probability, contributing equally to the overall valuation.[8]

No.	Product	Number of Reviews	Designed Queries
1	Wireless Controller	15,594	1. Does the controller experience unresponsiveness? 2. How would you rate the overall quality of the controller?
2	Hair Diffuser	1,328	1. Is the hair diffuser compact enough for travel? 2. How would you describe the quality of the hair diffuser?
3	PlayStation 5	2,700	1. How's the quality of the PlayStation? 2. How's the graphics of the PlayStation?
4	Headset	6,528	1. What's the overall quality of the headset? 2. Is the headset comfortable?
5	Gift Card	4,827	1. Does the gift card not work well? 2. How quick is the delivery of the gift card?
6	Hair Styling Agent	959	1. How does the texture of the hair styling agent feel? 2. What's the quality of this hair styling agent?
7	Headwrap	561	1. How stretchy is the headwrap? 2. Does the headwrap feel durable and high-quality?
8	Hair Curler	1,243	1. Is it easy and quick to use this hair curler? 2. How's the quality of this hair curler?
9	Hair Brush	1,372	1. Is the hair brush soft and gentle on hair? 2. How's the quality of the hair brush?
10	Makeup Brush	567	1. Does the makeup brush have a smell? 2. How's the quality of the makeup brush?
11	Bath Wash	1,962	1. Does the bath wash make skin softer? 2. Is the quality of the bath wash up to par?
12	Scalp Massager	381	1. What size is this scalp massager? 2. How's the quality of this scalp massager?
13	Audio Cable	1,511	1. How's the quality of the audio cable? 2. How's the noise level of the audio cable?
14	Tint Kit	1,750	1. How good is the quality of the tint kit? 2. Is the tint kit effective?
15	Super Mario	1,221	1. How's the quality of Super Mario? 2. How's the multiplayer capability of Super Mario?
16	Nail Polish	534	1. How's the quality of the nail polish? 2. How's the durability of the nail polish?
17	Nail Aid	323	1. How's the quality of the nail aid? 2. How effective is the nail aid?
18	Mannequin	881	1. How would you describe the quality of the mannequin? 2. Is it a good mannequin for practicing braiding?

Continued on next page

[8] In practice, the platform will have data on the true distribution of queries and can directly use that empirical distribution in its analyses. As discussed earlier, our approach is agnostic to the exact distribution of queries.

No.	Product	Number of Reviews	Designed Query
19	Headbands	1,153	1. How good is the quality of the headbands? 2. Are the headbands comfortable to wear?
20	Gauge Gear	785	1. What's the quality of the gauge gear? 2. Does the gauge gear help with healing?
21	Facial Wipe	446	1. How's the quality of the facial wipes? 2. How effective is the facial wiped at cleaning?
22	Dental Tool	1,343	1. How's the quality of the dental tool? 2. Does the dental tool do its job effectively?
23	Crystal Crowns	1,374	1. What's the quality of the crystal crowns? 2. How beautiful are the crystal crowns?
24	Blemish Formula	449	1. What's the quality of the blemish formula? 2. How effective is the blemish formula?

Table 1: Designed queries for selected products. We assume each product has an equal probability of appearing (i.e., uniform exposure), and each query within a product contributes equally (50%) to the valuation.

6 Implementation Details

The proposed document valuation framework is built on two key components: a summarization method $A(\cdot)$ and an evaluation method $v(\cdot)$, which are detailed in §6.1 and §6.2 respectively. Then, we discuss the implementation details for the exact Shapley and Cluster Shapley approaches in §6.3 and §6.4.

6.1 Summarization of Relevant Amazon Reviews via RAG

AI search engines are designed to provide real-time, contextually relevant responses to user queries. A key enabling technique behind many of these systems is retrieval-augmented generation (RAG), which integrates pre-trained LLMs with information retrieval modules to augment model outputs with domain-relevant documents (Lewis et al., 2020). A detailed introduction to the RAG architecture is provided in Web Appendix §C.1.

To illustrate the impact of RAG integration on LLMs, we provide an example with a query: "Who won the Super Bowl this year?" in Figure 2. In this case, the ChatGPT application (powered by GPT-4o) leverages web browsing to retrieve up-to-date information that is not present in the model's training data (GPT-4o has a knowledge cutoff of September 30, 2023). This example highlights how RAG enables LLMs to access and incorporate real-time information beyond their static training corpus. Note that the RAG system explicitly links to information sources in this example. This is standard in most RAG-based search systems since it allows users to verify response provenance and understand the basis of AI-generated content. Given the effectiveness and scalability of RAG architecture, it now forms the backbone of most of the modern AI-based search and summarization systems, e.g., Amazon's Rufus (Mehta and Chilimbi, 2024), Google AI Search (Reid, 2024), and OpenAI's ChatGPT Search (OpenAI, 2024).

Next, we construct a RAG-based search to find relevant documents S_q from Amazon reviews and summarization tool $A(q, S_q)$ for any given query q. Figure 3 shows the overview of the four-step procedure.

- **Step 0: Generate Text Embeddings**

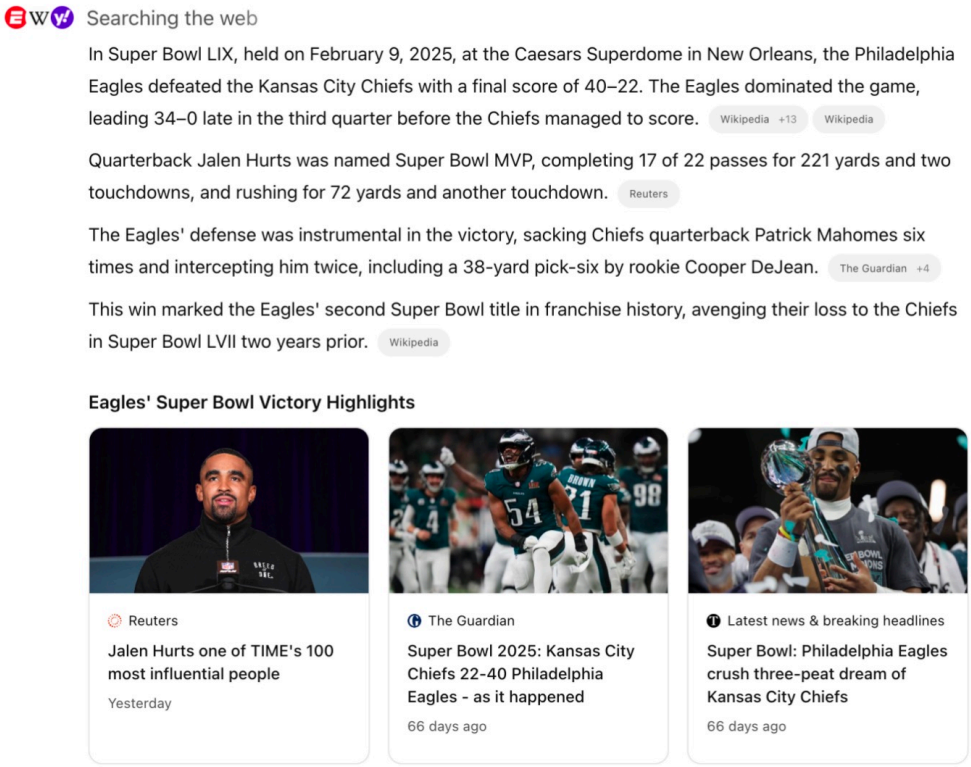


Figure 2: ChatGPT-4o with RAG-Enhanced Web Search

The pre-processing step consists of generating text embeddings for all D reviews/documents associated with a product generated using OpenAI's `text-embedding-3-large` model, which produces embeddings with a default size of 3072 dimensions. These embeddings are based on all the review text, including the title and the main content. In our analysis, we exclude reviews with fewer than 10 words, as they tend to be incomplete or uninformative. Our RAG architecture is agnostic to the exact source of embeddings, and it is possible to use alternative embedding models from open-source LLMs such as Llama, BERT, etc. However, recent research has shown that OpenAI embeddings tend to outperform the embeddings of such earlier models in discriminative tasks (Ye et al., 2025); hence we use the OpenAI embeddings for our application.

- **Step 1: Fetch user query** q
  The process begins with a welcome message from the AI assistant to the user, followed by the user's search query related to some aspect of a product.
- **Step 2: Retrieval of relevant documents** S_q
  We first process the user query to extract the key semantic information in it using an LLM (in our case `GPT-4o-2024-08-06`). The goal of this extraction is to identify the core meaning/consumer need expressed in the user's query. For example, in Figure 3, the user's query is, "I would like to know more details about the quality of the wireless controller." Here, the key semantic information is, "quality of the

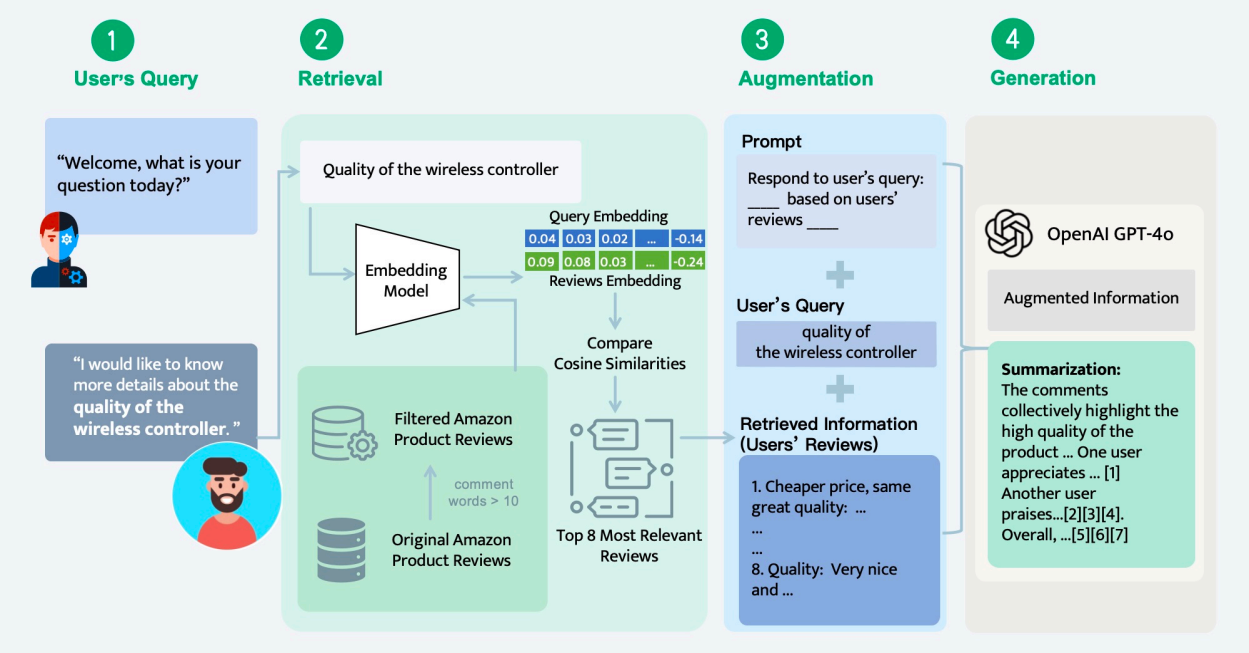


Figure 3: Architecture of our LLM-based search and summarization tool for Amazon Product Reviews. This flowchart illustrates the architecture of an AI-powered search engine designed for processing and summarizing reviews about the quality of DualShock 4 Wireless Controller. The process starts with the user query, where a specific question about the quality is posed. In the retrieval phase, the query's key semantic information, "the quality of the wireless controller", is embedded and compared to filtered Amazon product reviews using cosine similarity. The system then retrieves the top eight most relevant reviews. During the augmentation phase, these retrieved reviews are combined with the original user query and our designed prompt, guiding the generation process. Finally, the generation phase employs OpenAI's GPT-4o model to summarize the augmented information, providing a concise response that cites the specific product reviews to ensure traceability and relevance to the user's query.

wireless controller," which is extracted for further processing.

Next, we use OpenAI's `text-embedding-3-large` model to generate the embedding for the processed query. We denote the embedding of the query as e_q. For each review i in the set of reviews D, we represent its embedding as e_i. We then calculate the cosine similarity between the query embedding e_q and the review embedding e_i for each review. The cosine similarity between the document embedding e_i and the query embedding e_q is defined as $\frac{e_i \cdot e_q}{\|e_i\| \|e_q\|}$, where $e_i \cdot e_q$ is the dot product of the embeddings, and $\|e_i\|$ and $\|e_q\|$ are their respective Euclidean norms. For a detailed explanation of cosine similarity and its application in text similarity tasks, see Chapter 6 of (Schütze et al., 2008). The cosine similarity of a pair of embedding vectors captures the extent to which the two vectors are similar, with higher values indicating greater similarity. Thus, a higher cosine similarity indicates that a given review i has greater relevance to the query q. Next, we rank the cosine similarity scores of the query q for all the D reviews and retain the most relevant reviews.

We restrict ourselves to the eight most relevant reviews ($|S_q| = 8$) from D for all queries, though in practice, the retrieval process can be more complex.[9] Our choice reflects current industry practice. For

[9]We can either retrieve the top few documents or apply a cutoff based on the cosine similarity of the embedding values-excluding

example, Danny (2024) documented that Perplexity AI generated responses citing an average of 5.28 documents per query, and Gupta (2025) mentioned that "On average, (Google) AI overviews include 4-5 citations (the maximum is 33)" from analyzing 1 million overviews. In this context, summarizing the top 8 relevant sources aligns well with how real-world LLM summary applications present content today. That said, our proposed Cluster Shapley algorithm is not limited to small-scale applications. It can be readily scaled to larger document sets when needed.

- **Step 3: Augmentation by putting query q and documents S_q together**
  In the augmentation phase, we combine the user query with the relevant product reviews using a prompt given to OpenAI's GPT-4o model (`GPT-4o-2024-08-06`). The detailed prompt is shown in Figure A7 in Web Appendix §C.2. The prompt instructs the model to analyze the filtered reviews, exclude irrelevant information, and then generate a summary focusing solely on content related to the query.
- **Step 4: Generate the summary $A(q, S_q)$**
  Given the user query (q) and the contextual information from the relevant reviews (S_q), the LLM (GPT-4o) produces a grounded summary by retrieving and synthesizing evidence from the provided documents. Although the LLM retains knowledge from its pretraining corpus, the response in this step is guided primarily by the contextual input delivered through the augmentation prompt (see Web Appendix §C.2). The output summary cites supporting reviews in square brackets (e.g., [2]), enabling users to trace each statement back to specific source documents. If a review is not relevant, it is explicitly marked as such by the model, e.g., ("[4] is not related to the query").

6.2 Evaluation of Summarized Amazon Reviews

Next, we describe the implementation of $v(\cdot)$, a function that evaluates the quality of generated summaries. To operationalize this, we design a prompt for GPT-4o that serves as the performance evaluation method. The full prompt is shown in Figure A8 in Web Appendix §C.2. This prompt takes summaries as inputs and outputs a performance score. The LLM evaluates each summary's informativeness based on its "Information Coverage," reflecting how well the summary captures key aspects of the product reviews.[10] Each summary is rated on a scale from 0 to 10,[11] with higher scores indicating a more comprehensive and accurate reflection of relevant information. The LLM is instructed to prioritize clarity and relevance, emphasizing key details.

Note that LLM outputs are inherently stochastic due to the probabilistic nature of token generation. As a result, both the summarization process $A(q, S)$ and the evaluation process $v(q, A(q, S))$ can produce non-deterministic outputs, even under identical inputs. To quantify and manage this stochasticity, we analyze its impact on the variance of Shapley values. As shown in Web Appendix §C.4, we find that the summarization step contributes approximately 53% of the total output variance, while the evaluation step contributes about

those below a fixed threshold-or combine both approaches by first selecting the top few relevant documents and then filtering those that meet the similarity threshold.

[10] Nevertheless, as discussed in §3, our framework is agnostic to the exact evaluation tool used, and other approaches can be used.

[11] We chose a 0 to 10 scale to offer sufficient granularity for distinguishing levels of information coverage, as smaller scales (e.g., 0 to 5) lack subtlety, while larger scales (e.g., 0 to 100) add unnecessary complexity. We tested alternative ranges to confirm this choice for optimal consistency in evaluation.

47%. To reduce the effect of randomness in evaluation, we run four independent evaluations per summary and use their average score. This simple averaging method substantially reduces evaluation variance while maintaining computational efficiency. Please see Web Appendix §C.4.2 for additional analysis and results.

6.3 Exact Shapley Value Calculation

We now discuss the implementation details for both the exact Shapley value calculation and discuss the scalability and costs of this algorithm.

For each product, we calculate the exact Shapley values using the formula in Equation (4). Table 2 presents the Shapley values for the top eight most relevant reviews in response to the query, "How is the quality of the wireless controller?" for the first product. Other reviews not contributing to this query receive a Shapley value of zero. In this example, we see that Review #3 has the highest Shapley value (1.83), as it directly compares the controller's quality to other versions and emphasizes functionality, aligning well with the prompt's emphasis on "Information Coverage" for quality details. Similarly, Review #7 (1.61) and Review #2 (1.58) score highly for addressing quality explicitly-#7 in a positive tone and #2 by highlighting durability compared to off-brand controllers. Review #5 (1.44) also performs well by underscoring the superior quality of the original controller versus knockoff brands. Review #4 (1.25) is somewhere in the middle, highlighting the good quality but without additional information relevant to the query. The lower-scoring reviews, including Review #1 (0.59) and Review #6 (0.53), just generally mention the great quality and cheap price, lacking specific details. Reviews #8 has the lowest value of 0.17, as it emphasizes aspects like shipping and being good for gifts, which are less relevant to the query's focus on controller quality, though the title mentions the "Quality", which makes it relevant to the query.

No.	Title	Main Text	Shapley
1	Cheaper price, same great quality	This product stands as a testament to the reason I go to the store to find the product then buy it online at a cheaper price.	0.59
2	Quality	It's worth the price. Controllers last much longer than off brand.	1.58
3	Great Quality and Price	Great price and product and unlike others this one worked. Ordered one from ebay and it was garabe but this seller is legit 5 stars.	1.83
4	Great buy and Product is exactly what I expected!	I liked the red color and that the product quality was exactly what I needed!	1.25
5	Five Stars	I only recommend the original makers product, pay more but better then the knockoffs.	1.44
6	great product	Great product and so much cheaper than buying it in store.	0.53
7	Nice, new and crispy	Nice new and crispy! Very happy with the quality, the vendor and the price 10/10 would recommend.	1.61
8	Quality	Very nice and the shipping was very quick. My grandson loved it for Christmas.	0.17

Table 2: Shapley values of Top 8 relevant Amazon reviews for the query "How is the quality of the wireless controller?".

Next, we discuss the cost of implementing the exact Shapley calculation. For each query with 8 relevant reviews, we must process $2^8 - 1 = 255$ distinct subsets, with each subset requiring a summarization and four

evaluations to ensure reliable scoring. We use OpenAI's API to process both summarization and evaluation with GPT-4o, and it costs about $1.30 in OpenAI API fees per query.

To illustrate the scalability challenge, consider Perplexity AI, a leading LLM-based search engine, which serves over 400 million queries per month (Srinivas, 2024). Applying the exact Shapley algorithm to each query for this volume would imply over $520 million in monthly compute costs ($1.3 per query × 400 million queries). Switching to a lower-priced open-sourced LLM model such as `Llama-3-8B-Instruct-Lite` can reduce API costs by approximately 98.4%[12] However, even with Llama 3, the cost to calculate exact Shapley values would still exceed $8.3 million ($520 million per month x (1- 98.4%)) per month, which is still too high compared to Perplexity's annualized revenue of $100 million as of March 2025 (Srinivas, 2025). The scale of these computational expenses creates a significant gap between theoretical document valuation frameworks and practical implementation, highlighting the need for an efficient approximation algorithm like our proposed Cluster Shapley algorithm.

6.4 Cluster Shapley Implementation

For the implementation of our proposed Cluster Shapley algorithm, as described in Algorithm 1, the first step is to specify the clustering diameter hyperparameter ϵ. In our numerical comparison in §7.1, we evaluate a spectrum of ϵ values to illustrate the trade-off between approximation error and computational time. Specifically, we explore ϵ values ranging from 0.01 to 1.00 in increments of 0.025. As discussed in Web Appendix §B.1, one can apply standard hyperparameter tuning procedures to select an appropriate ϵ. To assess the robustness of this choice, we conduct an additional experiment in §7.3 (detailed in Web Appendix §D.4), where we randomly split the test dataset into two subsets-one for tuning ϵ and the other for implementation and evaluation. The results demonstrate that the Cluster Shapley algorithm is fairly robust to the choice of ϵ.

After specifying the clustering diameter ϵ, we apply Algorithm 3 to perform document clustering. Note that the distance matrix used in the algorithm is computed based on Equation (7). To obtain document embeddings, we extract query-relevant information from each comment individually, using the same LLM prompt as in the summarization step but applied to one document at a time. The computational cost of our adaptive clustering algorithm is negligible in our setting, especially when compared to the cost of LLM-based summarization and evaluation. We analyze and report the computation time of the clustering step using Algorithm 3 in Web Appendix §C.5.

Figure 4 presents the clustering results using Algorithm 3 for a sample query. In this instance, $\epsilon = 0.05$, which yields six clusters. Increasing ϵ results in fewer clusters, further reducing computational cost, but may introduce higher approximation error. Even with six clusters, the computational complexity is significantly reduced-shrinking from $2^8 - 1 = 255$ to $2^6 - 1 = 63$, representing a fourfold improvement in efficiency.

In Step 2, we append all the documents within a cluster, treat each cluster as a meta-document, and obtain cluster-level exact Shapley values. At this stage, the summarization and evaluation steps follow exactly the same prompts as those used in the exact Shapley calculation, except that we use the meta-documents at the cluster level as the input for summarization. Finally, we distribute the cluster-level Shapley values equally

[12] GPT-4o costs $1.313 per million tokens compared to just $0.021 per million tokens for Llama 3 (AI/ML API, 2025).

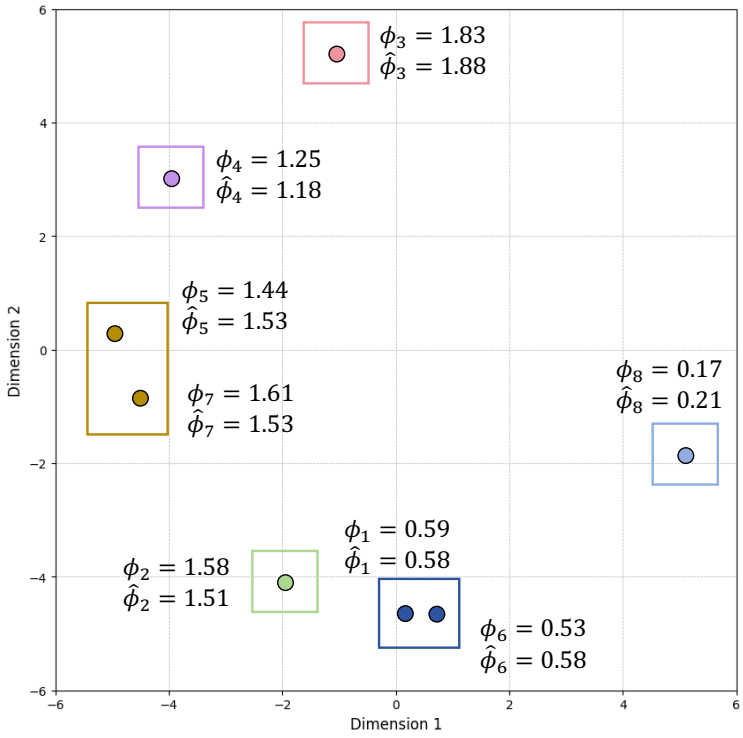


Figure 4: Clustering result of Top 8 relevant Amazon reviews for the query "How is the quality of the wireless controller?" We use 3072-dimensional OpenAI embeddings for the clustering. However, we use PCA to reduce the embedding dimension to 2 for better visualization. Dots represent the reviews, and squares represent clusters. ϕ_i is the exact Shapley value while $\hat{\phi}_i$ is the approximated Shapley value by the Cluster Shapley algorithm.

back to the individual documents within the clusters. As shown in the example in Figure 4, documents within the same cluster have similar exact Shapley values, and our Cluster Shapley approximation achieves high accuracy. For instance, reviews 1 and 6 are in the same cluster with similar exact Shapley values of $\phi_1 = 0.59$ and $\phi_6 = 0.53$, and the same approximated Shapley value $\hat{\phi}_1 = \hat{\phi}_6 = 0.58$. As shown in Table 2, both reviews 1 and 6 emphasize that the price is cheap but only mention that the product is great without details. The mean absolute error (MAE) between the exact and approximated Shapley values across all reviews in this example is only 0.04, demonstrating that our algorithm successfully approximates the Shapley value with a low error and reduces the computation cost. It also validates our theory that semantically similar documents tend to have comparable marginal contributions to the summary, and thus similar Shapley values.

7 Numerical Results

We now present the results from the application of our Cluster Shapley algorithm to the Amazon review setting. First, in §7.1, we report the numerical results of our proposed Cluster Shapley approach and compare its performance with other three off-the-shelf Shapley approximation algorithms. Next in §7.2, we further contrast Cluster Shapley with the simple attribution rules (with minimal computational cost) and discusses the corresponding findings. In §7.3, we present a series of robustness checks and extensions.

7.1 Comparison with Off-the-shelf Shapley Approximation Algorithms

We now introduce three widely used Shapley value approximation algorithms that serve as off-the-shelf benchmarks and compare the performance of our proposed algorithm against them.

- **Monte Carlo:** The Monte Carlo algorithm (or permutation sampling) is a popular approach for approximating Shapley values (Mann and Shapley, 1960). This method randomly samples permutations from the $|S_q|!$ possible combinations of documents and then for each document i and one permutation P_i^π, calculates its marginal contribution, i.e., $v(q, A(q, P_i^\pi \cup \{i\})) - v(q, A(q, P_i^\pi))$. Shapley value can then be approximated using the sample average of marginal contributions over all sampled permutations. As the number of permutation samples increases, the approximation error decreases, but the computational cost grows linearly. In our numerical experiments, we progressively increase the number of permutations to show the trade-off between accuracy and efficiency.
- **Truncated Monte Carlo:** The algorithm accelerates Shapley value calculation by adaptively reducing the number of evaluated samples. This method operates under the idea that the score function is non-decreasing, i.e., $v(q, A(q, S_1 \cup \{i\})) - v(q, A(q, S_1)) \leq v(q, A(q, S_2 \cup \{i\})) - v(q, A(q, S_2))$ if $S_2 \subseteq S_1$, i.e., the marginal contribution of document i decreases when more documents come into the permutation. This is because, with a larger set of permutations, document i is more likely to have higher overlapping information with other documents, reducing its marginal contribution.

  We briefly summarize the algorithm here and refer readers to Ghorbani and Zou (2019) for details. This algorithm randomly samples a permutation of reviews and sequentially calculates performance scores, v, by adding reviews in the permutation order. Since these scores are increasing, the algorithm truncates the computation by assigning zero marginal contributions to the remaining reviews when the gap between the current score and the maximum score (10 in our setting) is smaller than a pre-specified threshold, called *performance tolerance*. It basically means that when adding the remaining reviews, their marginal contributions are always smaller than this threshold. Thus, this algorithm simply assigns zero marginal contribution instead of calculating the negligible marginal value.

  The performance tolerance parameter, which governs the allowable change in Shapley values before truncation occurs, is tuned over a range of values – $\{0.1, 0.2, 0.3, 0.5, 0.7, 1, 2, 3\}$ – to balance estimation accuracy and computational efficiency. Smaller values reduce the effectiveness of truncation-causing Truncated Monte Carlo to behave similarly to standard Monte Carlo-while larger values result in early truncation, degrading estimation accuracy. Based on the tuning, we select 0.5 for our main experiments. After fixing the performance tolerance at 0.5, we vary the number of sampled permutations to construct the efficient frontier, which illustrates the trade-off between computational cost and approximation error.
- **Kernel SHAP:** Kernel SHAP is a model-agnostic approach to approximating Shapley values based on weighted least squares regression (Lundberg and Lee, 2017). Our implementation uses Python's SHAP package, which we adapt specifically for our LLM-based summarization task by implementing a custom mapping function between subset compositions and their corresponding summarization scores. The method employs KernelExplainer with an identity link function and L1 regularization to enhance

numerical stability. We tested increasing numbers of samples to evaluate the performance of the method under different computational budgets. Kernel SHAP has been widely used across domains, including NLP for transformer-based model interpretation (Kokalj et al., 2021), finance for credit risk analysis (Famà et al., 2024), healthcare for clinical decision support (Li et al., 2022), and marketing for optimizing content engagement (Kong et al., 2023). Recent work has pointed out some limitations of Kernel SHAP, especially when interactions are strong (Gosiewska and Biecek, 2019; Ragodos et al., 2024).[13] Nevertheless, given that it is one of the most widely used Shapley approximations used in practice, we include it in our evaluation for benchmarking purposes.

We now present the results from the empirical exercise comparing the performance of our algorithm with the above three algorithms. Our experiments include the 48 test queries described in §5. Each query comprises the eight most relevant reviews selected from the Amazon review dataset, forming the foundation for our comparative analysis of various Shapley value approximation algorithms.

To establish a stable evaluation baseline and reduce variance introduced by the summarization and evaluation steps, we standardize the process as follows: for each query, we generate a single summary for each subset (for all 255 possible subsets) and fix the evaluation score for each summary by averaging four evaluations, as detailed in §6.2. By fixing sample paths, we mitigate the inherent randomness in LLM outputs, ensuring consistent baseline measurements across different approximation methods.

We visualize the efficient frontier of different Shapley value approximation methods in Figure 5. The y-axis represents the Mean Absolute Error (MAE) of the Shapley values, averaged across all test instances and reviews, which serves as a measure of the approximation error for each algorithm. Results under performance metrics, including Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE), exhibit similar trends and can be found in Web Appendix §D.2. The x-axis represents the number of unique subsets used by the algorithms. Here, a "unique subset" refers to a distinct (non-replicated) subset of reviews used in the algorithm. For Cluster Shapley, this number represents the distinct cluster subsets that emerge after clustering, i.e., $2^m - 1$ averaged over 48 test instances. For Monte Carlo, Truncated Monte Carlo, and Kernel SHAP, while these methods can sample the same subset multiple times, we count only the unique subsets encountered during sampling to ensure fair computational comparison. For example, if the same subset appears multiple times in these algorithms, we only evaluate it once and cache its evaluation score for reuse. Because the computation time for clustering in the Cluster Shapley algorithm, the additional time for evaluating larger meta-reviews, and the regression step in Kernel SHAP are all negligible compared to the cost of summarization and evaluation prompts, the overall computation cost of all algorithms effectively scales linearly with the number of unique subsets. Thus, this figure highlights the cost-effectiveness of the various algorithms, where cost is represented by the x-axis and effectiveness by the y-axis. The lower left region indicates more desirable outcomes-lower cost and higher effectiveness. For clarity, Figure 5 truncates the x-axis at 180 subsets rather than showing the full range up to 255. This choice reflects two considerations.

[13] Note that the issues pointed out by these papers relate to the approximation process used in the Kernel SHAP algorithm, and not the Shapley concept itself.

First, all algorithms perform similarly beyond this point, and extending the axis would not add meaningful insight. Second, larger subset sizes imply higher computation costs, which are less relevant to practical applications. As such, our analysis focuses on the range where performance differences are most informative.

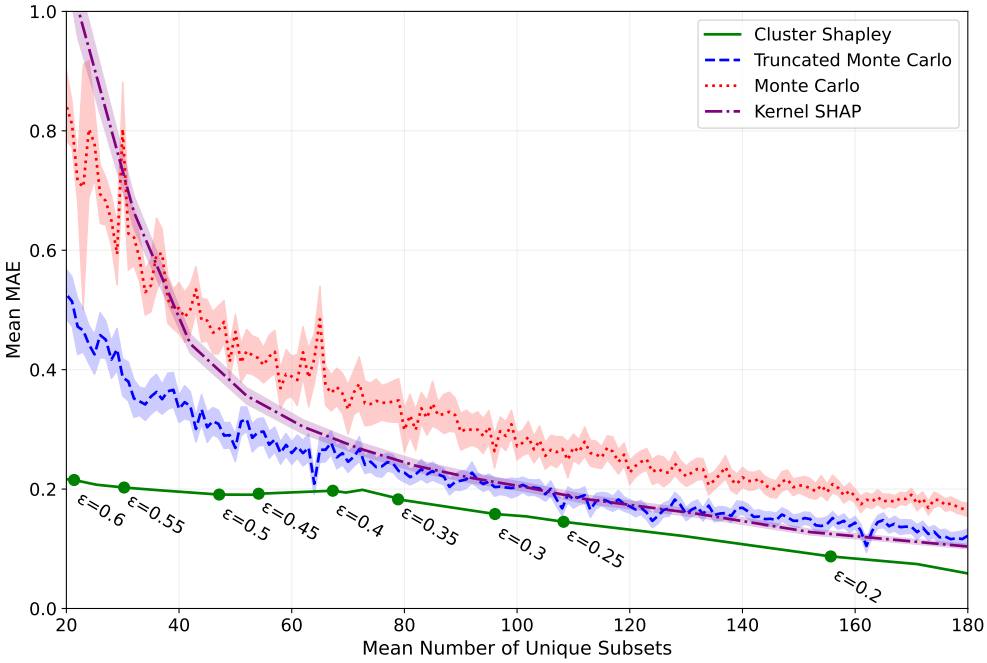


Figure 5: Efficient frontiers of Shapley approximation algorithms. The x-axis represents the number of unique subsets used by the algorithms, averaged across all test queries and reviews. The y-axis represents the Mean Absolute Error (MAE) of the Shapley values, averaged across all test queries and reviews. The points on the Cluster Shapley curve correspond to different clustering diameters ϵ. For reference on the size of MAE, the average Shapley value over all test samples is 1.084, indicating that 0.2 MAE is around a 20% percentage error. 95% CIs for Monte Carlo, Truncated Monte Carlo, and Kernel SHAP are computed through 10 replications of the algorithms.

We now discuss the main takeaways from Figure 5. First, we see that Cluster Shapley achieves the best overall performance across all algorithms. Its error curve consistently lies below those of other algorithms, indicating that it achieves lower approximation error for a given computation cost or, equivalently, requires less computation to reach the same level of accuracy. Truncated Monte Carlo ranks second, benefiting from its early stopping mechanism, which limits unnecessary evaluations when performance plateaus. Second, Cluster Shapley's advantage is particularly pronounced when the computation budget is small, i.e., the number of unique subsets is relatively small. In this regime, it achieves an MAE of around 0.2, whereas other methods exceed 0.4. This efficiency arises from clustering's ability to capture core similarity patterns with relatively few clusters, which effectively reduces the dimensionality of the approximation problem. However, as the number of subsets increases (i.e., with smaller ϵ), the advantage gradually diminishes. When the subset count exceeds 150, Cluster Shapley's performance converges with other methods as single-review clusters emerge and the benefit of clustering is lost. Third, all algorithms show decreasing MAE as the number of subsets increases, consistent with theoretical expectations. However, their stability varies substantially. Monte Carlo

and Truncated Monte Carlo exhibit noticeable fluctuations and wide confidence intervals due to their inherent sampling variability. In contrast, Cluster Shapley produces deterministic results across the computation range because we calculate the exact Shapley of clusters in Step 2. Kernel SHAP also uses sampling but reduces variance through its linear regression-based estimation, resulting in moderate stability.

In summary, our empirical results show that the proposed Cluster Shapley algorithm achieves substantially higher accuracy at lower computational costs by leveraging semantic similarity in LLM-based document embeddings. Unlike other approximation methods-such as Monte Carlo, Truncated Monte Carlo, and Kernel SHAP-that rely on random sampling without exploiting intrinsic semantic relationships, Cluster Shapley groups semantically similar documents to approximate cooperative contributions more efficiently. This scalability provides practitioners with a flexible toolkit to balance accuracy and computation cost according to platform constraints. More broadly, these findings highlight the value of incorporating semantic representations from LLMs into document valuation frameworks, enabling practical and principled approaches to fair value attribution at scale.

7.2 Comparison with Simple Attribution Rules

So far, we saw that Cluster Shapley method achieves the best efficiency–accuracy frontier among all Shapley approximation algorithms. Nevertheless, one might argue that simple attribution rules that are not based on Shapley can be both cheap and effective. To that end, we now consider two simple attribution rules – equal attribution and relevance-weighted attribution, and benchmark the performance of our approach against them.

- **Equal Attribution:** This baseline splits the full-set value uniformly across documents, i.e., all documents in S_q get equal attribution $\hat{\phi}_i^{\mathrm{Equ}}(q) = v(q, A(q, S_q))/|S_q|$. This baseline is simple to apply in practice, similar in spirit to Cloudflare's pay-per-crawl approach (Zeff, 2025).
- **Relevance-Weighted Attribution:** This baseline allocates $v(q, A(q, S_q))$ proportionally to the relevance score between the query and each document:

$$s_i(q) \;=\; \text{cosine_similarity}(e_i, e_q), \qquad w_i(q) \;= s_i(q) / \sum_{j \in S_q} s_j(q)$$

$$\hat{\phi}_i^{\mathrm{Rel}}(q) \;=\; v(q, A(q, S_q))\, w_i(q), \qquad i \in S_q.$$

These two simple rules are costless to compute but can introduce substantial bias, as their assigned document values deviate widely from the fair values defined by Shapley. Such comparisons are crucial because platforms must weigh fairness against computational feasibility. In practice, a simple rule such as equal attribution can appear attractive for its negligible cost, as in Cloudflare's pay-per-crawl model (Zeff, 2025), yet understanding how large the fairness loss is essential for responsible design.

Table 3 quantifies this fairness–efficiency trade-off. Panel A reports attribution accuracy relative to the exact Shapley baseline, together with normalized computation cost (where 1 corresponds to the cost of computing all 255 subsets). Panel B displays the corresponding results for Cluster Shapley across different clustering diameters ϵ, listed from coarse to fine (high ϵ to low), highlighting how accuracy improves as

computational investment increases.

The results exhibit a clear pattern: simpler attribution rules incur negligible computational cost but suffer from substantial bias, whereas more accurate methods require greater computation. In Panel A, equal attribution yields an MAPE exceeding 150%, indicating a large deviation from fair allocation, while relevance-weighted attribution achieves a lower but still significant 31.93%. By contrast, even a very coarse Cluster Shapley with $\epsilon = 0.70$ achieves a smaller MAPE of 26.49% at an extremely low cost—only 0.02 times that of the exact Shapley (roughly 3.5 out of 255 subset evaluations). Panel B further illustrates the fairness–efficiency trade-off within Cluster Shapley: as ϵ decreases, computational cost increases while accuracy improves. For instance, with $\epsilon = 0.20$, Cluster Shapley achieves an MAPE of 11.85% at 0.60 times the computational cost of the exact Shapley. These findings quantify a tangible "cost of fairness": achieving fair value attribution demands meaningful, though not prohibitive, computation.

For platform decision-makers, these results provide actionable guidance. When computational budgets are tight, the framework helps quantify how much fairness must be traded off for efficiency. Conversely, when fairness toward content contributors is a design priority, the same analysis clarifies the computational investment required. Although our experiments rely on limited or synthetic query data, the observed bias is likely heterogeneous across query types; once real user-level query distributions become available, the same framework can readily quantify this heterogeneity at scale.

Finally, Cluster Shapley should be viewed as a proof of concept rather than a prescriptive algorithm for deployment. Adapting it to specific LLM-based summarization architectures—or replacing it with improved approximation algorithms—may further enhance the efficiency frontier. The central insight is conceptual: the Shapley-based framework offers a principled way to evaluate and compare the computational cost of fairness in document-value attribution.

7.3 Robustness Checks and Extensions

We now present a series of robustness checks on various aspects of our approach and an extension that combines Cluster Shapley with other approximation algorithms.

- In the main analysis, we use GPT-4o for both summarization and for evaluating the summaries. This can potentially introduce biases because LLMs tend to give higher scores to their own summaries. To address this, we conduct an analysis where we use a different LLM, Claude, for evaluation. We find that Claude yields similar evaluation results and therefore similar Shapley values. See Web Appendix §D.1 for details.
- For completeness, we also compare the approximation error of different algorithms using alternative metrics, including MSE and MAPE, and find that the results are consistent with those shown in Figure 5. See Web Appendix §D.2 for additional details of this robustness check.
- In the main analysis, we use our proposed adaptive DBSCAN (Algorithm 3) to enforce tight distance constraints within clusters. To examine the impact of this design choice, we conduct a robustness check using the standard (non-adaptive) DBSCAN algorithm. We find that while standard DBSCAN performs reasonably well, its performance is consistently inferior to our proposed adaptive version. Details of this comparison are provided in Web Appendix §D.3.

Panel A: Accuracy–Efficiency Comparison of Simple Attribution Rules and Exact Shapley				
Method	**MAE**	**MSE**	**MAPE**	**Computation Cost**
Exact Shapley	0	0	0	1.00
Equal Attribution	0.3284	0.3955	157.86%	0.00
Relevance-Weighted	0.3374	0.3893	31.93%	0.00
Panel B: Cluster Shapley's Accuracy–Efficiency Frontier Across ϵ				
Cluster Shapley ϵ	**MAE**	**MSE**	**MAPE**	**Computation Cost**
0.70	0.2305	0.1636	26.49%	0.02
0.60	0.2152	0.1038	24.43%	0.08
0.50	0.1908	0.1074	21.05%	0.18
0.40	0.1972	0.1499	21.35%	0.26
0.30	0.1617	0.1723	17.16%	0.38
0.20	0.0913	0.0441	11.85%	0.60
0.10	0.0507	0.0184	8.62%	0.73
0.01	0.0381	0.0148	7.47%	0.77

Table 3: Approximation error (averaged over all documents) and computation cost of simple attribution rules (Panel A) and Cluster Shapley under varying ϵ (Panel B). The last column, computation cost, is normalized relative to the exact Shapley, which requires evaluating all 255 unique subsets (assigned value 1). Other methods' costs are expressed as a fraction of this baseline. Equal and relevance-weighted rules only require $v(q, A(q, S_q))$, hence their cost is close to 0 ($1/255 \approx 0.00$). Note that when calculating MAPE, we add a small constant (0.1, approximately 10% of the mean Shapley value) to the denominator to prevent near-zero Shapley values from inflating the error.

- In the main analysis, we use all 48 test queries and report average performance across them. To assess robustness to query selection and hyperparameter tuning, we conduct a sample-splitting check by randomly dividing the queries into two halves and replicating the analysis for each split. The results, shown in Web Appendix §D.4, confirm consistent comparative performance across splits and stable hyperparameter choices.
- As discussed in §4.4.3, in such large data settings, we can integrate Monte Carlo sampling into our Cluster Shapley to improve efficiency. We present additional experiments in Web Appendix §D.5 demonstrating the performance of our approach for such settings. The first experiment considers a setting where the number of relevant documents $S_q = 10$. This setup reflects more computationally intensive, yet still feasible, scenarios in which exact Shapley values can be computed. All other settings are consistent with the main analysis. The second experiment involves a setting where the number of relevant documents is extremely large, $S_q = 30$. This number is unlikely to be commonly used in practice; however, we consider this case as a proof of concept to demonstrate the scalability of our approach. For example, Gupta (2025) reports that Google AI Overviews include 4–5 citations on average, with a maximum of 33 in rare cases. Since computing exact Shapley values is intractable at this scale, we synthetically construct ground-truth Shapley values and simulate the evaluation function $v(q, A)$. Across both experiments, the Monte Carlo–based Cluster Shapley demonstrates favorable accuracy–cost trade-offs, reinforcing its practicality in realistic and large-scale applications.

8 Conclusion

The rapid integration of LLM-based summarization and search technologies into digital platforms has reshaped how information is consumed, shared, and monetized. While these advances improve user experience and platform engagement, they also raise serious concerns around attribution, compensation, and sustainability for content creators. This paper addresses a critical and timely challenge in this evolving ecosystem: how to fairly and efficiently value the contribution of individual documents to LLM-generated summaries.

Our work makes three core contributions. First, we propose a principled framework for fair document valuation based on Shapley values, offering a theoretically grounded approach that satisfies key desiderata such as fairness, generality, and scalability. This framework can operate across different summarization methods and evaluation mechanisms, making it broadly applicable across search, review, and Q&A platforms. Second, we introduce Cluster Shapley, a novel approximation algorithm that leverages semantic similarity between documents to reduce the computational burden of traditional Shapley value computation. This method enables a flexible trade-off between efficiency and accuracy via a tunable clustering parameter and is supported by theoretical guarantees. Third, using a real-world case study based on Amazon product reviews, we demonstrate the practical effectiveness of our algorithm: it achieves up to a 40% reduction in computation while keeping the MAPE below 20%, and consistently outperforms existing benchmark methods.

While our proposed framework represents a significant step forward and provides a proof-of-concept pipeline using a standard RAG system, several important directions remain open for future research. A key avenue is to adapt the framework to real-world LLM summarization systems, which often feature richer retrieval stacks, reranking modules, tool use, and reasoning capabilities. It would also be interesting to explore more dynamic and fine-grained valuation approaches that account not only for document content but also for metadata such as authorship, credibility, and temporal relevance. Moreover, extending the framework to multi-turn interactions and conversational query contexts, where user queries evolve over time, could also be interesting.

Another promising direction is to consider how document value attribution systems can be combined with reputation mechanisms. While a large stream of theoretical and empirical research has established the importance of reputation mechanisms in sustaining two-sided platforms (Resnick et al., 2000; Cripps et al., 2004; Yoganarasimhan, 2013), it is unclear how these systems can survive without direct feedback on contributor's performance. Thus, extending the document valuation framework to augment/replace existing reputation systems can be an important next step.

In summary, our work provides a robust foundation for fair content attribution in the generative AI ecosystem and opens up new possibilities for designing sustainable ecosystems that balance platform goals with content creator incentives.

Funding and Competing Interests Declaration

Author(s) have no competing interests to declare.

References

AI/ML API. AI/ML API inference pricing, 2025. Link. Accessed on May, 2025.

Yoram Bachrach, Evangelos Markakis, Ezra Resnick, Ariel D Procaccia, Jeffrey S Rosenschein, and Amin Saberi. Approximating power indices: theoretical and empirical analysis. *Autonomous Agents and Multi-Agent Systems*, 20 (2):105–122, 2010.

Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pages 1899–1909. PMLR, 2020.

Sarah Carroll. Will google's ai overview kill web traffic?, January 2025. Link. Accessed on May, 2025.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & operations research*, 36(5):1726–1730, 2009.

Ashley Chang. How i became yelp elite in 48 days, August 2023. Link. Accessed on May, 2025.

Foo Yun Chee. Exclusive: Google's AI Overviews hit by EU antitrust complaint from independent publishers, 2025. Link. Accessed on Nov, 2025.

Magie Cheng, Elie Ofek, and Hema Yoganarasimhan. Balancing engagement and polarization: Multi-objective alignment of news content using llms. *arXiv preprint arXiv:2504.13444*, 2025.

Judith A Chevalier and Dina Mayzlin. The effect of word of mouth on sales: Online book reviews. *Journal of marketing research*, 43(3):345–354, 2006.

R Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18, 1977.

Kevin Corder and Keith Decker. Shapley value approximation with divisive clustering. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 234–239. IEEE, 2019.

Martin W Cripps, George J Mailath, and Larry Samuelson. Imperfect monitoring and impermanent reputations. *Econometrica*, 72(2):407–432, 2004.

Goodwin Danny. 60% of perplexity citations overlap with top 10 google organic results, 2024. Link. Accessed on July, 2025.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231. AAAI Press, 1996.

Angelo Famà, Jurgena Myftiu, Paolo Pagnottoni, and Andrea Spelta. Explainable machine learning for financial risk management: Two practical use cases. *Statistics*, 2024.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501, 2024.

Shaheen S Fatima, Michael Wooldridge, and Nicholas R Jennings. A linear approximation method for the Shapley value. *Artificial Intelligence*, 172(14):1673–1699, 2008.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.

Danny Goodwin. Google ceo says ai overviews are increasing search usage, April 2024. Link. Accessed on May, 2025.

Google. AI overviews – search anything, effortlessly, 2025. Link. Accessed on May, 2025.

Alicja Gosiewska and Przemyslaw Biecek. Do not trust additive explanations. *arXiv preprint arXiv:1903.11420*, 2019.

Michael M. Grynbaum and Ryan Mac. The new york times sues OpenAI and Microsoft over content use, 2023. Link. Accessed on May, 2025.

Han Guo, Nazneen Fatema Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*, 2020.

Pragati Gupta. 40.58% of AI citations come from google's top 10 results (study of 1m+ ai overviews), 2025. Link. Accessed on July, 2025.

Xiaochuang Han, Byron C Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. *arXiv preprint arXiv:2005.06676*, 2020.

Tanjim Ul Haque, Nudrat Nawal Saber, and Faisal Muhammad Shah. Sentiment analysis on large scale amazon product reviews. In *2018 IEEE international conference on innovative research and development (ICIRD)*, pages 1–6. IEEE, 2018.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2019.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495.

Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed Chi, and Derek Zhiyuan Cheng. Do llms understand user preferences? evaluating llms on user rating prediction. *arXiv preprint arXiv:2305.06474*, 2023.

Enja Kokalj, Blaž Škrlj, Nada Lavrač, Senja Pollak, and Marko Robnik-Šikonja. Bert meets shapley: Extending shap explanations to transformer-based classifiers. In *EACL Hackashop on Explainability for NLP*, 2021.

Fanjie Kong, Yuan Li, Houssam Nassif, Tanner Fiez, Ricardo Henao, and Shreya Chakrabarti. Neural insights for digital marketing content design. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2023.

Joshua A Kroll, Joanna Huey, Solon Barocas, Edward W Felten, Joel R Reidenberg, David G Robinson, and Harlan Yu. Accountable algorithms. *UNIVERSITY of PENNSYLVANIA LAW REVIEW*, pages 633–705, 2017.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Anthony Li, Ming Lun Ong, Chien Wei Oei, Weixiang Lian, Hwee Pin Phua, Lin Htun Htet, Wei Yen Lim, and Mehul Motani. Unified auto clinical scoring (uni-acs) with interpretable ml models. In *Machine Learning for Healthcare Conference (MLHC)*, 2022.

Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3): 225–331, 2009.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *The 31st International Conference on Neural Information Processing Systems*, pages 4768–4777. PMLR, 2017.

Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based shapley value approximation. *arXiv preprint arXiv:1306.4265*, 2013.

Irwin Mann and Lloyd S. Shapley. Values of large games, IV: Evaluating the Electoral College by Montecarlo Techniques. Technical Report RM-2651, RAND Corporation, Santa Monica, CA, 1960.

Rajiv Mehta and Trishul Chilimbi. Amazon announces Rufus, a new generative AI-powered conversational shopping experience, 2024. Link. Accessed on May, 2025.

Microsoft. Copilot search in bing, 2025. Link. Accessed on May, 2025.

Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

OpenAI. Introducing ChatGPT search, 2024. Link. Accessed on May, 2025.

Rajvardhan Patil, Sorio Boit, Venkat Gudivada, and Jagadeesh Nandigam. A survey of text representation and embedding techniques in NLP. *IEEE Access*, 2023.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Perplexity AI, 2025. Link. Accessed on May, 2025.

ProRata AI. ProRata AI, 2025. Link. Accessed: Aug, 2025.

Xiao Pu, Mingqi Gao, and Xiaojun Wan. Summarization is (almost) dead. *arXiv preprint arXiv:2309.09558*, 2023.

Ronilo Ragodos, Tong Wang, Lu Feng, et al. From model explanation to data misinterpretation: Uncovering the pitfalls of post hoc explainers in business research. *arXiv preprint arXiv:2408.16987*, 2024.

Reddit Help. What is karma?, November 2024. Link. Accessed on May, 2025.

Elizabeth Reid. Generative AI in search: Let google do the searching for you, May 2024. Link. Accessed on May, 2025.

Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

Katie Robertson. Openai strikes a deal to license news corp content, May 2024. Link. Accessed on May, 2025.

Marijn Sax and Hao Wang. From threat to opportunity: Gaming the algorithmic system as a service. *Internet Policy Review*, 14(2), 2025.

Vaughn Schermerhorn. How amazon continues to improve the customer reviews experience with generative AI, 2023. Link. Accessed on May, 2025.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya Parameswaran, and Ian Arawjo. Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–14, 2024.

Lloyd S Shapley. A value for n-person games. *Contribution to the Theory of Games*, 2, 1953.

Kara Sherrer. Google: It's 'misleading' for websites to blame ai overviews for lost traffic, April 2025. Link. Accessed on May, 2025.

Aravind Srinivas. Perplexity serving over 100 million queries per week, April 2024. Link. Accessed on May, 2025.

Aravind Srinivas. Perplexity has crossed $100m in annualized revenue, March 2025. Link. Accessed on May, 2025.

Universal Music Group. ProRata invents generative ai attribution technology to compensate and credit content owners while facilitating fairness and fact, with support from universal music group, 2024. Link. Accessed: Aug, 2025.

Jiachen T Wang, Zhun Deng, Hiroaki Chiba-Okabe, Boaz Barak, and Weijie J Su. An economic solution to copyright challenges of generative ai. *arXiv preprint arXiv:2404.13964*, 2024.

Ruiyun Xu, Yue Feng, and Hailiang Chen. Chatgpt vs. google: A comparative study of search performance and user experience. *arXiv preprint arXiv:2307.01135*, 2023.

Zikun Ye, Hema Yoganarasimhan, and Yufeng Zheng. LOLA: Llm-assisted online learning algorithm for content experiments. *Forthcoming in Marketing Science*, 2025.

Hema Yoganarasimhan. The value of reputation in an online freelance marketplace. *Marketing Science*, 32(6):860–891, 2013.

Hema Yoganarasimhan. Search personalization using machine learning. *Management Science*, 66(3):1045–1070, 2020.

Maxwell Zeff. Cloudflare launches a marketplace that lets websites charge AI bots for scraping, 2025. Link. Accessed: Aug, 2025.

Shuo Zhang, Boci Peng, Xinping Zhao, Boren Hu, Yun Zhu, Yanjia Zeng, and Xuming Hu. Llasa: Large language and e-commerce shopping assistant. *arXiv preprint arXiv:2408.02006*, 2024.

Zhange Zhang, Yuqing Ma, Yulong Wang, Shan He, Tianbo Wang, Siqi He, Jiakai Wang, and Xianglong Liu. Lexical diversity-aware relevance assessment for retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27758–27781, 2025.

Yuting Zhu. Generative search: Evidence from a large-scale field experiment. *Available at SSRN 5297194*, 2025.

Web Appendix

A Illustration Examples of AI-Generated Summaries

We now present more examples of AI/LLM-generated summaries, showing how it is changing the traditional search industry, Q&A, and e-commerce websites.

Figure A1 illustrates Google's AI Overview in response to the user query "How to train for a 5K in a month for beginners." The system returns a detailed 4-week training plan synthesized from multiple web pages, with reference links displayed on the right-hand side. Each step in the summary also includes links to the original sources, enabling users to verify the information and credit the content providers. In contrast, Figure A2 shows a traditional Google Search results page, where users must manually click through a ranked list of relevant websites to extract and compile information on their own.

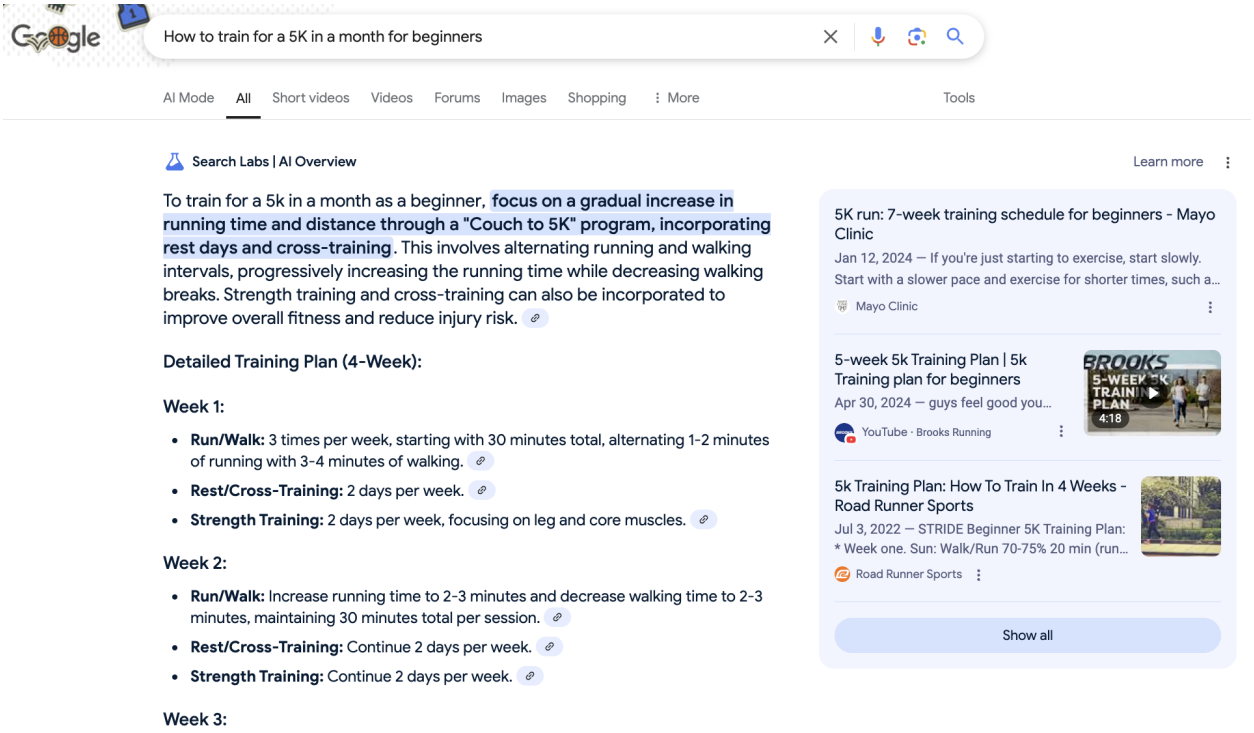


Figure A1: Google AI Overview

The next example, shown in Figure A3, comes from Reddit, a Q&A forum. Reddit is currently piloting a new RAG-enhanced Q&A assistant called *Reddit Answers* (Reddit, 2025). Using the same query, "How to train for a 5K in a month for beginners?", Reddit Answers generates a summarized response based on user-generated content from relevant Reddit threads. Similar to Google's AI Overview, the summary is accompanied by links to the original responses, though the content is limited to the Reddit platform.

AI assistants are also increasingly being integrated into e-commerce platforms. For example, Amazon now provides summarized product reviews directly on the product page, as shown in Figure 1. Similarly, Best Buy offers the same function, as illustrated in Figure A4. In this iPhone example, the summary highlights user mentions of the improved camera, the new camera button, and anticipation for future updates related to Apple Intelligence. In addition to overall review summaries, Amazon's AI assistant, Rufus, enables users

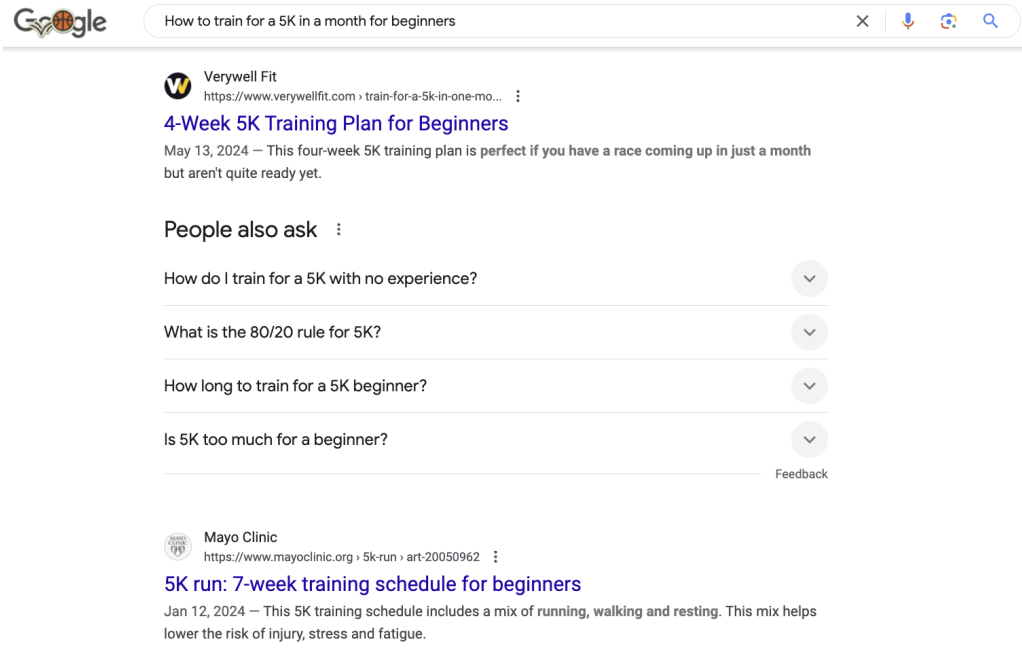


Figure A2: Google Search without AI Overview

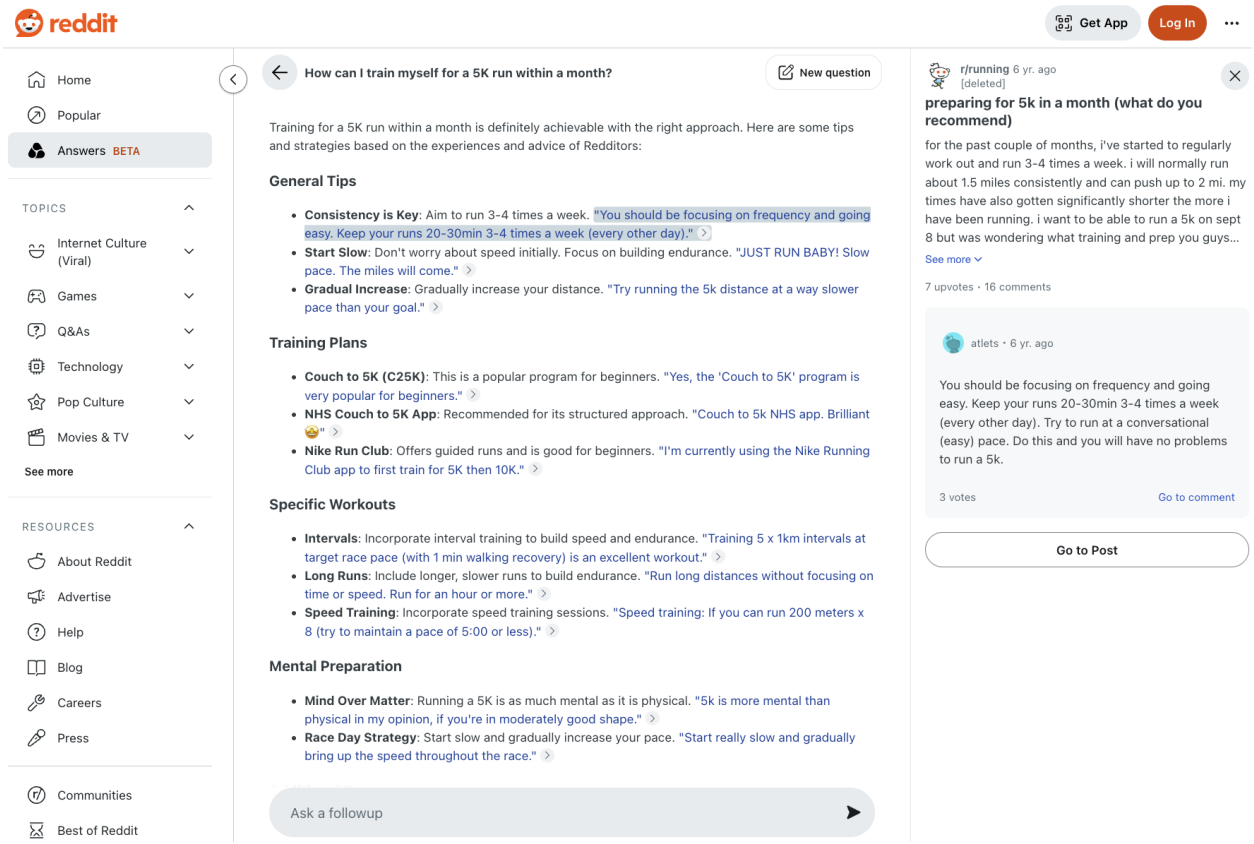


Figure A3: Reddit AI-Generated Answer

to ask specific questions about products. As shown in Figure A5, users can inquire about price history, product features, customer reviews, or comparisons with other products, and receive responses grounded in the information available on the product page.

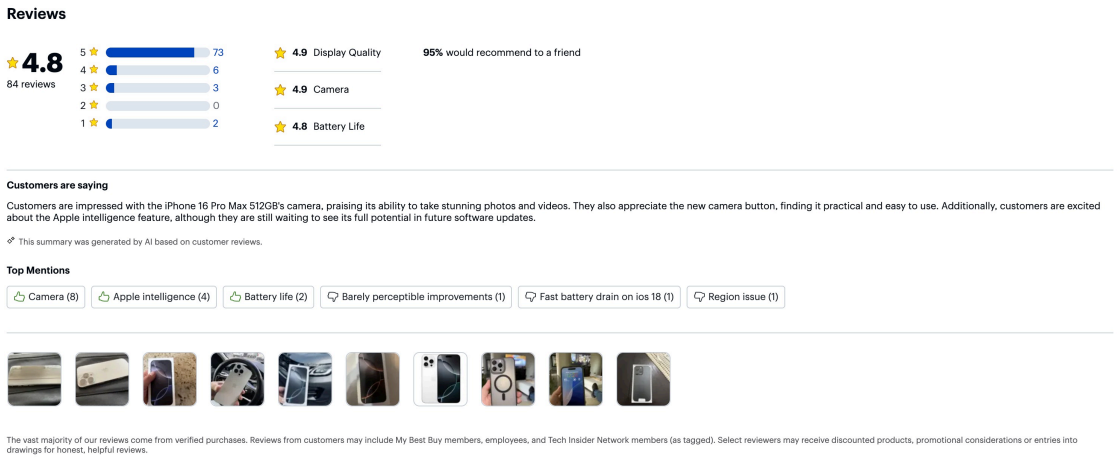


Figure A4: Best Buy AI-Generated Product Review

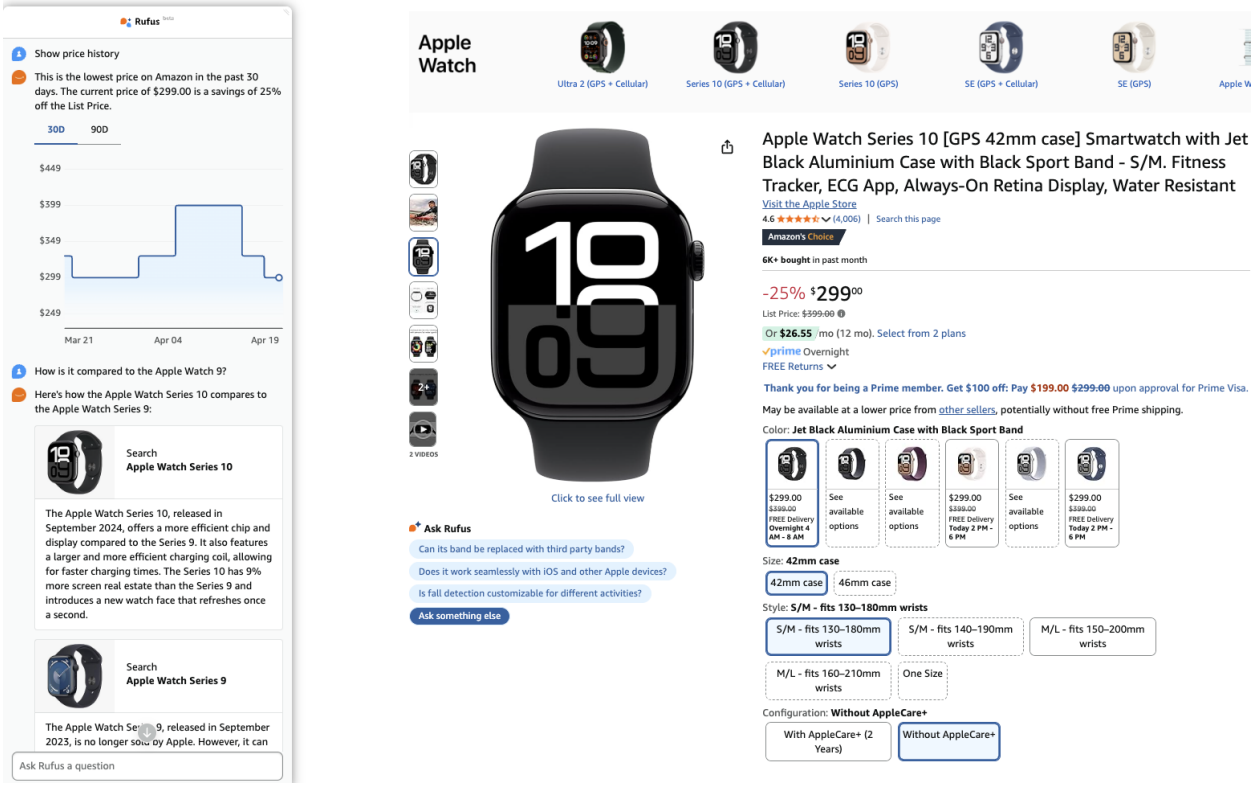


Figure A5: Amazon AI-Powered Shopping Assistant - Rufus

B Supplementary Materials to Section 4

B.1 Document Clustering with the Distance Constraint

We now discuss the document clustering step (Step 1). Note that the Standard clustering might be problematic for our purpose: the theoretical performance of the Cluster Shapley algorithm relies on a Lipschitz continuity assumption (Assumption 1), which bounds the Shapley value differences between documents by their embedding distance. Without bounding the pairwise distance within each cluster, we cannot guarantee that documents in the same cluster will have similar Shapley values-violating the condition needed to ensure our approximation guarantees hold. Therefore, unlike the standard clustering algorithm, we impose a distance constraint that documents within the same cluster should be strictly close to each other, with a distance less than ϵ; that is, if $i, j \in G_k$, then $d(e_i, e_j) \leq \epsilon$. We also find that this constraint can improve the empirical performance of the Cluster Shapley algorithm; see more details in Web Appendix §D.3. Intuitively, a smaller ϵ results in more clusters (larger m), leading to a more accurate Shapley estimation at the cost of increased computation. In the extreme case, we can set ϵ to be the smallest distance between any pair of documents, $\epsilon < \min_{i,j \in S_q, i \neq j} d(e_i, e_j)$, which yields clusters where each cluster contains only one document. In this case, our proposed algorithm reduces to the exact Shapley calculation. Therefore, tuning ϵ appropriately is essential, as it balances the trade-off between computational efficiency and the approximation error induced by clustering. We formalize this statement in §4.4.

To achieve this clustering goal, we propose Algorithm 3, which is essentially an adaptive version of the DBSCAN algorithm. The main advantage of DBSCAN is that it is non-parametric and clusters documents based on density rather than requiring parameters like the number of clusters (which algorithms such as K-Means need). This makes DBSCAN particularly suitable for our task, where the number of clusters is not predetermined. The standard DBSCAN operates through a density-based clustering mechanism, utilizing two key hyperparameters: r (the neighborhood radius) and `MinPts` (minimum points required to form a dense region). The algorithm identifies core points as those having at least `MinPts` points within their r-neighborhood and constructs clusters through density-reachability – a property where points are connected through a chain of core points. Points that fall within the r-neighborhood of a core point but do not qualify as core points themselves are classified as border points, while points that fulfill neither criterion are designated as noise. However, the standard DBSCAN does not guarantee that any two documents within the same cluster have strictly smaller distances than r because DBSCAN forms clusters based on local density connectivity rather than enforcing global distance constraints. Consider three documents i, j, and k: if $d(e_i, e_j) \leq r$ and $d(e_j, e_k) \leq r$, DBSCAN will assign all three points to the same cluster through density-reachability, even if $d(e_i, e_k) > r$. This transitive clustering property can result in clusters where some document pairs exceed the r threshold. It means that if we set the radius the same as our clustering diameter ϵ, the distance between two documents within the same cluster may exceed ϵ. This limitation necessitates modifications to the standard DBSCAN algorithm to enforce a global distance constraint for accurate Shapley value estimation in our context.

In our proposed Algorithm 3, we calculate the distance matrix using document embeddings and a predefined distance function d, and input this matrix into the DBSCAN algorithm. We set the minimum number of points per cluster to 1 (i.e., a DBSCAN hyperparameter `MinPts` $= 1$) to ensure no document is excluded as noise. However, standard DBSCAN does not guarantee that all documents within a cluster are within ϵ (the distance constraint we want to satisfy), nor does it ensure sufficient separation between clusters. To address this, we distinguish between two thresholds: the clustering diameter ϵ, which defines our desired upper bound on intra-cluster distances, and the DBSCAN neighborhood radius r, which governs the clustering procedure. Initially, we set $r \leftarrow \epsilon$, run DBSCAN, and then check whether all document pairs within each cluster satisfy the global diameter constraint $d(e_i, e_j) \leq \epsilon$. If any cluster violates this condition,

we adaptively tighten the local neighborhood by reducing r by a factor of $\alpha = 0.95$, i.e., $r \leftarrow 0.95r$, and rerun DBSCAN until all clusters satisfy the diameter constraint.

Algorithm 3 Adaptive Distance-Constrained DBSCAN

Input: Distance matrix M with $M_{ij} = d(e_i, e_j)$. Hyperparameters: clustering diameter ϵ defined in Cluster Shapley Algorithm, neighborhood radius r, $\texttt{MinPts} = 1$, scaling factor $\alpha = 0.95$

Initialize the neighborhood radius $r \leftarrow \epsilon$ ▷ Start with the original ϵ

while true **do** ▷ Iterate until all clusters satisfy the distance constraint

Run the standard DBSCAN with `MinPts` $= 1$:

- For each document i: find r-neighborhood $N_r(i) = \{j : M_{ij} \leq r\}$.
- Connect points that are within r distance.

Check the distance constraint:

- Check all clusters: $d(e_i, e_j) \leq \epsilon$ for all documents i, j in the same cluster.
- If all clusters satisfy the distance constraint, exit the loop. Otherwise, update $r \leftarrow \alpha \cdot r$ and continue the loop.

end while

Output: Return clusters such that $d(e_i, e_j) \leq \epsilon$ for all documents i, j in the same cluster.

B.2 Empirical Validation of Lipschitz Continuity Assumption

In this appendix, we empirically validate Assumption 1 (Lipschitz continuity in embedding space). The results are shown in Figure A6. The x-axis represents the embedding distance $d(e_i, e_j) = 1 - \text{cosine similarity}(e_i, e_j)$ between document pairs i and j, where e_i and e_j are their respective text embeddings. The y-axis shows the absolute difference in marginal contributions $|v(S \cup \{i\}) - v(S) - (v(S \cup \{j\}) - v(S))|$ across all possible coalitions $S \subseteq S_q$ that do not contain documents i or j.

The scatter plot demonstrates that marginal contribution differences are linearly bounded by embedding distances, providing strong empirical support for the Lipschitz continuity assumption. The data reveals a clear pattern: document pairs with small embedding distances (high similarity) consistently exhibit small differences in their marginal contributions across coalitions. Conversely, document pairs with large embedding distances show a wider range of marginal contribution differences, from small to large values.

This asymmetric relationship aligns with our theoretical expectations and practical intuition. When two documents have similar embeddings, they naturally contribute similarly to any coalition, resulting in consistently small marginal contribution differences. However, when documents have dissimilar embeddings, their marginal contribution differences can vary significantly depending on the specific aspects they emphasize. For instance, two very different documents describing distinct aspects of a product (e.g., technical specifications versus user experience) may still provide comparable value to certain coalitions, leading to small marginal contribution differences despite large embedding distances.

The observed upper bound behavior-where embedding distance provides a ceiling for marginal contribution differences-validates the Lipschitz continuity assumption while capturing the nuanced relationship between semantic similarity and functional contribution in our evaluation framework. This empirical evidence supports the use of embedding-based distances as meaningful constraints in Shapley value computations.

B.3 Proof for Theorem 1

Proof. Without loss of generality, we assume the document i is in the cluster G_k. By the clustering construction, for any two documents i, j in the same cluster G_k, we have $d(e_i, e_j) \leq \epsilon$. Then by Assumption 1, for every coalition S not containing i or j, the marginal contributions of i and j differ by at most $L\epsilon$. That

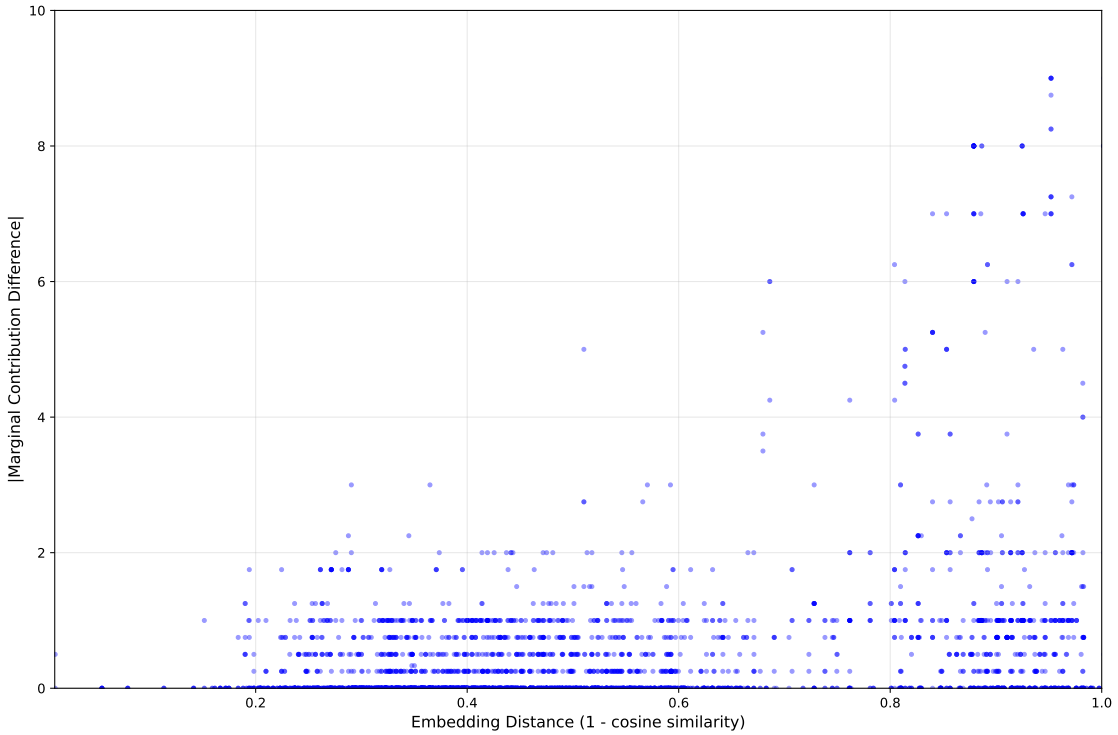


Figure A6: Empirical validation of Lipschitz continuity assumption. Each point represents a document pair (i, j) and a coalition S combination. The data encompasses all 48 queries in our dataset, and each query has 8 documents. Note that we sampled 5,000 data points for better visualization when the total number of points is $48 \times 2^{(8-2)} \times \binom{8}{2} = 86,016$.

is:

$$\left|[v(S \cup \{i\}) - v(S)] - [v(S \cup \{j\}) - v(S)]\right| \leq L\epsilon. \tag{A1}$$

Now consider the exact Shapley values ϕ_i and ϕ_j. By the definition of Shapley in Equation (4), $\phi_i - \phi_j$ can be expressed as the difference in i and j's marginal contributions to S, and then the weighted average over all coalitions S. Because each such difference is bounded by $L\epsilon$ as shown in Equation (A1), it follows that $|\phi_i - \phi_j| \leq L\epsilon$. In other words, all members of a cluster have Shapley values within a range of at most $L\epsilon$, i.e., for any i and j in G_k,

$$|\phi_i - \phi_j| \leq L\epsilon. \tag{A2}$$

Note that our Cluster Shapley implicitly forms a cluster-level cooperative game over the set of clusters $\{G_1, \ldots, G_m\}$ with value function $v_{\mathcal{G}}$:

$$v_{\mathcal{G}}(T) := v\left(\bigcup_{G_k \in T} G_k\right), \quad \text{for } T \subseteq \{G_1, \ldots, G_m\}.$$

We now claim that:

$$\hat{\phi}_{G_k} = \sum_{j \in G_k} \phi_j.$$

To see this, define a function $\psi(G_k) := \sum_{j \in G_k} \phi_j$. This defines an allocation of value to clusters based

on the sum of Shapley values in the original game. Since the original Shapley allocation is efficient,

$$\sum_{k=1}^{m} \psi(G_k) = \sum_{k=1}^{m} \sum_{j \in G_k} \phi_j = \sum_{j \in S_q} \phi_j = v(S_q) = v_{\mathcal{G}}(\{G_1, \ldots, G_m\}),$$

we see that ψ is an efficient allocation over the clusters. Moreover, the marginal contribution of G_k to any coalition of clusters T in $v_{\mathcal{G}}$ is given by:

$$v_{\mathcal{G}}(T \cup \{G_k\}) - v_{\mathcal{G}}(T) = v\left(\bigcup_{G_j \in T} G_j \cup G_k\right) - v\left(\bigcup_{G_j \in T} G_j\right).$$

This is precisely the aggregated marginal contribution of all documents in G_k to the documents in the union of T. Thus, the average marginal contributions of documents in G_k across permutations yield $\psi(G_k)$ as the total contribution of G_k, and by the uniqueness of the Shapley value under efficiency, symmetry, null document, and linearity (the other three properties symmetry, null document, and linearity hold simply by the definition of $v_{\mathcal{G}}$ and ψ), we know that $v_{\mathcal{G}}(G_k) = \psi(G_k)$. Then, it follows that:

$$\hat{\phi}_{G_k} = \psi(G_k) = \sum_{j \in G_k} \phi_j.$$

The Cluster Shapley algorithm assigns each $i \in G_k$ the same value $\hat{\phi}_i = \hat{\phi}_{G_k}/|G_k|$ as constructed in the last step. Let $\phi_{\min}$ and $\phi_{\max}$ be the minimum and maximum exact Shapley values among cluster G_k's members. Since:

$$\hat{\phi}_i = \frac{\hat{\phi}_{G_k}}{|G_k|} = \frac{1}{|G_k|} \sum_{j \in G_k} \phi_j,$$

this average lies between the minimum and maximum of the $\{\phi_j : j \in G_k\}$. In particular, $\phi_{\min} \leq \hat{\phi}_i \leq \phi_{\max}$ for all $i \in G_k$. Therefore the absolute error for any $i \in G_k$ satisfies

$$|\hat{\phi}_i - \phi_i| \leq \max\{\phi_{\max} - \phi_i,\ \phi_i - \phi_{\min}\} \leq \phi_{\max} - \phi_{\min}. \tag{A3}$$

But from the Equation (A2), we know $\phi_{\max} - \phi_{\min} \leq L\epsilon$. Hence $|\hat{\phi}_i - \phi_i| \leq L\epsilon$ as claimed in Equation (10). In the limit $\epsilon \to 0$, every document will eventually stand alone (since there is a finite minimum nonzero distance between any two distinct document embeddings in S_q). In that extreme case, the algorithm reproduces the exact Shapley values with zero error. Thus $\hat{\phi}_i \to \phi_i$ for all i. □

B.4 Proof for Corollary 1

Proof. Given $\phi_{\max} - \phi_{\min} = \delta$ and as shown in the proof of Theorem 1 (see Equation (A3)), we have $|\hat{\phi}_i - \phi_i| \leq \delta$ for each i. In the symmetric case ($\delta = 0$ by the definition of the symmetry property), then we have $|\hat{\phi}_i - \phi_i| = 0$. □

B.5 Proof of Theorem 2

Proof. For any document $i \in G_k$, the total error can be decomposed as:

$$|\tilde{\phi}_i - \phi_i| \leq |\tilde{\phi}_i - \hat{\phi}_i| + |\hat{\phi}_i - \phi_i|.$$

The second term $|\hat{\phi}_i - \phi_i|$ corresponds to the error due to clustering. By Theorem 1 (or by the assumption of Lipschitz continuity of v), we have:

$$|\hat{\phi}_i - \phi_i| \leq L\epsilon.$$

The first term $|\tilde{\phi}_i - \hat{\phi}_i|$ corresponds to the approximation error from algorithm $\mathcal{A}$. By assumption, $\mathcal{A}$ guarantees:

$$|\tilde{\phi}_{G_k} - \hat{\phi}_{G_k}| \leq \epsilon_{\mathcal{A}}(\eta) \quad \text{with probability at least } 1 - \eta.$$

Since $\tilde{\phi}_i = \frac{\tilde{\phi}_{G_k}}{|G_k|}$ and $\hat{\phi}_i = \frac{\hat{\phi}_{G_k}}{|G_k|}$, we have:

$$|\tilde{\phi}_i - \hat{\phi}_i| = \frac{|\tilde{\phi}_{G_k} - \hat{\phi}_{G_k}|}{|G_k|} \leq \frac{\epsilon_{\mathcal{A}}(\eta)}{|G_k|}.$$

Combining both parts completes the error bound:

$$|\tilde{\phi}_i - \phi_i| \leq L\epsilon + \frac{\epsilon_{\mathcal{A}}(\eta)}{|G_k|}.$$

Finally, regarding computational complexity:

- The clustering step requires $O(n^2)$ operations.
- The cluster-level Shapley approximation uses algorithm $\mathcal{A}$, with assumed cost $C_{\mathcal{A}}(m)$.

Therefore, the total computational complexity is:

$$O(n^2 + C_{\mathcal{A}}(m)).$$

□

B.6 Proof for Corollary 2

Proof. This result follows by applying Theorem 2 with the Monte Carlo approximation error derived via Hoeffding's inequality.

The total error decomposes as:

$$|\tilde{\phi}_i - \phi_i| \leq |\tilde{\phi}_i - \hat{\phi}_i| + |\hat{\phi}_i - \phi_i|,$$

where the term $|\hat{\phi}_i - \phi_i| \leq L\epsilon$.

Next, let X_t denote the marginal contribution of G_k in the t-th permutation. Then:

$$\tilde{\phi}_{G_k} = \frac{1}{N} \sum_{t=1}^{N} X_t.$$

By Hoeffding's inequality, for any $\delta > 0$:

$$\mathbb{P}\left(|\tilde{\phi}_{G_k} - \hat{\phi}_{G_k}| \geq \delta\right) \leq 2\exp\left(-\frac{2N\delta^2}{V_{\max}^2}\right).$$

To ensure deviation at most δ with probability $1-\eta$:

$$\delta = V_{\max}\sqrt{\frac{\log(2/\eta)}{2N}}.$$

Finally, noting $\tilde{\phi}_i = \tilde{\phi}_{G_k}/|G_k|$ gives:

$$|\tilde{\phi}_i - \hat{\phi}_i| \leq \frac{\delta}{|G_k|}.$$

Combining yields:

$$|\tilde{\phi}_i - \phi_i| \leq L\epsilon + \frac{V_{\max}}{|G_k|}\sqrt{\frac{\log(2/\eta)}{2N}}.$$

The computational complexity includes $O(n^2)$ for clustering and $O(Nm)$ for Monte Carlo sampling, totaling $O(n^2 + Nm)$. □

C Supplementary Implementation Details

C.1 Introduction to RAG

AI search engines are designed to provide real-time, contextually relevant responses to user queries. A key technique behind many of these systems is RAG, which integrates pre-trained LLMs with information retrieval to enhance response accuracy and relevance (Lewis et al., 2020). RAG addresses the limitations of static, pre-trained LLMs by incorporating new information from up-to-date, domain-relevant documents (which can be potentially proprietary to the firm). By grounding responses in reliable documents, RAG improves the relevance of AI-generated answers, reduces hallucinations, and mitigates the issue of outdated information that plagues static models (Gao et al., 2023).

RAG models require two pieces of machinery – (1) A generative model or LLM that was pre-trained on a large corpus of text, e.g., GPT, Llama, Claude, Gemini, Deepseek. These models can generate coherent general-purpose text, although they are often unable to incorporate proprietary documents and recent news, and (2) a set of documents, D, that can be used to provide additional information to the generative model. Depending on the use case, D can take many forms. For example, if the goal is to generate a search engine for news aggregation, then D would consist of a set of licensed news articles from news websites. Alternatively, if the goal is to generate a conversational search chatbot for aiding consumers in e-commerce websites, then D would consist of the platform's own proprietary database, including product details, consumer reviews, etc. The RAG architecture has three components:

- **Retriever (R):** When a search query comes in, the retriever locates and retrieves relevant information by identifying a set of documents that are relevant to the search query. Essentially, given a query q and a set of documents D, the retriever's goal is to identify a subset $S_q \subseteq D$ that is most relevant to the query.
- **Augmentation (A):** In this phase, the retrieved documents (S_q) are integrated with the original input (user query, q) to provide additional context for the generative model. This augmentation ensures that the response from the generative model is grounded in retrieved reliable information, enhancing both its accuracy and relevance.
- **Generator (G):** The generator, typically an LLM such as GPT or Claude, synthesizes the user's query (q) and the retrieved information (S_q) to produce a coherent response. By incorporating the retrieved documents, the LLM can generate outputs that go beyond its pre-trained knowledge, delivering more comprehensive and contextually appropriate responses.

C.2 Prompt Designs

We now provide our designed GPT-4o prompts for both summarization ($A(q, S)$) and evaluation ($v(q, A)$) steps, outlined in Figure A7 and Figure A8 respectively.

```
You are tasked with generating a high-quality summary based on user comments. Follow these steps
    to ensure that your summary is accurate, relevant, and well-structured.

1. Carefully Analyze the Comments:
   - Read through all the comments provided in the context.
   - Identify the key points that are related to the topic '{original_query}'.

2. Select Relevant Information:
   - Only include information in your summary that is relevant to the topic '{original_query}'.
   - For comments marked as "not relevant", simply state "[X] is not related to the query."
       Replace '[X]' with the corresponding comment number.

3. Construct a Coherent Summary:
   - Use an unbiased and journalistic tone in your summary.
   - Ensure that the summary is medium to long in length and that it covers the key points
       effectively.

4. Cite the Source of Information:
   - For each part of the summary, include a citation in the form '[NUMBER]', where 'NUMBER'
       corresponds to the comment's index.
   - Start numbering from '0' and continue sequentially, making sure not to skip any numbers.
   - The citation should be placed at the end of the sentence or clause that it supports.
   - If a sentence in your summary is derived from multiple comments, cite each relevant
       comment, e.g., '[0][1]'.

5. Final Review:
   - Double-check your citations to ensure they accurately correspond to the comments used.
   - Make sure that every sentence in the summary is cited and that irrelevant comments are
       correctly identified and excluded after the initial irrelevant statement.
   - Make sure every comment is cited. For example, if comment [0], [1], and [2] are all not
       related to the topic, then just summarize: '[0] is not related to the query. [1] is not
       related to the query. [2] is not related to the query.' If comment [0] is relevant, while
       [1], [2], and [3] are irrelevant, then summarize like this: provide a summary of [0], and
       then state '[1] is not related to the query. [2] is not related to the query. [3] is not
       related to the query.' Do not miss any comment even though they are irrelevant.
   - Ensure that your response is structured in JSON format with the following fields:
     - "key": A string that represents the indices of the comments used to generate this
         summary, e.g., "012" for comments 0, 1, and 2.
     - "summary": The final generated summary text, with citations included.

6. Key Reminders:
   - Do not include any irrelevant information in your summary. If a comment is not related to
       the topic, state it as described and move on.
   - Ensure that your summary is comprehensive, accurate, and clearly tied to the topic
       '{original_query}'.
```

Figure A7: Prompt for GPT-4o to analyze the relevant reviews to the original query and generate a summary. The prompt specifies citation rules and explicitly requires noting if reviews are irrelevant. The structured output is generated and formatted in JSON, consisting of two fields: "key" for indexing the source reviews and "summary" for the final generated text, complete with appropriate citations.

```
You are an AI model trained to evaluate summaries. Below, you will find several summaries
    identified by their labels. Your task is to rate each summary on one metric. Please make
    sure you read and understand every single word of these instructions.

Evaluation Criteria:
Information Coverage MUST be an integer from 0 to 10 - How well the summary captures and clearly
    describes one or several key characteristics of the product. A high-quality summary should
    convey the important features, benefits, or drawbacks of the product as highlighted in the
    reviews. It should provide a rich and accurate depiction of key points.

Pay attention: The most important consideration is how effectively the summary communicates the
    product's key characteristics. The clearer and more richly it conveys these characteristics,
    the higher the score. If it fails to adequately describe the product's features, it should
    receive a low score.

Evaluation Steps:
1. Read all summaries provided and compare them carefully. Ensure the summary clearly and richly
    describes the key points relevant to the product without including irrelevant information.
2. Identify any important details or characteristics of the product that are missing from the
    summary.
3. Rate each of the summary based on how well it covers and conveys the important information
    from the reviews. The MORE comprehensively the summary covers the relevant information, the
    HIGHER the score it should receive. Pay attention: The primary focus should be on the topic
    {original_query}. If the summary deviates from the topic, it should receive a low score,
    regardless of the amount of information it contains.
4. If a summary contains only the sentence "[X] is not related to the query." where X is a
    number, then give it a score of 1. However, if the summary contains other content besides
    this sentence, just ignore it when scoring.

Your response should be in JSON format, with an array of objects. Each object should have two
    properties:
1. "key": The key of the summary (e.g., "0", "1", "01", etc.)
2. "score": The score for that summary (an integer from 1 to 10)
```

Figure A8: Prompt for GPT-4o to evaluate the extent to which a summary accurately and comprehensively captures the key product attributes as requested in the query.

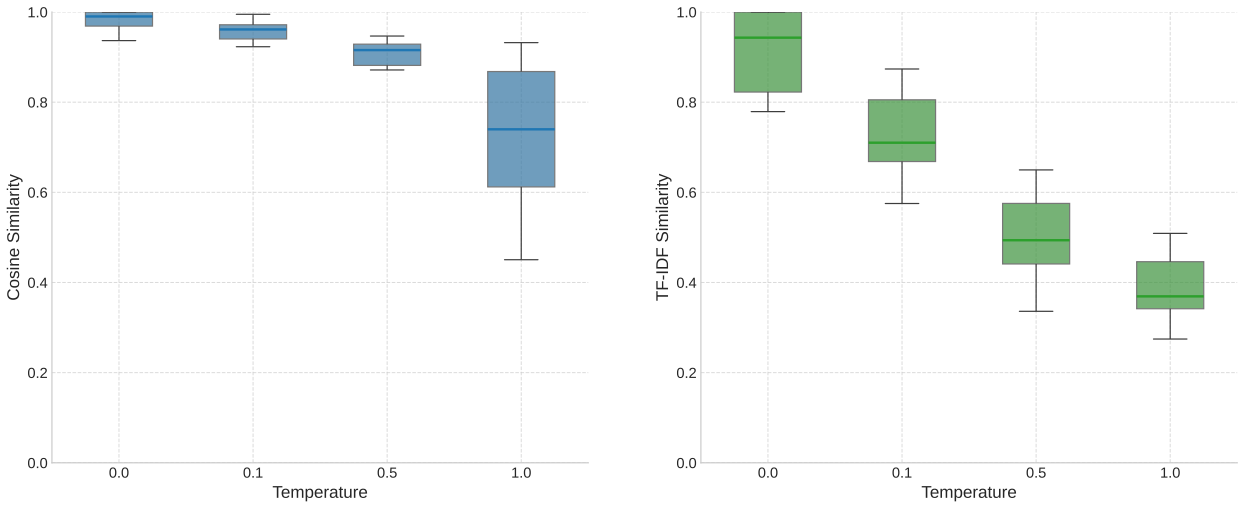
C.3 Temperature Selection in GPT-4o for Summarization and Evaluation Prompts

Due to the inherent stochastic nature of LLM outputs, we now investigate how the temperature setting in GPT-4o can affect the outcomes from summarization and evaluation prompts.

To gauge the temperature's impact, we conduct two numerical experiments examining the summarization and evaluation outcomes, respectively. For both experiments, we randomly sample five queries from the Amazon dataset and retrieve the five most relevant reviews per query. This filtering step helps mitigate overfitting, as our goal is to select an appropriate temperature setting and evaluate its effect using samples distinct from those used in the numerical experiment for testing algorithms. We test four temperature levels (0.0, 0.1, 0.5, and 1.0), and for each setting, we replicate the prompt 10 times to compute output variance.

Because the outputs of summarization prompts are text information, to gauge the performance, we use two metrics: (1) We assess semantic consistency through embedding-based cosine similarity, which measures the degree to which different summaries preserve the same underlying meaning regardless of specific word choices; (2) We examine lexical diversity using TF-IDF (Term Frequency–Inverse Document Frequency) similarity measures, which quantify the variation in vocabulary and phrasing across multiple generations. While semantic consistency measures reliability in meaning preservation, lexical diversity reflects the model's creativity in expression. Figure A9 visualizes the effect of temperature settings on both metrics.

As shown in Figure A9(a), even at temperature 0.0, where theoretically deterministic behavior is expected, the model exhibits slight variations in output (mean cosine similarity = 0.9820, std = 0.0217), confirming the



(a) Effect of Temperature on Summarization Semantic Consistency

(b) Effect of Temperature on Lexical Diversity

Figure A9: Analysis of Temperature's Effect on Summary Generation. (a) Cosine similarity of embeddings measures semantic consistency, where higher values indicate stronger preservation of meaning across generated summaries. (b) TF-IDF similarity reflects lexical choice patterns, where lower values indicate more diverse vocabulary usage in the generated text, demonstrating the trade-off between consistency and diversity at different temperatures.

presence of hardware-level computational variability. As temperature increases, we observe a decrease in semantic consistency, with mean similarities of 0.9576 (temp = 0.1), 0.8545 (temp = 0.5), and 0.7339 (temp = 1.0). While the decline in semantic consistency from temperature 0.0 to 0.1 is modest (approximately 2.5%), this minor trade-off shows significant improvements in lexical diversity, as demonstrated in the TF-IDF analysis in Figure A9(b). Specifically, when transitioning from temperature 0.0 to 0.1, we observe a beneficial decrease in TF-IDF similarity from 0.9102 to 0.7244, indicating substantially more diverse vocabulary usage while maintaining semantic integrity. This optimal balance point at temperature 0.1 enables richer and more nuanced expression through varied word choices, while preserving the essential meaning of the content with high semantic consistency. However, at higher temperatures, both metrics indicate potential instability in the generation process. The substantial increase in semantic standard deviation (from 0.0217 at temp = 0.0 to 0.1696 at temp = 1.0) suggests increasingly unpredictable semantic variations, while the further decrease in TF-IDF similarity (0.5013 at temp = 0.5 and 0.3845 at temp = 1.0) indicates excessive vocabulary variation. These patterns at higher temperatures could potentially compromise both semantic reliability and textual coherence, reinforcing our choice of temperature 0.1 as the optimal setting for balancing semantic preservation with expressive diversity.

Choosing the temperature for evaluation is similar and straightforward, as the output is a numerical score ranging from 0 to 10. Since our ultimate goal is to obtain stable Shapley values, which are computed as weighted averages of these scores, we focus directly on how temperature affects the variance of the Shapley values.

Using the same setup as in our summarization analysis, we fix the generated summaries and repeat the evaluation process under different temperature settings. This isolates the variance caused solely by the evaluation prompt. As shown in Figure A10, even at temperature 0.0, evaluation outputs exhibit variability due to hardware-level computational noise, similar to what we observed in summarization.

Among the tested settings, temperature 0.0 yields the lowest Shapley variance but still shows noticeable fluctuations. Temperature 0.1 displays slightly higher variance but achieves similar stability after averaging

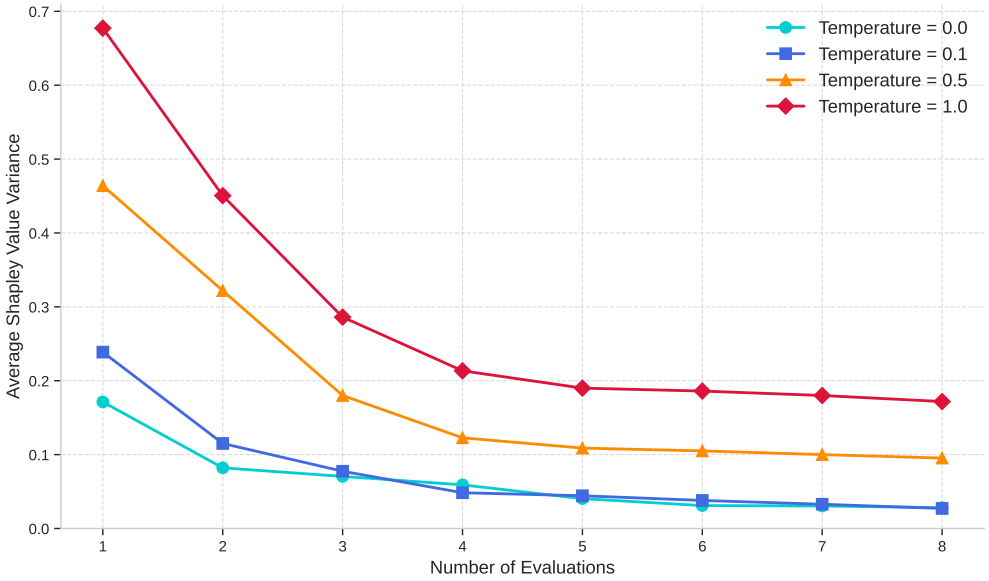


Figure A10: Shapley value variance under different temperatures and different replications of evaluation prompts. The x-axis represents the number of replications of evaluations. The y-axis is the variance of the average Shapley value over replications.

the evaluation score over multiple replications, making it a practical choice that balances consistency and alignment with our summarization process. In contrast, higher temperatures (0.5 and 1.0) result in substantially larger variances, suggesting diminished reliability.

We therefore adopt a temperature of 0.1 for evaluation prompts to ensure stable Shapley values without introducing excessive rigidity or randomness.

C.4 Variance Analysis of Summarization, Evaluation, and Shapley Value

Due to the stochastic nature of LLM prompts, we now analyze how this randomness can finally affect the variance of Shapley values, and propose a way to reduce the variance in this appendix.

The variance comes from both the summarization prompt $A(S)$ and the evaluation prompt $v(A(S))$. We omit the notation q in functions for better clarity, as q is fixed and its omission does not introduce ambiguity.

Formally, we can write down the random summarization and evaluation process as:

$$v(A(S)) = \mu(A(S)) + \varepsilon, \tag{A4}$$

where ε is white noise in the evaluation process, and $\mu(A(S))$ represents the expected performance score of the summarization $A(S)$. Note that $A(S)$ is a random event rather than a random variable, as the LLM may generate different summaries even under the same set of reviews S due to intrinsic randomness in GPT responses. This formula reflects that the randomness in the observed evaluation score $v(A(S))$ originates from two sources: the randomness in A and the evaluation noise.

We assume that the random summarizations $\{A(S)\}_{S\subseteq D}$ are mutually independent. Given this assumption and independent white noise, and based on the Shapley formula -Equation (4)-which is essentially a weighted average of v, the variance in Shapley can be expressed as the weighted average of the variances of v. Thus, in the following, we focus on analyzing the variance of $v(A(S))$:

$$\mathrm{Var}(v(A(S))) = \mathbb{E}[v(A(S))^2] - (\mathbb{E}[v(A(S))])^2 .$$

Substituting Equation (A4) into the above variance expression yields:

$$\text{Var}(v(A(S))) = \mathbb{E}_{A,\varepsilon}\left[(\mu(A(S)) + \varepsilon)^2\right] - (\mathbb{E}_{A,\varepsilon}[\mu(A(S)) + \varepsilon])^2 .$$

Expanding the squared terms and leveraging the linearity of expectation, we can get:

$$\text{Var}(v(A(S))) = \mathbb{E}_A[\mu(A(S))^2] + 2\mathbb{E}_{A,\varepsilon}[\mu(A(S))\varepsilon] + \mathbb{E}_\varepsilon[\varepsilon^2] - (\mathbb{E}_A[\mu(A(S))] + \mathbb{E}_\varepsilon[\varepsilon])^2 .$$

Given that ε is independent of $\mu(A(S))$ and has zero mean, we have $\mathbb{E}[\mu(A(S))\varepsilon] = \mathbb{E}[\mu(A(S))]\mathbb{E}[\varepsilon] = 0$ and $\mathbb{E}[\varepsilon] = 0$. Therefore, the expression simplifies to:

$$\text{Var}(v(A(S))) = \mathbb{E}[\mu(A(S))^2] + \mathbb{E}[\varepsilon^2] - (\mathbb{E}[\mu(A(S))])^2 .$$

Recognizing that $\text{Var}(\mu(A(S))) = \mathbb{E}[\mu(A(S))^2] - (\mathbb{E}[\mu(A(S))])^2$ and $\text{Var}(\varepsilon) = \mathbb{E}[\varepsilon^2]$, we can rewrite the variance of $v(A(S))$ as:

$$\text{Var}(v(A(S))) = \text{Var}(\mu(A(S))) + \text{Var}(\varepsilon). \tag{A5}$$

This result demonstrates that the total variance of the evaluation score $v(A(S))$ is the sum of the variance due to the summarization process $\text{Var}(\mu(A(S)))$ and the variance due to the evaluation noise $\text{Var}(\varepsilon)$.

C.4.1 Empirical Variance in Summarization and Evaluation

To validate this variance decomposition and gauge the magnitudes of these variances, we conduct an experiment to quantify the variance contributions from both the summarization and evaluation stages. We select five distinct queries from Amazon product reviews, each containing five reviews, which is the same setting as in Web Appendix §C.3. For each query, we calculate the Shapley value to explore how the summarization and evaluation processes contribute to the overall variance. Specifically, for each subset of reviews in a query, we replicate both summarization and evaluation prompts three times to calculate the following empirical variance.

- **Total Variance** $\text{Var}(v(A(S)))$ - This variance represents the overall variability of the evaluation scores for a given subset, considering all summarization and evaluation rounds. For each subset of reviews, we generate multiple summarizations and perform several evaluation rounds for each summarization. The total variance is calculated as the variance of all the evaluation scores across these rounds, capturing the combined effects of both summarization and evaluation.

- **Summarization Variance** $\text{Var}(\mu(A(S)))$ - This variance reflects the variability introduced during the summarization process. After generating multiple summaries for each subset, we compute the mean evaluation score for each summary. The summarization process variance is then calculated as the variance of these mean evaluation scores across different summarizations. This captures how much the content of the summaries themselves contributes to the overall variability in evaluation scores, independent of the evaluation noise.

- **Evaluation Variance** $\text{Var}(\varepsilon)$ - This variance isolates the variability introduced during the evaluation process. For each summarization, we evaluate the subset multiple times. The evaluation noise variance is computed as the average variance of the scores within each summarization round, reflecting the inconsistency of GPT-based evaluations across the same summary. In other words, it measures how

much the scores fluctuate due to noise in the evaluation model rather than changes in the summaries themselves.

The results of this experiment, presented in Table A1, compare the total variance, evaluation noise variance, and summarization process variance for each subset. Consistent with the variance decomposition analysis, the total variance equals the sum of the variances from both the evaluation noise and the summarization process. On average, the summarization process variance accounts for approximately 53.08% of the total variance, while evaluation noise contributes around 46.92%.

Subset S	**Total Variance**	**Evaluation Variance**	**Summarization Variance**
{1}	0.3729	0.0741	0.2988
{2}	0.2654	0.0741	0.1914
{3}	0.1173	0.0741	0.0432
{4}	0.1358	0.0371	0.0988
{5}	0.3642	0.1667	0.1975
{1, 2}	0.3519	0.0926	0.2593
{1, 3}	0.3933	0.1111	0.2822
{1, 4}	0.2037	0.0371	0.1667
{1, 5}	0.1975	0.1482	0.0494
{2, 3}	0.1543	0.0741	0.0802
{2, 4}	0.1605	0.0741	0.0864
{2, 5}	0.2099	0.1296	0.0802
{3, 4}	0.1975	0.0926	0.1049
{3, 5}	0.2778	0.1482	0.1296
{4, 5}	0.2778	0.1482	0.1296
{1, 2, 3}	0.2469	0.1111	0.1358
{1, 2, 4}	0.2963	0.1111	0.1852
{1, 2, 5}	0.2099	0.1482	0.0617
{1, 3, 4}	0.1790	0.1111	0.0679
{1, 3, 5}	0.2654	0.1667	0.0988
{1, 4, 5}	0.3599	0.1852	0.1747
{2, 3, 4}	0.1481	0.0926	0.0556
{2, 3, 5}	0.2160	0.1667	0.0494
{2, 4, 5}	0.2346	0.2037	0.0309
{3, 4, 5}	0.2778	0.1914	0.0864
{1, 2, 3, 4}	0.2407	0.0556	0.1852
{1, 2, 3, 5}	0.2346	0.0741	0.1605
{1, 2, 4, 5}	0.2716	0.1482	0.1235
{1, 3, 4, 5}	0.2407	0.1296	0.1111
{2, 3, 4, 5}	0.3086	0.0741	0.2346
{1, 2, 3, 4, 5}	0.0432	0.0185	0.0247
Average Variance	0.2457	0.1153	0.1304

Table A1: Performance score variance for different subsets. There are a total of $2^5 - 1 = 31$ different subsets.

C.4.2 Reduce the Variance of Shapley Value

The variance decomposition shown in Equation (A5) provides a foundation for understanding how variance arises in our system, guiding our approach to measuring and managing variance when calculating the Shapley value. Specifically, we generate multiple instances of the summarization $A(S)$ and take the average score across these summaries to reduce variance introduced by A, and/or evaluate each summarization multiple times to obtain an averaged score across evaluations, thereby reducing variance introduced by ε.

However, more summarization and evaluation replications mean higher computation cost, although with lower variance in Shapley. To determine the most cost-effective approach, we test variance under various configurations, including multiple evaluations and summarization. Specifically, we conduct the experiment using a single query, "the delivery speed of the card," with four selected reviews. For each combination of

summarization counts (ranging from 1 to 4) and evaluation counts (also from 1 to 4), we repeat the process six times to calculate the variance in Shapley value. We measure computational cost in terms of the total response time of the GPT-4o API required for each configuration.

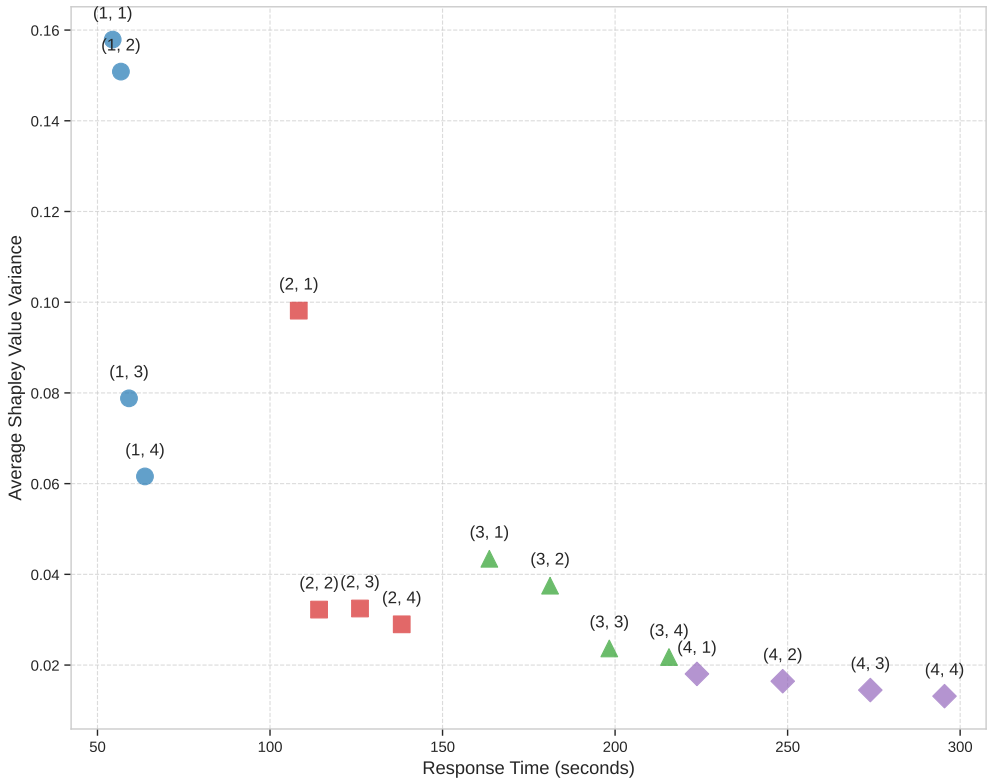


Figure A11: Cost-Effectiveness Analysis. The x-axis is the API response time (in seconds), and the y-axis is the average Shapley value variance. Each labeled point represents a configuration, with the first number indicating the number of summarizations and the second the number of evaluations.

The results are displayed in Figure A11. We can observe that increasing both the summarization and evaluation counts reduces the average variance of the Shapley values. Notably, the most significant decrease in variance occurs when moving from lower to moderate counts of summarizations and evaluations. For example, increasing the evaluation count from 1 to 3 while keeping the summarization count at 1 reduces the average variance from approximately 0.1579 to 0.0788. Similarly, increasing the summarization count from 1 to 2 with an evaluation count of 2 decreases the average variance from approximately 0.1508 to 0.0322.

However, the rate of variance reduction diminishes with higher counts. Beyond certain thresholds, additional reductions in variance become marginal. For instance, increasing the evaluation count from 3 to 4 with a summarization count of 1 results in a variance reduction of only about 0.0172, from 0.0788 to 0.0616. These findings indicate a trend of diminishing returns, suggesting that conducting more than three evaluations or more than two summarizations provides limited benefits in terms of variance reduction.

Also, notice that one more summarization incurs more computation cost than one more evaluation because summarization outputs have more tokens than only one integer score out of the evaluation prompt. Considering practical applications where computational resources and time are constrained, a configuration with 1 summarization and 3 or 4 evaluations achieves substantial variance reduction while maintaining reasonable computation times.

To further examine whether evaluations should be replicated three or four times, we conduct an experiment using the test dataset consisting of five queries, each associated with five reviews-the same setup as in Web Appendices C.3 and C.4.1. We fix the summarization to a single run and vary the number of evaluation repetitions from 1 to 8. For each configuration, we compute Shapley values across 10 independent replications to estimate the variance.

As shown in Figure A12, the variance in Shapley values decreases as the number of evaluation repetitions increases, but the rate of reduction diminishes over time. The most notable improvement occurs between one and four evaluations, where the average variance drops from approximately 0.25 to 0.05. Beyond four repetitions, additional variance reduction becomes marginal. These findings suggest that averaging over four evaluations provides an effective trade-off between computational cost and variance reduction. Accordingly, we adopt this configuration in our main numerical experiments.

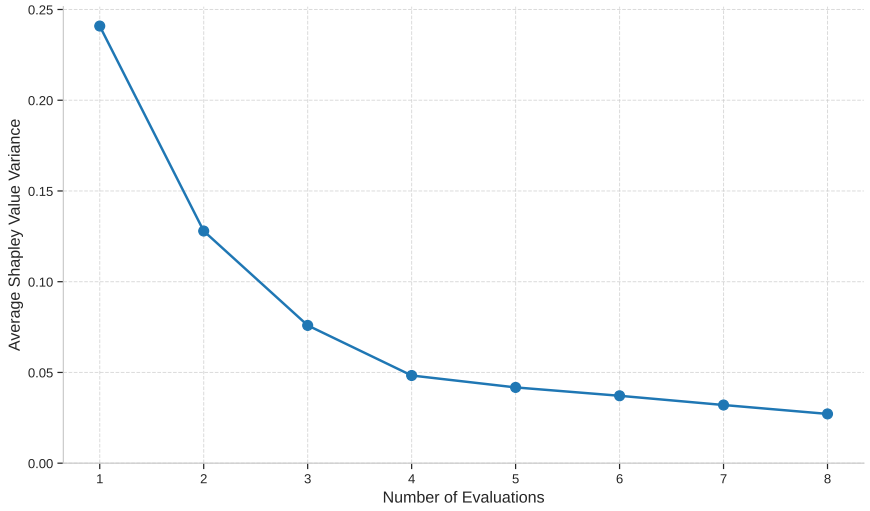


Figure A12: Reduction in Shapley Value Variance with Increased Evaluations

C.5 Computation time of the Clustering Step

We now analyze our adaptive clustering (i.e., Algorithm 3) using the same experimental setup as our benchmark comparisons: 48 test queries, each evaluated under 41 different ϵ settings ranging from 0 to 1. The results demonstrate efficient convergence behavior, with 71.19% of cases converging immediately without requiring iterations. When iterations are needed, the process requires an average of 1.8 iterations, with a maximum of 19 iterations observed in extreme scenarios.

To precisely measure the computational cost of individual iterations, we conducted a separate timing experiment using identical code implementation on an Intel i7-13900K processor. We performed 1,000,000 iterations and completed them within 1.53×10^{-3} seconds, yielding an average time of 1.53×10^{-9} second per iteration. While we acknowledge that measurements at such small time scales can be subject to system-level variations due to factors such as CPU scheduling, memory access patterns, and hardware-specific optimizations, these results consistently demonstrate that the iteration overhead is orders of magnitude smaller than the LLM operations in our proposed Cluster Shapley Algorithm Step 2 in §4.3, which require approximately 3.5 seconds per subset evaluation. This substantial difference in time scales confirms that the computational cost of our adaptive clustering approach remains negligible in the overall Shapley value calculation pipeline.

D Robustness Checks and Extensions

We now present detailed results and discussion for several aspects of the robustness checks and extensions described in §7.3.

D.1 Robustness Check using Claude for Evaluation

In the main study, we use the same GPT-4o for both summarization and for evaluating the summaries, which may introduce biases because LLMs tend to favor their own summaries. To address this, we conduct a robustness check here using `Claude-3.5-Sonnet` as an alternative evaluation model to compare against the results obtained from `GPT-4o-2024-08-06`, aiming to examine whether the evaluation outcomes remain consistent using different LLMs.

To ensure a fair comparison, we use the same prompt (as shown in Figure A8) and the same temperature setting of 0.1 for evaluation in Claude-3.5. For each summary generated by GPT-4o, we perform four independent evaluations using both GPT-4o and Claude-3.5, and take the average over four replications as the final evaluation score, to mitigate the variance caused by the inherent stochasticity of LLM outputs. To quantify the alignment between the two LLM models' evaluations, we conduct the Pearson correlation test on the relative rankings of summaries assigned by the two models, as this ranking metric captures whether the models agree on the comparative quality of summaries even if their absolute scores differ. We also do the correlation test on Shapley values to see the impact of using different LLM models for evaluation on the final Shapley values.

The correlation between evaluation rankings obtained from two different LLM evaluation models is 0.788, and the correlation between the resulting Shapley values is 0.915. Both results are statistically significant at the 0.05 level, suggesting that our evaluation framework yields consistent outcomes across different evaluation approaches, and the validity of using the same GPT-4o for both summarization and evaluation.

D.2 Performance under MAPE and MSE Metrics

For completeness, we evaluate the performance of the Cluster Shapley algorithm using alternative error metrics-MAPE and MSE-and compare it to other benchmark methods in Figure A13. For ease of comparison, we replicate the results from Figure 5 as panel (c) in Figure A13. As shown, Cluster Shapley consistently outperforms the benchmarks across all three metrics.

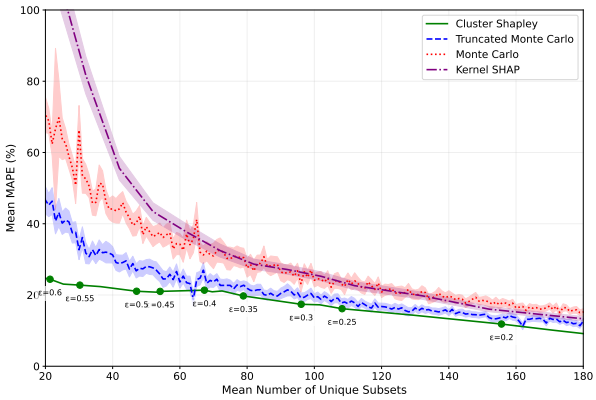
D.3 Cluster Shapley using the Standard DBSCAN

We now report the performance of the Cluster Shapley algorithm using the standard DBSCAN algorithm, instead of our proposed adaptive variant, in Figure A14. As discussed in §4.3, the standard DBSCAN does not guarantee that all pairs of documents within the same cluster have embedding distances strictly smaller than ϵ, which may affect approximation quality.

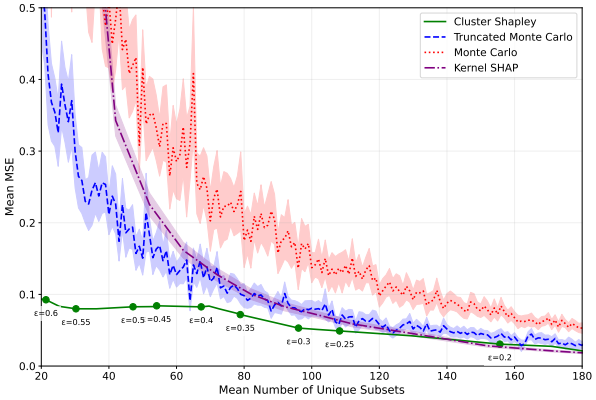
Comparing Figure A14 (which uses standard DBSCAN for clustering) with Figure A13 (which uses our proposed adaptive DBSCAN, i.e., Algorithm 3), we observe that Cluster Shapley achieves consistently better performance with adaptive clustering across all three approximation error measures. Moreover, when using more than 60 unique subsets, Cluster Shapley with standard DBSCAN performs significantly worse than both Truncated Monte Carlo and Kernel Shapley, highlighting the importance of enforcing tighter clustering constraints through the adaptive procedure.

D.4 Sample Splitting Robustness Check

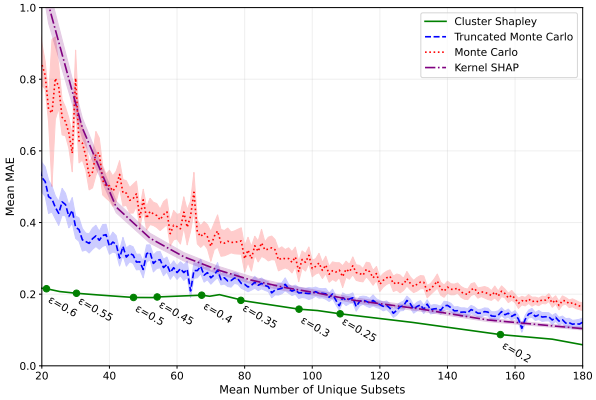
To assess the robustness of our results to the choice of test queries and the stability of hyperparameter tuning, we conduct a sample-splitting analysis. Specifically, we randomly divide the 48 test queries used in the



(a) MAPE Performance



(b) MSE Performance



(c) MAE Performance

Figure A13: Efficient frontiers of algorithms under MAPE, MSE, and MAE measures. Subfigure (c) replicates Figure 5. The x-axis represents the number of unique subsets used by the algorithms, averaged across all test queries and reviews. The y-axis represents the Mean error measures of the Shapley values, averaged across all test queries and reviews.

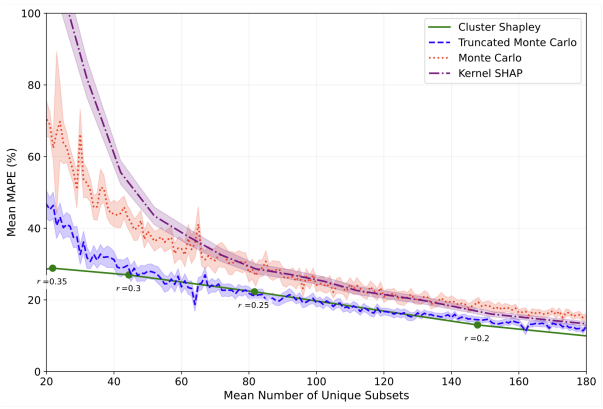
main analysis into two equal subsets, referred to as Split 1 and Split 2, each containing 24 queries. We then replicate the algorithm comparison analysis (shown in Figure 5 of the main text) separately for each split.

The results, presented in Figure A15, yield two key findings. First, the comparative performance of different algorithms remains consistent across the two query subsets, indicating that the results are not driven by specific queries. Second, the choice of the clustering diameter ϵ is stable across splits, suggesting that one subset can be reliably used for hyperparameter tuning without compromising performance on the other subset.

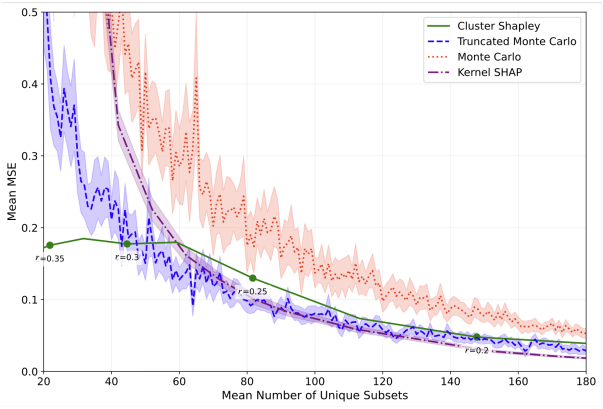
D.5 Cluster Shapley with Cluster-Level Approximation Algorithms

While the proposed Cluster Shapley algorithm substantially reduces computational complexity by limiting the number of unique subsets requiring evaluation, certain real-world applications may involve substantially larger document sets. To further enhance efficiency in such settings, we explore the integration of approximation algorithms within Cluster Shapley to approximate cluster-level Shapley values. Specifically, in this appendix, we evaluate Algorithm 2 by incorporating the Monte-Carlo approximation algorithm in Step 2.

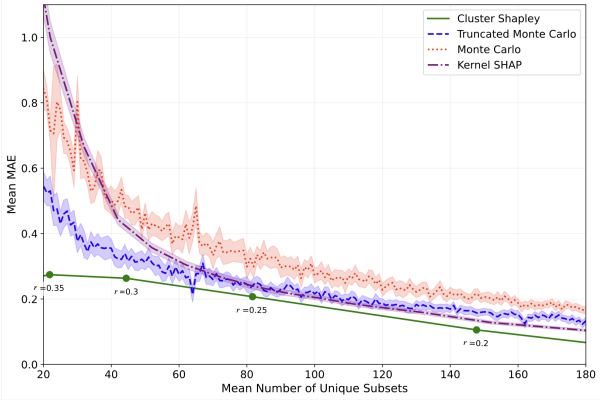
We present two experiments. The first experiment (10 documents) replicates the main benchmark setting



(a) MAPE Performance

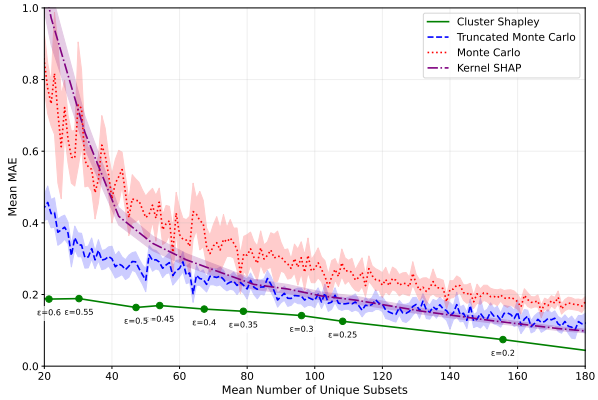


(b) MSE Performance

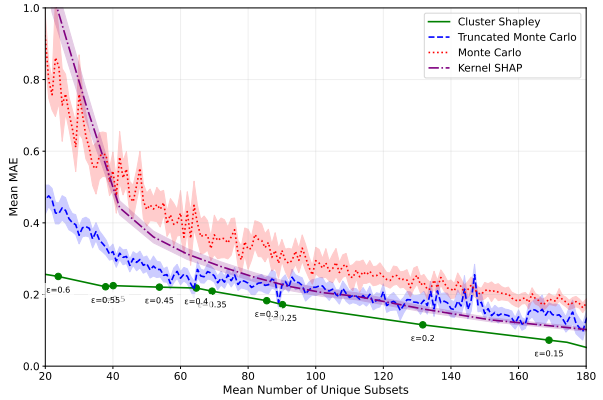


(c) MAE Performance

Figure A14: Efficient frontiers of Cluster Shapley using the **standard DBSCAN** in Step 2. Note that the performance curves for the other three algorithms are identical to those in Figure A13. Each point on the Cluster Shapley curve corresponds to a different neighborhood radius r, which is the same as the clustering diameter ϵ in the standard DBSCAN.



(a) Split 1 (24 queries)



(b) Split 2 (24 queries)

Figure A15: Robustness check based on random query splitting. Algorithm comparison results for Split 1 and Split 2 remain consistent, confirming that the main findings in Figure 5 are not sensitive to the choice of test queries.

but increases the number of documents per query to reflect more computationally intensive scenarios. The second experiment scales up to 30 documents per query to assess the algorithm's behavior under much larger document sets, where exact Shapley computation is infeasible.

Experiment 1: 10 Documents with LLM-Based Evaluation. We randomly sample two queries from our benchmark dataset ("How quick is the delivery of the gift card?" and "How's the quality of the PlayStation?"), each associated with 10 reviews (instead of 8 in the main analysis). This increase results in a total of $2^{10} - 1 = 1{,}023$ coalitions per query, leading to nearly 300,000 tokens processed by LLMs.

For the standard Cluster Shapley algorithm, we use the implementation described in the main paper. For the Monte Carlo–based variant, we vary two hyperparameters: the clustering diameter $\epsilon \in [0.01, 1.0]$ and the number of Monte Carlo permutations used in Step 2. We evaluate all combinations of these hyperparameters and report the efficient frontier-i.e., the lowest approximation error achieved for each level of computational cost (measured by the number of unique subset evaluations).

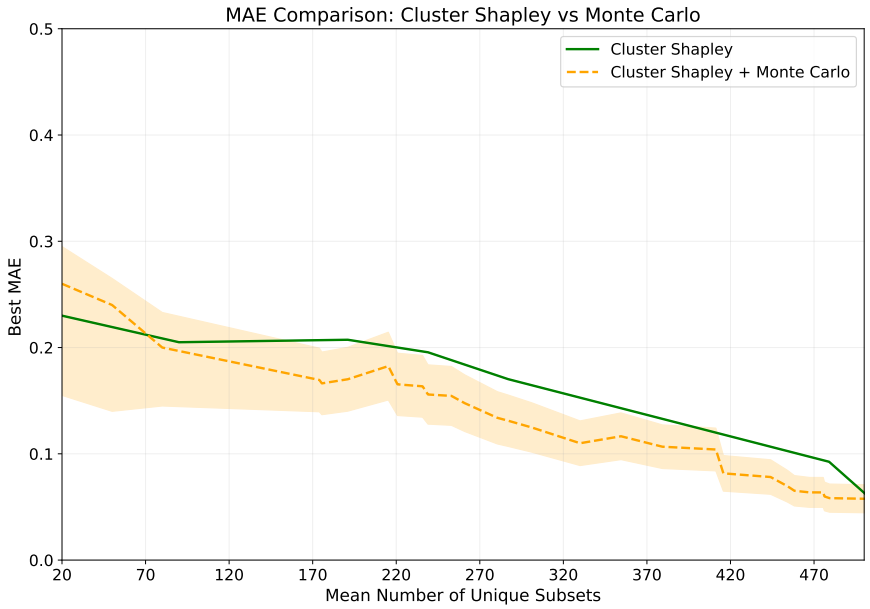


Figure A16: Efficient frontiers of Cluster Shapley and Cluster Shapley with Monte Carlo sampling (10 documents). The x-axis shows the number of unique subsets evaluated (averaged over 10 replications), and the y-axis reports the Mean Absolute Error (MAE) relative to exact Shapley values computed with LLM evaluation. 95% confidence intervals are shown.

Figure A16 shows that Cluster Shapley with Monte Carlo consistently achieves lower approximation error than the standard version for the same computational cost (though the difference is not statistically significant below 170 subset evaluations), suggesting that introducing Monte Carlo sampling in Step 2 improves the efficiency–accuracy trade-off.

Experiment 2: 30 Documents with Synthetic Evaluation. To evaluate Cluster Shapley in larger-scale settings, we extend the experiment to queries with 30 documents. Exact Shapley value computation becomes intractable at this scale ($2^{30} \approx 10^9$ subsets), so we generate synthetic summarization scores for evaluation.

Specifically, we assign each document a true Shapley value linearly proportional to its cosine similarity with the query embedding, normalized to sum to 10. The value function $v(q, A(q, S))$ for any subset S is then computed as the sum of the true Shapley values of documents in S, plus additive Gaussian noise. We simulate four levels of noise with standard deviation $\sigma \in \{0.05, 0.10, 0.15, 0.20\}$, all with mean zero. This

setup allows us to evaluate the robustness and scalability of Cluster Shapley with Monte Carlo sampling under noisy yet controlled evaluation.

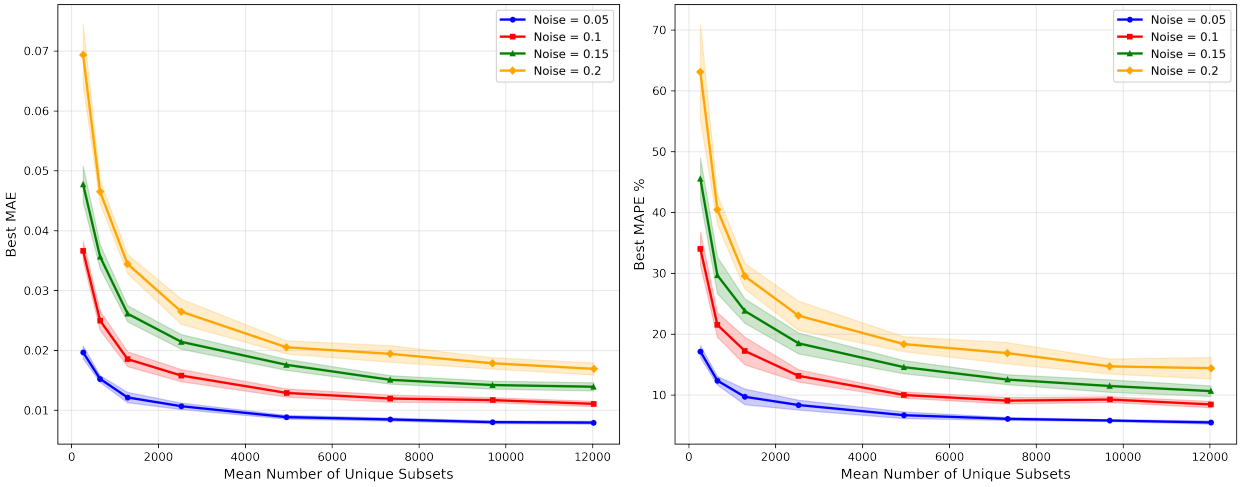


Figure A17: MAE and MAPE of Cluster Shapley with Monte Carlo sampling under four noise levels (30 documents). The x-axis shows the number of unique subsets evaluated. The y-axes report the Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) relative to the true synthetic Shapley values. Each line corresponds to a different level of Gaussian noise added to the evaluation.

As shown in Figure A17, Cluster Shapley with Monte Carlo remains effective even in large-scale settings with noisy evaluation. As expected, higher noise levels slightly degrade performance, but the method still achieves low MAE and MAPE with a reasonable number of subset evaluations.

References

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Reddit. Reddit answers (currently in beta), 2025. Link. Accessed on May, 2025.