Document Valuation in LLM Summaries: A Cluster Shapley Approach

Zikun Ye *
University of Washington

Hema Yoganarasimhan
University of Washington

Abstract

Large Language Models (LLMs) combined with the Retrieval-Augmented Generation (RAG) framework are transforming consumer search and information consumption. Consumers increasingly rely on LLM-based summaries (generated from multiple source documents in response to search queries) instead of reading individual documents/sources. While this simplifies the search and information retrieval process, there exists no established method to quantify the contribution of each document to the overall quality of the LLM-summary, and compensate document creators. In this paper, we first propose a theoretical solution to this problem based on the Shapley value framework that is fair and transparent. While this framework is theoretically appealing, it is practically infeasible since the computational costs of calculating Shapley values grow exponentially with the number of documents. We therefore develop a novel *Cluster Shapley* algorithm, that leverages embedding-based semantic similarity to cluster similar documents, thereby reducing the number of LLM summarizations and evaluations while maintaining high attribution accuracy. Our approach is particularly well-suited to LLM settings since LLMs themselves can be used to obtain high-quality text embeddings. We apply our approach to a large dataset consisting of Amazon reviews, where LLMs are used to summarize review content. Using a series of numerical experiments, we show that our algorithm achieves up to a 40% reduction in computation time compared to exact Shapley values, and outperforms other baseline approximations like Monte Carlo sampling and Kernel SHAP in both efficiency and accuracy. To our knowledge, this work is the first to propose an economic and computational framework for document value attribution in LLM-summarization systems. Our approach is agnostic to the exact LLM used, the summarization process used, and the evaluation procedure, which makes it broadly applicable to a variety of summarization settings.

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

*We thank Yizhuo Chang and Lei Wang for outstanding research assistance. Please address all correspondence to: zikunye@uw.edu and hemay@uw.edu.

1 Introduction

In recent years, digital transformation has fundamentally reshaped how people consume information, from traditional print media to dynamic online platforms. Today, this evolution has entered a new phase with the advent of Large Language Models (LLMs), which are revolutionizing information consumption through their ability to generate contextual, summarized responses (Sharma et al., 2024). Major technology platforms have integrated LLMs into their search infrastructure, such as OpenAI's ChatGPT Search (OpenAI, 2024), Microsoft's Bing AI (Mehdi, 2023), and Google's Grounding with Google Search (Reid, 2023), substantially improving information accessibility and processing efficiency (Xiong et al., 2024).

This LLM-driven paradigm shift represents a structural change in information markets and search behavior. First, compared to traditional search engines like Google that return a list of links, LLM-powered search provides an interactive dialogue with synthesized answers, freeing 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, systems that augment LLMs with real-time retrieval and source grounding can overcome key limitations such as hallucinations and outdated knowledge, ensuring that answers are both current and accurate (Perplexity, 2024). The market has rapidly embraced this evolution, as demonstrated by Perplexity AI's growth to 15 million monthly active users and projected annual revenue of $35 million—representing a 350% year-over-year increase (DemandSage, 2024a).

The impact of this transformation extends beyond web search. In e-commerce, for example, Amazon has introduced an AI-powered tools to enhance product search and discovery. One such feature is an AI-generated review summary on product pages review, see Figure 1 for an illustration with a wireless controller product, which summarizes customer reviews to reduce information overload for consumers. Amazon has also developed Rufus, a conversational shopping assistant that provides real-time, dialog-based product recommendations (Schermerhorn, 2023; Mehta and Chilimbi, 2024). These applications leverage Retrieval-Augmented Generation (RAG) architecture, a framework that integrates external information retrieval with LLM-based generation to ensure that the AI's responses are grounded in up-to-date, relevant content while minimizing user search effort.

However, alongside these innovations come significant challenges for LLM-integrated search systems. First, the operational costs of running LLM-based services at scale are extraordinarily high. Processing a single user query via a state-of-the-art LLM (e.g., ChatGPT) can cost over 1,000 times more than a traditional Google search (Will, 2023). This high cost creates an urgent need for sustainable revenue models, yet existing subscription-based approaches have proven insufficient to cover these operational expenses (Smith, 2023). Traditional search engines rely on advertising for monetization — content publishers are compensated for user traffic through metrics like impressions (CPM) and clicks (CPC). Yet this model breaks down in an LLM-based search paradigm where users receive answers directly on the platform without clicking through to the original content. In response, search providers are beginning to experiment with alternative monetization strategies: Perplexity AI has introduced brand-sponsored queries (Criddle, 2024), and Google is exploring auction systems allowing the winning content to show up in LLM responses (Dubey et al., 2024; Hajiaghayi

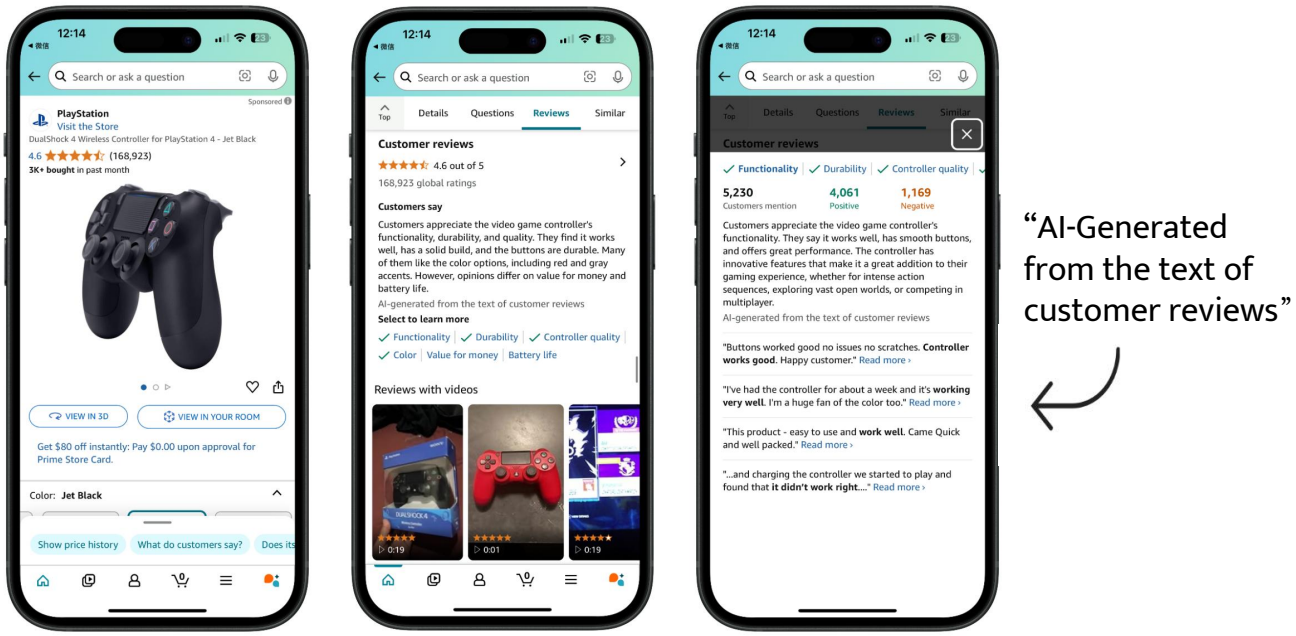


Figure 1: Amazon's AI-generated customer review for a wireless controller product: The left image shows the wireless controller product page on Amazon. The center image displays Amazon AI-generated comprehensive review of this product. Users can click "Select to learn more" to focus on specific aspects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

et al., 2024).

Second, the effectiveness of RAG-based systems hinges on access to high-quality information sources, but content providers are increasingly reluctant to provide documents to this new ecosystem. In the traditional search model, publishers at least gain traffic (and advertising revenue) when users click their links. In contrast, LLM-powered search delivers a synthesized answer directly in the search interface, allowing users to obtain information without ever visiting the source websites. This shift threatens the revenue streams of content creators and raises concerns about uncompensated use of their data. Indeed, several major publishers 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). Even when content providers do allow their data to be used by LLM platforms, attributing and rewarding each provider's contribution to a synthesized answer is non-trivial. The value that a particular document adds to the final summary depends not just on its presence, but on qualitative factors like the relevance, uniqueness, and reliability of the information it contains. This multi-faceted value contribution cannot be captured by simple metrics (e.g., word count or presence/absence in the answer), and traditional frameworks (such as copyright or link-based attribution) are ill-suited to ensure fair compensation in this context.

Addressing the challenge of **equitable document valuation** for document/content contributors in LLM-generated answers is thus both an urgent and underexplored problem. In this paper, we propose a framework for fair and transparent source attribution by leveraging **Shapley values** from cooperative game theory (Shapley, 1953). Shapley values provide a principled way to distribute total value among multiple documents by considering each document's marginal impact on the outcome, such as the quality or usefulness of an LLM-generated summary. Importantly, Shapley values ensure fair attribution through four key properties: Efficiency (ensuring the total value is fully distributed), Symmetry (treating equally contributing documents identically), Null Document (assigning zero value to non-contributing documents), and Linearity (ensuring additive consistency across multiple evaluations). These properties uniquely determine the Shapley value, making it an ideal framework for our application, see detailed discussion in §4.1. By computing Shapley values, we can quantify exactly how much each document contributes to the overall answer while accounting for complex interactions such as redundancy and overlapping information. To the best of our knowledge, this work represents the first application of Shapley value attribution to explain document contributions in an LLM-based summarization system.

However, a major hurdle in applying Shapley values to LLM outputs is the computational complexity. Calculating exact Shapley values requires generating and evaluating the LLM's summary output for every possible subset of documents, which is computationally infeasible when the number of documents is large. To reduce the computation cost without introducing large errors of Shapley value, many approximation algorithms are 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 those algorithms treat each document independently and do not leverage the text information of documents.

To address this computation challenge, we propose an approximation algorithm, Cluster Shapley, which effectively leverages textual information of documents to further reduce the computation. The idea behind Cluster Shapley is intuitive. If two documents provide very similar information, including or excluding one versus the other has nearly the same effect on the summary. Cluster Shapley exploits this by grouping together similar documents (using embedding-based clustering) and treating each cluster as a single combined document for the purposes of the Shapley calculation. This significantly reduces the number of document combinations the LLM must summarize and evaluate. Another advantage of our algorithm is that it provides flexibility in balancing speed and precision through the adjustable clustering ϵ hyperparameter. For tasks requiring high precision, a smaller ϵ can be selected, while tasks prioritizing speed can utilize a larger ϵ.

We evaluate our approach on the Amazon product review dataset, a large-scale corpus spanning May 1996 to September 2023, containing over 571 million reviews across 33 product categories (Hou et al., 2024). Beyond its extensive coverage of consumer opinions, this dataset closely aligns with real-world applications, as Amazon's AI-generated review summaries are built on the same repository. Since real user queries are not publicly available, we construct a diverse set of 48 queries across different product categories to ensure a representative evaluation.

To assess performance, we compare our Cluster Shapley algorithm against three widely used approx-

imation methods: Monte Carlo, Truncated Monte Carlo, and Kernel SHAP. Our results show that Cluster Shapley achieves substantial computational savings while maintaining comparable accuracy. Specifically, our method requires only 20–40 unique permutations out of a total of 255 to reach a Mean Absolute Error (MAE) of 0.20, whereas baseline methods—such as permutation sampling, Truncated Monte Carlo, and Kernel SHAP—require at least 110 permutations to achieve the same accuracy level. On average, our approach reduces computation time by 40% while maintaining high accuracy, achieving an MAE of 0.0913 and MAPE of 11.85% when choosing a clustering parameter $\epsilon = 0.20$. This significant improvement in efficiency establishes Cluster Shapley as a practical solution for large-scale Shapley value computation in LLM-based systems.

In summary, our paper makes two key contributions. First, we address the critical yet underexplored challenge of document valuation in LLM-generated summaries. By leveraging Shapley values, we propose a framework for fair and transparent document attribution, marking the first study, to the best of our knowledge, that applies this concept to LLM-based search systems. Second, we introduce the Cluster Shapley algorithm, which enhances the efficiency of Shapley value computation by leveraging semantic similarity among documents. This method significantly reduces computational costs while preserving attribution accuracy, making it well-suited for large-scale LLM-driven applications that summarize documents.

2 Related Literature

Our work intersects with several strands of literature. First, a substantial body of prior research has shown that online reviews can have a significant impact on sales and consumer behavior; see the seminal paper by Chevalier and Mayzlin (2006) and the review of this research stream by Seiler et al. (2018). While these studies primarily focus on the valence of reviews, our work extends this literature by examining how to attribute value when reviews are synthesized and summarized by LLMs.

Second, our work builds upon advances in RAG framework for text summarization. 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 in §6.2 is designed to be agnostic to specific summarization techniques, ensuring its applicability even as LLM and RAG technologies advance.

Finally, our methodology 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). 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. Additionally, our proposed Cluster Shapley algorithm addresses the computational complexity challenges inherent in Shapley approximation methods (Mann and Shapley, 1960; Ghorbani and Zou, 2019). Alternative approaches, such as leave-one-out (Cook, 1977) and the influence function approach (Barshan et al., 2020; Han et al., 2020; Guo et al., 2020), have also been explored for valuation. However, these methods often struggle in scenarios where multiple similar reviews exist, as they tend to assign nearly zero value to all redundant reviews. This limitation arises because the LLM summarization's effectiveness does not significantly improve with duplicated content, making such methods unsuitable for capturing nuanced contributions in document valuation. By contrast, our Cluster Shapley algorithm explicitly accounts for semantic similarity, ensuring a fair and computationally efficient valuation framework for LLM-generated summaries.

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 between 0 and ∞ to generate a summary in response to query q. Note that in practice, the platform can 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 oracle 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). 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 derive an equitable valuation for each document $i \in D$. In our setting, summarization, and evaluation are fully under the platform's control, which guarantees the valuation is incentive-compatible as players (content providers) cannot manipulate these processes. Let $\phi_i(D, q, A(q, S_q), v(q, A(q, S_q) \in \mathbb{R}$ denote the value of document i in dataset D, for query q, given summarization $A(q, S_q)$ and score

$v(q, A(q, S_q)$. Then, we can write the value of each document i over all the queries as:

$$\rho_i\left(D, A(\cdot), v(\cdot), g(\cdot)\right) = \int \phi_i\left(D, q, A(q, S_q), v(q, A(q, S_q))\right) g(q) dq, \tag{1}$$

where the value of each document i for a given query q is integrated over the distribution of queries $g(q)$.

Note that this characterization has two advantages. First, documents that are irrelevant to a query q and therefore not used in the summarization response can be automatically assigned zero value, i.e., $\phi_i = 0$ if $i \notin S_q$. Second, it allows us to weigh the relative importance of a document within a query with the prevalence of the query itself. For example, one document i may be critical to generate a high-quality summary for a niche query q_1, whereas another document (i') may be somewhat useful for a very popular query (q_2). In such cases, while document i should be valued highly for the query q_1, the overall value of the document for the platform shouldn't be very high since the query itself is rare. In contrast, while document i' is only marginally important for query q_2, the popularity of query q_2 can imply a high platform-level value for this document.

Our goal is to develop a 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. While we present a standard RAG implementation in §6.2, numerous alternatives exist—from simpler methods to more sophisticated frameworks like TextRank and GraphRAG (Mihalcea and Tarau, 2004; Edge et al., 2024). As LLM technologies rapidly evolve, our document valuation framework is designed to remain effective regardless of underlying summarization advancements, ensuring broad applicability across current and future implementations.
- Evaluation Process Agnostic: The approach should apply to any scoring 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 §6.3

In sum, our goal is to develop a solution concept that is agnostic to the details of the generative summarization model used (A) and the scoring procedure ($v(\cdot)$), and is broadly applicable across a variety of domains and business applications of LLM summaries.

4 Solution Concept: Shapley Framework for Document Valuation

This section develops a principled framework for the problem of document valuation in LLM-based summaries, as defined in §3. We begin by introducing the Shapley value, a game theoretical framework for document valuation in cooperative settings in §4.1, and then elaborate on how it is applicable to our problem. Next, we discuss the challenges associated with calculating Shapley values and note that it is not practically applicable in most settings (including ours). Therefore, in §4.2, we propose an approximation algorithm that

uses similarity measures obtained from the LLM itself to reduce the computational complexity associated with Shapley calculations. All the implementation details are discussed in §6.

4.1 Shapley Value

We now introduce the Shapley value formula along with a concise, self-contained explanation of the framework. Based on §3, recall that our goal is to find a document valuation function $\rho_i\left(D, A(\cdot), v(\cdot), g(\cdot)\right) \in \mathbb{R}$ to quantify the value of document i in set D. To obtain this valuation, we need to first accurately estimate the query-level document valuation function $\phi_i\left(D, q, A(q, S_q), v(q, A(q, S_q)\right)$. Henceforth, we denote this value function as $\phi_i(q)$ because the retrieval process S_q, the LLM-based summarization process $A(q, S_q)$, and the performance score function $v(q, A(q, S_q))$ are all uniquely defined by q.

Following the standard Shapley literature, we adapt and present the following four properties of $\phi_i(q)$ within the context of our problem to guarantee **equitable document valuation**, which we believe are important for our LLM tasks.

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:

$$\sum_{i \in D} \phi_i(q) = v(q, A(q, D)). \tag{2}$$

2. **Symmetry.** Symmetry ensures that documents with equal contributions are valued equitably. Formally, if i and j contribute equally to every possible coalition:

$$v(q, A(q, S \cup \{i\})) = v(q, A(q, S \cup \{j\})), \tag{3}$$

   for all subsets $S \subseteq D \setminus \{i, j\}$, then $\phi_i(q) = \phi_j(q)$.

3. **Null Document.** A null document implies that if a document provides no marginal value to any subset of reviews, 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 in a query q is null, then the value $\phi_i(q) = 0$.

   For example, for any document not used for the summarization $i \in D \setminus S_q$ which contributes no value to the user query q, its value is zero $\phi_i(q) = 0$.

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:

$$\phi_i(q_1 + q_2) = \phi_i(q_1) + \phi_i(q_2), \tag{4}$$

   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.

For linearity, first 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 combined value is $\phi_i(q_1 + q_2)$.

In practice, platforms often process a large number of queries rather than single queries in isolation. A document's overall value on such a platform should be defined using the performance score aggregated across all queries. For instance, the aggregated performance score could be expressed as, $v(q_1, A(q_1, S)) + v(q_2, A(q_2, S)) + \cdots + v(q_n, A(q_n, S))$. This approach is natural because queries typically arrive sequentially and are processed independently. The linearity property ensures that under this document value framework, it is sufficient to calculate the value at the query level and then sum them up to represent the total value across all queries on the platform accurately.

Linearity also has significant practical implications. It allows document values from different queries to be aggregated linearly, reflecting the total document value across multiple queries. This property is particularly crucial in our LLM setting, where subscription revenues from services like OpenAI's ChatGPT or Perplexity AI are typically generated in batches (e.g., monthly) rather than per query. Platforms can leverage this property to distribute revenue proportionally among document providers based on their aggregated contributions across all queries. For example, subscription revenue can be calculated by summing up document values over a period (e.g., a month) and allocating subscription fees proportionally to document providers. Additionally, query-level incentives, such as ChatGPT's per-query API charges or query-level advertising revenue, can still be calculated directly at the query level.

While other desirable properties are worth discussing, these four – Efficiency, Symmetry, Null Document, and Linearity – uniquely define the value function $\phi(q)$, Shapley value, with no further conditions needed. This is a well-known and foundational result (Shapley, 1953). We refer interested readers to Shapley's original paper for a formal proof. Notably, if the efficiency property is not enforced, the value function ϕ is only determined up to a proportional constant (see Proposition 2.1 in Ghorbani and Zou (2019)). In this paper, we only focus on how these properties apply to our context and their importance in our setting.

Under the above four properties, the Shapley value $\phi_i(q)$ for a document $i \in S_q \subseteq D$ is uniquely

expressed as the expected marginal contribution of document i across all possible coalitions:

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

This can be stated equivalently as:

$$\phi_i(q) = \frac{1}{|S_q|!} \sum_{\pi \in \Pi(S_q)} \left[v(q, A(q, P_i^\pi \cup \{i\})) - v(q, A(q, P_i^\pi)) \right], \tag{6}$$

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. For $i \notin S_q$, its Shapley value $\phi_i(q) = 0$ because only documents in S_q are used for summarization. Documents outside S_q have no contribution to the query and thus receive a Shapley value of zero. The Shapley value formula can also be written based on the original document set D:

$$\phi_i(q) = \frac{1}{|D|} \sum_{S \subseteq D \setminus \{i\}} \frac{v(q, A(q, S \cup \{i\})) - v(q, A(q, S))}{\binom{|D|-1}{|S|}}. \tag{7}$$

However, many of these evaluation score calculations are redundant since it is clear that any document outside S_q does not increase the performance score. In fact, this formula defined on D is equivalent to the formula defined only on S_q in Equation (5). The equivalence follows directly from the permutation-based definition in Equation (6), as all permutations of $D \setminus S_q$ do not affect the performance score.

While Shapley valuation is a theoretically appealing construct, evaluating Shapley values for source attribution presents a significant computational challenge due to their exponential complexity. Since Shapley values require assessing the marginal contribution of each document across all possible subsets, the number of evaluations scales as $2^{|S_q|} - 1$, where $2^{|S_q|}$ is the number of relevant documents. This rapid growth makes exact Shapley computation infeasible for large datasets, as the number of summarizations and evaluations quickly becomes overwhelming. Because each combination of documents requires an LLM to generate a new summary and undergo an evaluation step. While parallelization and batch processing can reduce latency, the overall computational burden remains substantial. As discussed in a later section §6.4, even for a query with eight documents, exact Shapley computation involves processing 255 subsets, leading to significant API costs and delays in LLM-settings. These constraints suggest that exact Shapley methods in large-scale applications (e.g., document valuation for large platforms using LLMs), necessitates efficient approximation algorithms.

4.2 Approximating Shapley Value: Cluster Shapley Approach

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

cost compared to exact Shapley, 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.

Motivated by this limitation, we propose a novel Cluster Shapley algorithm that integrates semantic information from text embeddings to improve efficiency while preserving accuracy. Instead of treating documents as independent, our method utilizes LLM-generated embeddings to identify and group similar documents, reducing redundant evaluations. The core idea is intuitive: documents with similar content should have comparable contributions to the final summary and, therefore, should receive similar Shapley values. The key strength of our approach is that it 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. These embedding vectors have been widely used in various applications, including text classification, document retrieval, and sentiment analysis (Patil et al., 2023). Modern LLM-based embeddings, such as OpenAI's latest text-embedding models, operate in high-dimensional spaces (e.g., 3072-dimensional vectors), leveraging Transformer architectures and extensive pretraining on large-scale corpora. Unlike generative LLMs optimized for text generation, embedding models are specifically designed to produce semantically meaningful representations that enhance downstream tasks. Their effectiveness has been demonstrated in various domains—for example, Ye et al. (2025) showed that OpenAI's 3072-dimensional embeddings outperform direct prompting of state-of-the-art GPT models in predicting the attractiveness of news headlines. In our setting, we leverage these embeddings to cluster similar documents before computing Shapley values, allowing us to reduce redundant calculations.

Formally, we outline Cluster Shapley in Algorithm 1. Our Cluster Shapley begins with a retrieval step, where for a given query q, we 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 §6.2, 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.

Because similar documents tend to have similar embeddings, making it possible to cluster them. 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. Specifically, we employ cosine similarity, a widely used metric for measuring the closeness of embeddings.

It is defined as:

$$\text{cosine\_similarity}(e_i, e_j) = \frac{e_i \cdot e_j}{\|e_i\|\|e_j\|}.$$

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 rang $[0, 1]$, and given by:

$$d(e_i, e_j) = 1 - \text{cosine\_similarity}(e_i, e_j) = 1 - \frac{e_i \cdot e_j}{\|e_i\|\|e_j\|}.$$

This definition ensures that similar documents, which have high cosine similarity, are assigned a smaller distance value. Consequently, 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

1: **Inputs:** Retrieval of relevant documents S_q for a query q and the corresponding embeddings $e_i,\ i \in S_q$. A control hyper-parameter ϵ for clustering.
2: **Step 1: Clustering.** Perform clustering on documents S_q based on similarity of documents to get clusters $\{G_1, G_2, \ldots, G_m\}$ such that $d(e_i, e_j) \leq \epsilon$ for any document i, j within the same cluster.
3: **Step 2: Shapley values of clusters.** Calculate the Shapley value for clusters $\{\hat{\phi}_{G_k}\}_{k=1}^m$.
4: **Outputs:** The Shapley value of all documents in group $k \in \{1, 2, \ldots, m\}$ is assigned as $\hat{\phi}_{G_k}/|G_k|$.

4.2.1 Clustering

Next, we discuss the clustering step (Step 1) in further detail. First, note that this step is agnostic to a specific clustering algorithm. The goal is to partition the documents into non-overlapping clusters, i.e., $D = \bigcup_{k=1}^m G_i$ with $G_i \cap G_j = \emptyset$ for $i \neq j$. One can employ some 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, unlike the standard clustering algorithm, we impose a 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 introduce this constraint because we found that it can improve the performance of the Cluster Shapley algorithm; see more details in Web Appendix E.4. 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.

To achieve this clustering goal, we propose Algorithm 2, 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 (as in algorithms such as K-Means). This makes DBSCAN particularly suitable for our task, where the number of clusters is not predetermined. Specifically, the standard DBSCAN operates through a density-based clustering mechanism, utilizing two key hyperparameters: the neighborhood radius ϵ' and `MinPts` (minimum points required to form a dense region). The algorithm identifies core points as those having at least `MinPts` points within their ϵ'-neighborhood, and constructs clusters through density-reachability—a property where points are connected through a chain of core points. Points that fall within the ϵ-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, it is important to note that standard DBSCAN does not guarantee that any two documents within the same cluster have strictly smaller distances than ϵ'. Because DBSCAN forms clusters based on local density connectivity rather than enforcing global distance constraints. Consider three points p_1, p_2, and p_3: if $d(p_1, p_2) \leq \epsilon'$ and $d(p_2, p_3) \leq \epsilon'$, DBSCAN will assign all three points to the same cluster through density-reachability, even if $d(p_1, p_3) > \epsilon'$. This transitive clustering property, while effective for many applications, can result in clusters where some point pairs exceed the specified ϵ' threshold. This limitation necessitates modifications to the standard DBSCAN algorithm to enforce stricter distance constraints for accurate Shapley value estimation in our context.

In our proposed Algorithm 2, we calculate the distance matrix using embeddings and the predefined distance function d, inputting this matrix into the DBSCAN algorithm. Second, we set the minimum number of points per cluster (a DBSCAN hyperparameter) to 1, ensuring individual reviews are not excluded as noise. However, standard DBSCAN does not guarantee that all documents within a cluster are within ϵ, nor does it ensure clusters are sufficiently separated. Therefore, after initial clustering with a preset $\epsilon' \leftarrow \epsilon$, we verify if all documents in the same cluster have distances smaller than ϵ. If a condition is violated, we iteratively reduce ϵ' by a factor of 0.95, i.e., $\epsilon' \leftarrow 0.95\epsilon'$, and rerun DBSCAN until the constraint is met.

4.2.2 Shapley Values of Clusters

In Step 2 of the 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 $k \in \{1, 2, \ldots, m\}$, significantly reducing the computational complexity. Specifically, following the formation of meta-documents, we proceed with the same summarization and evaluation process outlined in §4. These meta-documents, now representing aggregated information from multiple documents, are 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.

Notably, the 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

Algorithm 2 Iterative Distance-Constrained DBSCAN

Input: Distance matrix M with $M_{ij} = d(e_i, e_j)$. Hyperparameters: ϵ, $\texttt{MinPts} = 1$, scaling factor $\alpha = 0.95$

Initialize $\epsilon' \leftarrow \epsilon$ ▷ Start with the original ϵ

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

Run the standard DBSCAN:

- For each document i: find ϵ'-neighborhood $N_{\epsilon'}(i) = \{j : M_{ij} \leq \epsilon'\}$.
- If $|N_{\epsilon'}(i)| \geq \texttt{MinPts}$, mark i as a core point.
- Connect core points that are within ϵ' distance.
- Assign non-core points to clusters of nearby core points.

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 condition: Exit the loop. Otherwise, Update $\epsilon' \leftarrow \alpha \cdot \epsilon'$ 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.

techniques. In cases with very large datasets or a high number of clusters, approximation methods such as Monte Carlo sampling can be employed to further reduce computational costs while maintaining reasonable accuracy.

Finally, we attribute the cluster's Shapley value equally across its individual documents, assigning $\hat{\phi}_{G_k}/|G_k|$ as the Shapley value for each document within cluster G_k.

In summary, our Cluster Shapley Algorithm offers two key advantages. First, it reduces complexity by grouping similar documents into clusters, which decreases the number of subsets to process from $2^{|S_q|} - 1$ to $2^m - 1$ (m is the number of clusters), thereby significantly reducing computation time for summarization and evaluation. Second, it provides flexibility in balancing speed and precision through the adjustable ϵ parameter. For tasks requiring high precision, a smaller ϵ can be selected, while tasks prioritizing speed can utilize a larger ϵ.

5 Application Setting: Amazon Review Dataset

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 extensively utilized 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. This component contains 30.14 billion review tokens. 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 the reviews. The dataset exhibits a highly skewed distribution of the number of reviews, where nearly 90% of products have fewer than 25 reviews, the most frequently purchased products tend to have hundreds or even thousands of reviews. For example, the top 2% of products alone account for over 40% of all reviews, meaning that for popular items, consumers must 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 (and did) partially 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.

Today, Amazon has 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 ease of computation cost.

To compute the Shapley value of each individual review (within the context of a given product), we, therefore, need to first specify the distribution of queries that consumers use when requesting summaries for this product. However, this query distribution cannot be accessed. Instead, we crafted two user queries that closely mimic the real distribution of queries observed on e-commerce platforms. Specifically, we selected popular attributes frequently mentioned by customers. As shown in Figure 1, the center subfigure under the AI-summarized review highlights several commonly discussed aspects of products. For instance, for the wireless controller product, attributes such as "Functionality" and "Controller quality" are among the most commonly mentioned aspects by users. Based on this observed information, we designed two queries for

each product, as shown in the last column in Table 1, ensuring they aligned with real consumer concerns and reflected realistic evaluation scenarios.

No.	Product	Number of Reviews	Designed Query
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?

Continued on next page

No.	Product	Number of Reviews	Designed Query
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?
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: Examples of designed queries for different products

6 Implementation Details of Shapley Value

This section is organized as follows. In §6.1, we first introduce the RAG architecture. Recall that the two key inputs to our document valuation framework are a summarization method $A(\cdot)$ and an evaluation method $v(\cdot)$. As discussed in §4, while our solution concept is agnostic to summarization and evaluation methods, we nevertheless need to specify these two methods for our empirical evaluations. In §6.2 and §6.3, we detail our summarization $A(\cdot)$ and evaluation $v(\cdot)$. Last, we carry out implementation details towards exact Shapley and Cluster Shapley in §6.4 and §6.5 respectively.

6.1 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. 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 at hand.
- **Augmentation (A):** In this phase, the retrieved documents (S_q) is 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.

By combining these three components, RAG architecture enhances AI search systems in two fundamental aspects: technical system capability and user experience. From a technical perspective, this integration enables AI systems to generate responses that leverage both pre-trained model knowledge and real-time retrieved information, significantly expanding their capabilities beyond the constraints of static training data (Jiang et al., 2023). From a user interaction perspective, RAG architecture improves three critical dimensions of system trustworthiness: source verification, decision support, and accountability (Perplexity, 2024). Specifically, by exposing information sources and their integration process, RAG systems enable users to verify response provenance and understand the basis of AI-generated content. This transparency is particularly important in real-world applications such as online shopping, news, and finance, where understanding and validating the foundation of AI responses directly impacts decision-making processes.

Given the effectiveness and scalability of RAG architecture, it 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). In the retriever phase, these systems adapt their information retrieval strategies based on their specific objectives. For instance, ChatGPT Search leverages Bing's search engine capabilities, enhanced by strategic partnerships with news

organizations like Associated Press and Financial Times to ensure access to verified information (OpenAI, 2024). Google's implementation employs a dynamic search mechanism by assigning a prediction score (ranging from 0 to 1) to each query, measuring the potential benefit of incorporating external information. When this score exceeds a configurable threshold, the system activates RAG to ground responses with retrieved web data; otherwise, it defaults to the conventional LLM model, thus optimizing computational efficiency (Google, 2024). The augmentation and generation phases are also adapted to their specific contexts. For example, in the augmentation phase of ChatGPT Search, the retrieved information from both web searches and partnered content providers is processed and formatted to align with the user's query context. This augmented context is then passed to the generation phase, where outputs from OpenAI's o1-preview model are used to create synthetic training datasets for fine-tuning the more efficient GPT-4o model[1]. This approach enables comparable task-specific performance while significantly reducing computational costs. The generated responses maintain conversational coherence while preserving source attribution through embedded references.

To illustrate the impact of RAG integration on LLMs, we provide an example with a simple query: "Who won the Super Bowl this year?" for two cases – (1) LLM response without RAG integration (see Figure 2) and (2) LLM with RAG integration (see Figure 3 for ChatGPT Search). As we can see RAG integration has key three advantages over the baseline LLM: (1) the ability to interpret temporal references without explicit dates, requiring systems to understand and ground "this year" in current context, and (2) the capacity to provide accurate, up-to-date information through real-time retrieval rather than relying on training data, and (3) The ability to clearly cite the sources used to answer the query, which allows the user to verify the accuracy of the generated answer. Notice that, without RAG integration, GPT-4 is constrained by its training cutoff date and provides outdated information from 2023, whereas both RAG-enabled systems successfully deliver current, verified information with proper source attribution.

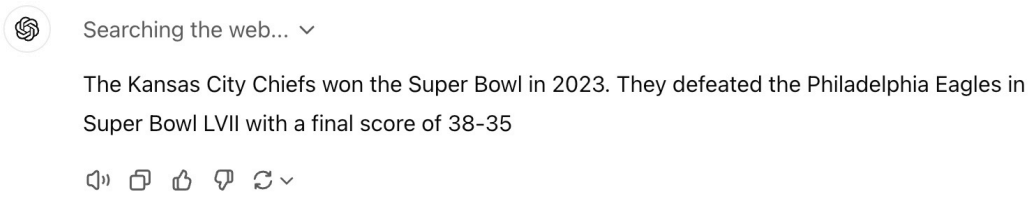


Figure 2: Traditional ChatGPT-4 Response without RAG-Enhanced Web Search

6.2 Summarization of Relevant Amazon Reviews via RAG

Note that there exist a few open-source RAG-based AI search engines such as (Morphic, 2024; Perplexica, 2024). However, they typically rely on web search APIs, e.g., (SearXNG, 2021), which do not include the Amazon review dataset. Thus, in this part, we construct a RAG-based search to find relevant documents S_q

[1]Distillation is a model compression technique where a smaller model is trained to emulate the behavior of a larger, more powerful model. For technical details, see: https://openai.com/index/api-model-distillation/

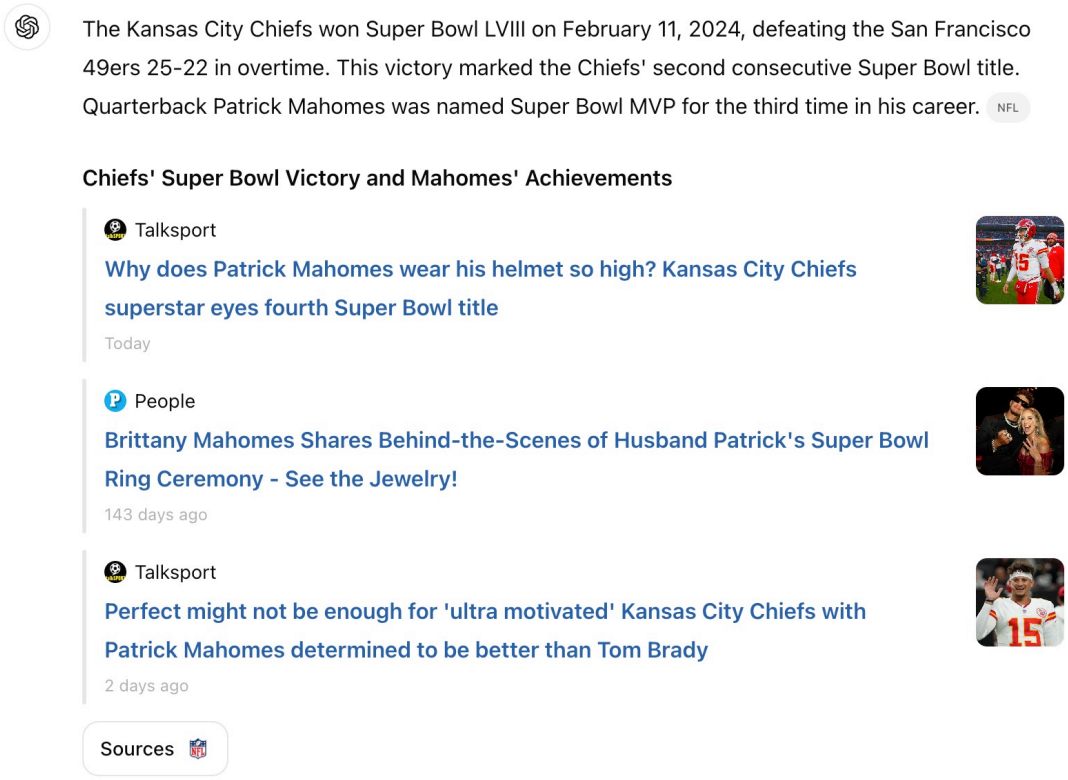


Figure 3: ChatGPT Search with RAG-Enhanced Web Search

from Amazon reviews and summarization tool $A(q, S_q)$ for any given query q. Figure 4 shows the overview of the four-step procedure.

- **Step 0: Generate Text Embeddings**
  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) and capture the information in the review text. In our analysis, we exclude reviews with fewer than 10 words, as they tend to be incomplete or non-informative.
  Note that 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

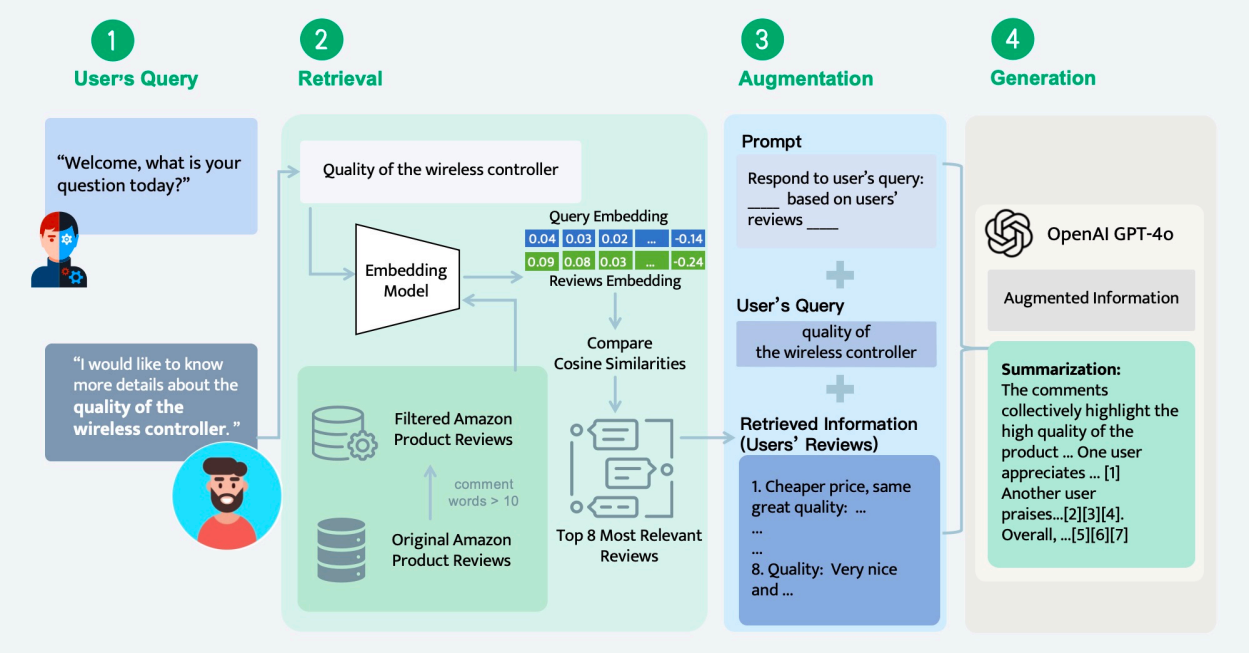


Figure 4: Architecture of LLM-based search and summarization tool for Amazon Product Reviews. This flowchart illustrates the architecture of an AI-powered search engine designed for processing and summarizing reviews about the quality of DualShock 4 Wireless Controller. The process starts with the user 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 8 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.

We first process the user query to extract the key semantic information in it using a 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 4, 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 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:

$$\text{cosine\_similarity}(e_i, e_q) = \frac{e_i \cdot e_q}{\|e_i\| \|e_q\|},$$

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, refer to Chapter 6 of (Schütze et al., 2008).

The cosine similarity of a pair of embedding vectors captures the extent to which the embedding 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. Note that in our case, we choose $|S_q| = 8$ across different queries. We can either retrieve the top few documents or apply a cutoff based on the cosine similarity of the embedding values—excluding 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. This retrieval process is quite flexible, and our Shapley approach is agnostic to the specific retrieval process. To keep the implementation simple, we apply the rule to select the top 8 most relevant reviews, though, in practice, the retrieval process can be much more complex.

Note that we select the top 8 most relevant reviews instead of using all D reviews/documents for three reasons. First, from a computational and monetary cost perspective, providing a very large piece of text (e.g., consisting of all available reviews, including irrelevant reviews) can be costly since the LLM has to process all the tokens associated with this text as context and generate text (see the next step). Second, giving irrelevant context to the generative model can worsen the quality of summaries, which is detrimental to consumers. Third, prior research has shown that excessive information (or information overload) can hinder customers' ability to process content effectively (Jacoby et al., 1974; Eppler and Mengis, 2004). Indeed, a recent study on modern AI search engines (such as Perplexity AI) shows that responses typically include between 5 and 8 citations, with an average of 5.28 sources per query (Danny, 2024).

- **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 custom-designed prompt in OpenAI's GPT-4o model (`gpt-4o-2024-08-06`). The prompt, as shown in Figure 5, instructs the model to analyze the filtered reviews, exclude irrelevant information, and generate a summary focusing solely on content related to the query. The prompt ensures that the summaries remain focused, neutral in tone, and transparently cite sources in square brackets, e.g., [1], so that each summarized detail can be traced back to its source. If a review is deemed irrelevant, the model explicitly marks it with a note like "[X] is not related to the query." This systematic approach guarantees that both relevant and irrelevant reviews are transparently accounted for, ensuring clarity and precision in the final output.
- **Step 4: Generate the summary $A(q, S_q)$**
  Following the prompt in Figure 5, the LLM model generates a summary response to the user query based that combines the information from both the original corpus that the LLM was trained on and the new context (the set of eight relevant reviews provided in the prompt). The summary clearly cites the source reviews when needed, so users can refer to the original document when necessary.

```
Inputs:
- Original user query.
- Indexed Amazon reviews.

GPT Prompt:
1. Analyze all reviews related to the query main topic.
2. Include only relevant information in the summary.
3. Use an objective and neutral tone.
4. Cite sources by appending the review's index to the end of each sentence.
5. State explicitly if a review is unrelated to the query: "[X] is not related to the
    query."

Structured Outputs:
- A summary covering the key information related to the query.
- JSON formatted output with fields:
  - "key": A string representing the indices of the reviews used.
  - "summary": The final summary text, with appropriate citations.
```

Figure 5: Summarization prompt on 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. The complete prompt is provided in Appendix A

6.3 Evaluation of Summarized Amazon Reviews

Next, we carry out the details of the performance score function $v(\cdot)$, which essentially evaluates the quality of summaries. We design an LLM prompt as the performance score function, as shown in Figure 6. This prompt provides summaries along with the original query to generate 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. Note that our framework is evaluation-tool agnostic, allowing the integration of fine-tuned evaluation LLMs that are based on human-labeled data, providing flexibility for incorporating more efficient and tailored evaluation mechanisms into the document valuation process.

Using the prompt, each summary is rated on a scale from 0 to 10, 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. If a summary includes irrelevant content—such as the explicitly marked phrase "[X] is not related to the query"—it is assigned a minimum score of 0. 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 scoring.

```
Inputs:
- "{original_query}": Original user query.
- "{summaries}": Contains multiple summaries generated from different subsets of
    reviews. Each summary is indexed with reference numbers corresponding to the
    original reviews.

GPT Prompt:
1. Read the provided "{summaries}" carefully and compare them.
2. Evaluate each summary based on the criterion "Information Coverage," which
    measures how well the summary captures and describes the key characteristics of
    the product mentioned in "{original_query}".
3. Use a score between 0 and 10 to rate the summary, where a higher score indicates
    more comprehensive and clearer coverage of the product's key features.
4. If a summary contains only the sentence "[X] is not related to the query." then
    assign a score of 0. If the summary includes other relevant content, ignore the
    irrelevant sentence when evaluating.

Structured Output:
- A JSON formatted output with the fields:
  - "key": A string representing the indices of the reviews used in the summary.
  - "score": An integer from 0 to 10 representing the score of the summary based on
      "Information Coverage".
```

Figure 6: Evaluation prompt (simplified version): The prompt guides the evaluation of how well each summary captures and describes key characteristics from the source reviews. The complete evaluation prompt with detailed scoring criteria is provided in Appendix A.

Note that we use the same GPT-4o model as the one used for summarization, to evaluate summaries. One potential issue is that using the same LLM from OpenAI for both summarization and evaluation may introduce biases, i.e., LLM tends to give higher scores to its summaries. To address this, we conducted robustness checks using a different model, Claude, for evaluation. The results show that using Claude yields similar performance scores and thus similar Shapley values, demonstrating the consistency of our framework across different LLMs. Detailed results of these experiments are provided in Appendix D.

One challenge of using the GPT model for evaluation is the inherent stochastic nature of their outputs. To address this, we leverage OpenAI's structured output technology, which enforces strict schema adherence in model responses. The structured outputs ensure that the LLM returns integer scores between 0 and 10. Another method to reduce output randomness is by adjusting the temperature parameter, which modulates response variability [2]. While a temperature of 0 theoretically produces deterministic outputs by always selecting the highest probability token, residual variability persists even at this setting due to several fundamental challenges in LLM computation. Most notably, hardware-level floating-point arithmetic variations, a common

[2]The LLM temperature serves as a critical parameter influencing the balance between predictability and creativity in generated text. Lower temperatures prioritize exploiting learned patterns, yielding more deterministic outputs, while higher temperatures encourage exploration, fostering diversity and innovation.

characteristic across all GPU-based LLM inference processes, can introduce subtle rounding errors that differ across devices, architectures, and even computational threads (Astekin et al., 2024; Renze and Guven, 2024). The impact of these computational variations becomes particularly pronounced in large language models processing extended text sequences, where subtle distributional shifts compound across the sequence length. To quantitatively assess the impact of temperature settings, we conducted systematic experiments examining both summarization and evaluation processes across different temperature values (detailed in Appendix B). Our findings demonstrate persistent output variance even at temperature 0, confirming these theoretical predictions. Moreover, setting the temperature to 0 while minimizing variability tends to produce overly rigid and potentially less coherent outputs (Holtzman et al., 2020), which is suboptimal for natural language generation tasks. Based on our specific task and analysis, we selected a temperature setting of 0.1 for both summarization and evaluation tasks. For summarization, this setting maintains output stability while allowing sufficient linguistic flexibility to generate natural and contextually appropriate responses. For evaluation, we adopted the same temperature setting to maintain methodological consistency across our pipeline. This balanced approach effectively addresses the competing demands of reliability and expressiveness in both generation and evaluation processes.

To further reduce the inherent variance in LLM output scores, we conduct four independent evaluations for each summary and use their average as the final score. This approach balances variance reduction with computational efficiency. Although increasing the number of evaluations can improve robustness, it also raises time and resource demands. Through experimentation, we found that averaging four evaluations achieves sufficient accuracy without excessive computational overhead. A formal variance analysis of the summarization and evaluation process is presented in Web Appendix C.

6.4 Exact Shapley Implementation

Following the evaluation phase, where scores are assigned to all subsets of reviews based on the structured prompt described previously, we calculate the exact Shapley values using the formula in (5). Table 2 presents the Shapley values for the top 8 most relevant reviews in response to the query, "How is the quality of the wireless controller?". Other reviews not contributing to this query receive a Shapley value of zero.

Review #3 has the highest Shapley value (1.84), 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 (1.63) and Review #2 (1.56) score highly for addressing quality explicitly—#7 in a positive tone and #2 by highlighting durability compared to off-brand controllers. Review #5 (1.40) also performs well by underscoring the superior quality of the original controller versus knockoff brands. Review #4 (1.27) is somewhere in the middle, highlighting the good quality but without additional information relevant to the query. In contrast, lower-scoring reviews reveal interesting patterns. Review #1, with a Shapley value of 0.73, has a somewhat misleading structure: while the title suggests "same great quality," the review text itself focuses solely on price comparisons, lacking specific details on quality. This mismatch results in a lower Shapley value. Reviews #6 and #8, with values of 0.29 and 0.28, respectively, receive the lowest scores, as they emphasize aspects like price, shipping, or gift satisfaction, which are less

relevant to the query's focus on controller quality.

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 implementation cost of our proposed Shapley-based document valuation. 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. Our experiments indicate that processing a single query averages 15 minutes,[3] costing approximately $1.30 in OpenAI API fees per query. Batch processing, i.e., simultaneous API calls to OpenAI, can effectively reduce the processing time from 15 minutes to around 3.5 seconds by parallelizing the 255 summarizations and evaluations. However, total computation time and cost remain unchanged. Alternatively, open-source LLMs for summarization and evaluation can further reduce both time and costs. For simplicity, we report the total computation time based on sequential GPT-4o processing throughout the paper.

Despite these optimizations, the computational and financial overhead in the exact Shapley value remains high for large-scale applications like Perplexity AI, which processed 400 million queries for 15 million monthly active users as of August 2024 (DemandSage, 2024b). This high resource demand highlights a significant gap between theoretical document valuation frameworks and practical implementation, emphasizing the need for an efficient approximation algorithm, like our proposed Cluster Shapley algorithm.

[3]This time includes the full process for both summarization and evaluation of Python-based API calls, network latency, time to first token, and all computational overheads. Processing time depends significantly on the OpenAI API tier level; our experiments used Tier 2 access.

6.5 Cluster Shapley Implementation

For the implementation of our proposed Cluster Shapley, as described in Algorithm 1, we use OpenAI's `text-embedding-3-large` model to generate documents' embeddings and then calculate the distance matrix, which is used for clustering.

In Step 1, we leverage Algorithm 2 for clustering. We explore a range of clustering control hyperparameter ϵ values from 0.01 to 1.00 (in increments of 0.025) and set the minimum number of reviews per cluster to 1, allowing individual reviews to form separate clusters if necessary. The computational cost of our proposed adaptive clustering algorithm is relatively low in our setting, requiring only nanoseconds per iteration, compared to the several seconds needed for LLM summarization and evaluation. We provide more empirical details for this computational efficiency in Web Appendix E.1. For completeness, we also report the performance of Cluster Shapley using the standard (non-adaptive) DBSCAN in Web Appendix E.4.

To illustrate the clustering process, Figure 7 presents the clustering results for a sample query using two-dimensional OpenAI embeddings. In this instance, we set $\epsilon = 0.05$, 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, after getting clusters of documents with similar sentiments, we use the exact Shapley value calculations to determine each cluster's value. The summarization and evaluation steps follow exactly the same prompts as the one used in the exact Shapley calculation, as shown in Figure 5 and Figure 6 respectively, except that we use clusters of documents as the input for summarization. It's worth noting that in Step 2, any Shapley value algorithm can be used. In cases with very large datasets or numerous clusters, one might employ approximation algorithms, such as Monte Carlo, to reduce the computation cost. However, due to the manageable number of document clusters in our setting, we just use the exact Shapley value calculation as outlined in §4.

Last, we distribute these values of clusters equally back to the individual documents within the clusters.

7 Results

In this section, we present the results of applying our Cluster Shapley algorithm to the Amazon review setting. In §7.1, we first discuss a set of alternative Shapley approximation algorithms that can serve as a benchmark for our proposed algorithm. Then, in §7.2, we present the numerical results from our approach and present comparisons to the other benchmark algorithms.

7.1 Benchmark Algorithms

We now briefly summarize three widely-used Shapley value approximation algorithms, that serve as benchmarks against which we compare the performance of our proposed algorithm.

- **Monte Carlo:** The Monte Carlo algorithm (or permutation sampling) is a widely adopted approach for approximating Shapley values (Mann and Shapley, 1960). This method randomly samples permutations from the $|S_q|!$ possible combinations of documents (see Equation (6)) and then for each document i and

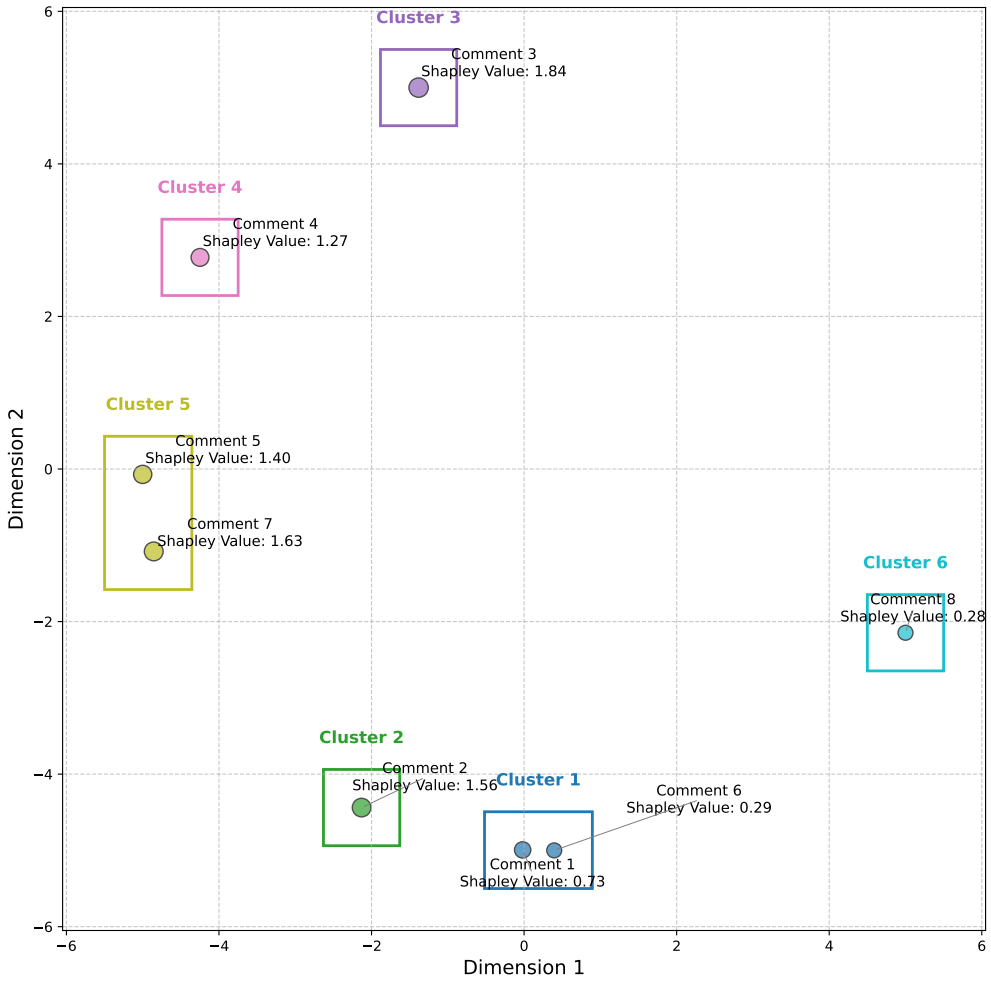


Figure 7: Clustering of Top 8 relevant Amazon reviews for the query "How is the quality of the wireless controller?"

one permutation P_i^π, calculate its marginal contribution, i.e., $v(q, A(q, P_i^\pi \cup \{i\})) - v(q, A(q, P_i^\pi))$. Last, the Shapley can 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 Truncated Monte Carlo 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 $S_2 \subseteq S_1$, meaning the marginal contribution of document i decreases when more documents come into the permutation. This is because with a larger set of permutation, document i has higher overlapped information, reducing its marginal contribution.

  We briefly summarize the algorithm here, while referring to Ghorbani and Zou (2019) for full 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 this negligible marginal value. This performance tolerance parameter, which controls the allowable change in Shapley values before truncation occurs, is tested across multiple values $\{0.1, 0.2, 0.3, 0.5, 0.7, 1, 2, 3\}$. For our experiments, we use 0.5, as smaller values reduce the effectiveness of truncation, making Truncated Monte Carlo behave similarly to the standard Monte Carlo method, while larger values cause premature truncation that compromises estimation accuracy. In our numerical experiments, we also test progressively larger numbers of permutations for comparison, although different stopping or convergence criteria can be used in practice.

- **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 the KernelExplainer with an identity link function and L1 regularization to enhance numerical stability. We test increasing numbers of samples to evaluate the method's performance under different computational budgets.

7.2 Numerical Results

Our numerical experiments include 48 test queries, designed as 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 (from all 255 possible subsets) and fix the evaluation score for each summary by averaging 4 evaluations, as detailed in §6. By fixing sample paths, we mitigate the inherent randomness in LLM outputs, ensuring consistent baseline measurements across different approximation methods.

We visualize the performance of different Shapley value approximation methods in Figure 8. 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 for additional performance metrics, including Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE), exhibit similar trends and can be found in Web Appendix E.3. The X-axis represents the number of unique subsets/samples 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 subset combinations that emerge after clustering, where each cluster is treated as a new meta-review. For Monte Carlo and

Truncated Monte Carlo, while these methods theoretically sample from all possible permutations, we count only the unique subsets encountered during sampling to ensure fair computational comparison. For example, if the same subset appears in multiple permutations, we only evaluate it once and cache its result for reuse. For Kernel SHAP, we similarly track unique coalition combinations that require actual LLM evaluation rather than the total number of samples used in the weighted regression. While algorithms like Monte Carlo, Truncated Monte Carlo, and Kernel SHAP, allow replicated subsets/samples, we can just store computed values of summarizations with negligible cost to avoid redundant computations, ensuring that the actual computation cost 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, clearly comparing their performance relative to computational effort.

Figure 8's X-axis extends to 180 rather than 255 (the total possible subsets) for two key reasons tied to algorithmic characteristics. For Truncated Monte Carlo, this limitation emerges from its early stopping mechanism. While we could theoretically force the algorithm to evaluate more subsets by setting an extremely small performance tolerance threshold, doing so would effectively eliminate the key advantage of Truncated Monte Carlo over standard Monte Carlo sampling. For Cluster Shapley, even with very small ϵ values, the algorithm naturally forms clusters due to inherent similarities in review content. For instance, commonly occurring enthusiastic reviews like "the product is awesome, I love it" or similar short positive expressions tend to form semantic clusters regardless of ϵ settings. In our experiments with the Amazon review dataset, as we decreased ϵ to extremely small values ($\epsilon < 0.01$), we observed that the algorithm consistently converged to around 180 unique subsets. This natural boundary in our experimental results suggests that comparing performance up to 255 subsets would not be meaningful, as none of the clustering configurations in our tests reached this number.

Next, we compare and discuss the performance of different algorithms. First, Cluster Shapley achieves the best performance because the responding curve is below all other curves, as evidenced in Figure 8. It means that Cluster Shapley has smaller approximation errors than other methods under the same computation cost/time, or Cluster Shaply requires the least computation time to achieve the same accuracy. The Truncated Monte Carlo performs the second best because it leverages the bounded performance score for early stopping.

Second, this superior performance is particularly pronounced in the early phase of computation, where Cluster Shapley achieves an MAE of around 0.2 while other algorithms exhibit significantly higher approximation errors (MAE > 0.4). This efficiency stems from clustering's ability to capture fundamental similarity patterns among reviews, even with relatively few clusters, effectively reducing the dimensionality of the approximation problem. However, as the number of evaluated subsets increases (the number of clusters increases), the relative advantage gradually diminishes. When using a large number of subsets (more than 150), the advantage of Cluster Shapley becomes marginal. This is because setting ϵ too small results in an excessive number of clusters, eventually leading to some clusters containing only one review. In these cases, those single-review clusters no longer benefit from the similarity-based dimensionality reduction, thereby reducing the overall advantage of the clustering approach.

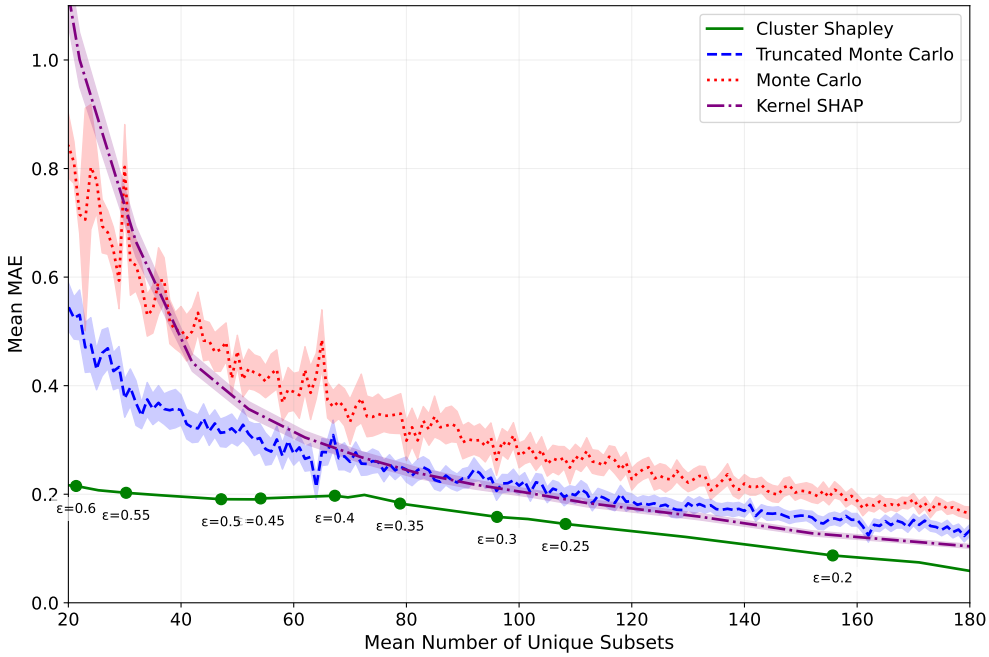


Figure 8: Performance of Shapley value approximation methods. The Y-axis represents the Mean Absolute Error (MAE) of the Shapley values, averaged across all test instances and reviews, while the X-axis represents the number of unique subsets or samples used by the algorithms. The points on the Cluster Shapley curve correspond to different epsilon settings. For reference of how large the MAE is, the average Shapley value over all test samples is 1.084, indicating 0.2 MAE is around 20% percentage error. 95% CIs for Monte Carlo, Truncated Monte Carlo, and Kernel SHAP are computed through 10 replications of the algorithms.

Third, all algorithms exhibit decreasing MAE as the number of unique subsets increases, aligning with theoretical expectations. Monte Carlo and Truncated Monte Carlo show substantial fluctuations and larger confidence intervals in their error curves due to their inherent sampling randomness. In contrast, Cluster Shapley demonstrates significantly more stable and deterministic performance throughout the computation range, though it exhibits a plateau when starting with larger ϵ values (fewer clusters). During this plateau phase, while the clustering structure effectively captures essential similarity patterns, it lacks the granularity to perform more fine-grained groupings. As ϵ decreases, the clustering becomes more refined, enabling the detection of more subtle similarity relationships and leading to a steadily decreasing trend in MAE before eventually reaching the diminishing returns phase discussed above.

To further analyze Cluster Shapley, we examine its computational efficiency trade-offs, as detailed in Table 3. The exact Shapley calculation requires evaluating all 255 possible unique subsets, with each subset evaluation taking an average of 3.53 seconds, resulting in a total computation time of approximately 15 minutes per query. While processing clustered reviews introduces slightly more tokens compared to individual reviews, this marginal increase in token count results in minimal variation from the average computation time per evaluation. The primary computational cost comes from the number of subset evaluations required. The table demonstrates that increasing the clustering parameter ϵ leads to greater computational savings but at the cost of accuracy. Notably, at $\epsilon = 0.20$, the algorithm achieves a 40% reduction in computation time while

Clustering Parameter (ϵ)	MAE	RMSE	MAPE	Reduction Percentage
0.01	0.0381	0.0461	7.47%	23.01%
0.10	0.0507	0.0604	8.62%	26.67%
0.20	0.0913	0.1110	11.85%	40.00%
0.30	0.1617	0.1957	17.16%	62.39%
0.40	0.1972	0.2404	21.35%	73.61%
0.50	0.1908	0.2417	21.05%	81.52%
0.60	0.2152	0.2691	24.43%	91.62%
0.70	0.2305	0.2879	26.49%	98.63%
0.80	0.2259	0.2821	26.33%	99.13%

Table 3: Approximation error and computation time reduction of Cluster Shapley under varying ϵ. Computational savings are calculated based on the percentage reduction in required unique subset evaluations compared to the exact Shapley calculation. Note that we add a small constant (0.1, approximately 10% of the mean Shapley value) to all values to address the limitation of MAPE with near-zero values while minimizing the distortion of the original value relationships.

maintaining reasonable accuracy with an MAE of 0.0913 and MAPE of 11.85%. This represents an attractive balance point between efficiency and accuracy.

A key advantage of our proposed algorithm, Cluster Shapley, is its ability to leverage semantic similarity in LLM embeddings of reviews/documents. Unlike other approximation methods, such as Monte Carlo, Truncated Monte Carlo, and Kernel SHAP, which rely on random sampling without utilizing intrinsic semantic properties, Cluster Shapley exploits the semantic similarities encoded in embeddings. By clustering documents based on semantic similarity, our approach achieves more accurate and computationally efficient Shapley value approximations, underscoring the importance of intrinsic semantic information in document valuation. This demonstrates the power of advanced textual representations from LLMs in enhancing document valuation frameworks.

8 Conclusion

In this paper, we introduce a novel framework for document valuation in LLM-generated summaries using Shapley values, providing a systematic and transparent approach to quantifying the document contribution. To address the computational challenges of exact Shapley value calculations, we propose the Cluster Shapley algorithm, which leverages embedding similarity to significantly reduce computation time while maintaining accuracy. Our method outperforms several benchmarks in efficiency and scalability, demonstrating its practical applicability to LLM-based systems. This work represents the first application of Shapley values for source attribution in LLM summaries, laying the foundation for fair and efficient document valuation across various AI-driven applications.

Funding and Competing Interests Declaration

Author(s) have no competing interests to declare.

References

Merve Astekin, Max Hort, and Leon Moonen. An exploratory study on how non-determinism in large language models affects log parsing. In *Proceedings of the ACM/IEEE 2nd International Workshop on Interpretability, Robustness, and Benchmarking in Neural Software Engineering*, InteNSE '24, pages 13–18, New York, NY, USA, 2024. Association for Computing Machinery. doi: 10.1145/3643661.3643952. URL https://doi.org/10.1145/3643661.3643952.

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.

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.

Cristina Criddle. Perplexity in talks with top brands on ads model as it challenges google, 2024. URL https://www.ft.com/content/ecf299f4-e0a9-468b-af06-8a94e5f0b1f4. Accessed: 2024-10-19.

Goodwin Danny. 60% of perplexity citations overlap with top 10 google organic results, 2024. URL https://searchengineland.com/perplexity-citations-top-10-google-organic-results-439029. Accessed: 2024-12-28.

DemandSage. Perplexity AI statistics: Everything you need to know in 2024, 2024a. URL https://www.demandsage.com/perplexity-ai-statistics/. Accessed: 2024-10-19.

DemandSage. Perplexity AI statistics: Everything you need to know in 2024, 2024b. URL https://www.demandsage.com/perplexity-ai-statistics/. Accessed: 2024-10-19.

Avinava Dubey, Zhe Feng, Rahul Kidambi, Aranyak Mehta, and Di Wang. Auctions with llm summaries. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 713–722, 2024.

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 J Eppler and Jeanne Mengis. The concept of information overload: A review of literature from organization science, accounting, marketing, mis, and related disciplines. *The Information society*, 20(5): 325–344, 2004.

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.

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.

Google. Grounding with google search, 2024. URL https://cloud.google.com/vertex-ai/generative-ai/docs/grounding/overview. Accessed: 2024-12-28.

Michael M. Grynbaum and Ryan Mac. The new york times sues OpenAI and Microsoft over content use, 2023. URL https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html. Accessed: 2024-10-19.

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.

MohammadTaghi Hajiaghayi, Sébastien Lahaie, Keivan Rezaei, and Suho Shin. Ad auctions for llms via retrieval augmented generation. *arXiv preprint arXiv:2406.09459*, 2024.

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.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

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.

Jacob Jacoby, Donald E Speller, and Carol A Kohn. Brand choice behavior as a function of information load. *Journal of Marketing Research*, 11(1):63–69, 1974.

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. URL https://aclanthology.org/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.

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.

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.

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.

Yusuf Mehdi. Reinventing search with a new AI-powered microsoft bing and edge – your copilot for the web, 2023. URL https://blogs.microsoft.com/blog/2023/02/07/reinventing-search-with-a-new-ai-powered-microsoft-bing-and-edge-your-copilot-for-the-web/. Accessed: 2024-12-28.

Rajiv Mehta and Trishul Chilimbi. Amazon announces Rufus, a new generative AI-powered conversational shopping experience, 2024. URL https://www.aboutamazon.com/news/retail/amazon-rufus. Accessed: 2024-12-28.

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.

Morphic. An open-source RAG-based AI search engine, 2024. URL https://github.com/miurla/morphic. Accessed: 2024-12-28.

OpenAI. Introducing ChatGPT search, 2024. URL https://openai.com/index/introducing-chatgpt-search/. Accessed: 2024-12-28.

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

Perplexica. An AI-powered search engine, 2024. URL https://github.com/ItzCrazyKns/Perplexica. Accessed: 2024-12-28.

Perplexity. How does perplexity work?, 2024. URL https://www.perplexity.ai/hub/faq/how-does-perplexity-work. Accessed: 2024-10-19.

Elizabeth Reid. How AI is making search more helpful than ever, 2023. URL https://blog.google/products/search/generative-ai-search/. Accessed: 2024-12-28.

Liz Reid. Generative ai in search: Let google do the searching for you, 2024. URL https://blog.google/products/search/generative-ai-google-search-may-2024/. Accessed: 2024-12-28.

Matthew Renze and Erhan Guven. The effect of sampling temperature on problem solving in large language models, 2024. URL https://arxiv.org/abs/2402.05201.

Vaughn Schermerhorn. How amazon continues to improve the customer reviews experience with generative AI, 2023. URL https://www.aboutamazon.com/news/amazon-ai/amazon-improves-customer-reviews-with-generative-ai. Accessed: 2024-12-28.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*,

volume 39. Cambridge University Press Cambridge, 2008.

SearXNG. A free internet metasearch engine, 2021. URL https://github.com/searxng/searxng. Accessed: 2024-12-28.

Stephan Seiler, Song Yao, and Georgios Zervas. Causal inference in word-of-mouth research: Methods and results, 2018.

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.

Nikhil Sharma, Q Vera Liao, and Ziang Xiao. Generative echo chamber? effect of LLM-powered search systems on diverse information seeking. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2024.

Craig S. Smith. What large models cost you – there is no free AI lunch, 2023. URL https://www.forbes.com/sites/craigsmith/2023/09/08/what-large-models-cost-you--there-is-no-free-ai-lunch/. Accessed: 2024-10-30.

Oremus Will. AI chatbots lose money every time you use them. that is a problem., 2023. URL https://www.washingtonpost.com/technology/2023/06/05/chatgpt-hidden-cost-gpu-compute/. Accessed: 2024-10-30.

Haoyi Xiong, Jiang Bian, Yuchen Li, Xuhong Li, Mengnan Du, Shuaiqiang Wang, Dawei Yin, and Sumi Helal. When search engine services meet large language models: visions and challenges. *IEEE Transactions on Services Computing*, 2024.

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

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

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.

Web Appendix

A Prompt Designs

```
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}'.
```

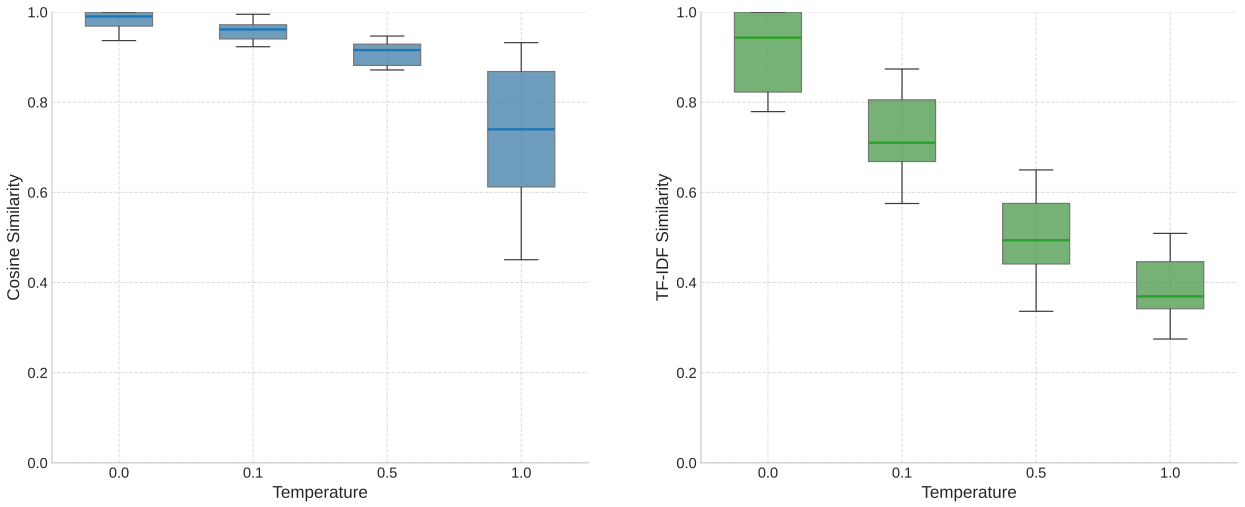
Figure A1: Complete Summarization Prompt Design

```
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 A2: Complete Evaluation Prompt Design

B Temperature Impact Analysis on Summarization and Evaluation

To systematically analyze the impact of temperature settings on LLM outputs in our framework, we conducted two parallel experiments examining both the summarization and evaluation processes. Both experiments followed a consistent experimental design: we selected five distinct datasets from Amazon product reviews, each containing five representative reviews. For each dataset and temperature combination (0.0, 0.1, 0.5, and 1.0), we performed 10 repeated trials to ensure statistical robustness. This experimental framework allows us to systematically evaluate how different temperature settings affect both the consistency of generated summaries (Appendix B.1) and evaluation scores (Appendix B.2), two critical components in our Shapley value calculation pipeline.



(a) Impact of Temperature on Summarization Semantic Consistency

(b) Impact of Temperature on Lexical Diversity

Figure A3: Analysis of Temperature's Impact 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.

B.1 Temperature Impact on Summary Generation Consistency

In this experiment, we evaluated two aspects of summary generation consistency. First, we assessed semantic consistency through embedding-based cosine similarity, which measures the degree to which different generations preserve the same underlying meaning regardless of specific word choices. Second, we examined lexical diversity using TF-IDF similarity measures, which quantify the variation in vocabulary and phrasing across multiple generations of the same summary. These two metrics together provide a comprehensive view of generation stability for different temperature settings: while semantic consistency measures reliability in meaning preservation, lexical diversity reflects the model's flexibility in expression. Figure A3 visualizes the relationship between temperature settings and these metrics, and Table A1 presents the detailed statistical results across all temperature configurations.

The semantic consistency analysis (Figure A3a) reveals several key insights. 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 presence of hardware-level computational variability. As temperature increases, we observe a systematic 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 (Figure A3b). 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)

Temperature	Cosine Similarity		TF-IDF Similarity	
	Mean	Std	Mean	Std
0.0	0.9820	0.0217	0.9102	0.0992
0.1	0.9576	0.0219	0.7244	0.0991
0.5	0.8545	0.1370	0.5013	0.0966
1.0	0.7339	0.1696	0.3845	0.0818

Table A1: Statistical analysis of semantic consistency and lexical diversity across different temperature settings.

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.

These empirical findings present two critical insights for our experimental design. First, even at temperature 0.0, where theoretically deterministic behavior is expected, the model exhibits inherent output variability (mean cosine similarity = 0.9820), indicating that perfect determinism remains unachievable due to hardware-level computational variations. Second, given this inherent variability, the optimal strategy is to balance output consistency with expression richness. Our analysis demonstrates that temperature 0.1 achieves this balance effectively: while it shows only a minimal decrease in semantic consistency from temperature 0.0 (0.9576 vs 0.9820), it enables substantially greater lexical diversity (0.7244 vs 0.9102). This configuration thus maintains reliable meaning preservation while allowing for more natural and varied language expression, making it ideal for our summarization tasks.

B.2 Temperature Impact on Summary Evaluation Consistency

Following the same experimental setup as our summarization analysis, we further investigated the impact of temperature on evaluation consistency. To isolate the evaluation process variance, we fixed the generated summaries and conducted repeated evaluations across different temperature settings, maintaining consistency with our experimental design in §C.2. Figure A4 presents our findings: even with temperature set to 0 and using identical input summaries, we observed persistent variations in evaluation scores. This observation aligns with our previous discussion in §6 regarding hardware-level computational variability and extends its implications to the evaluation process.

The variance analysis reveals that while temperature 0.0 shows the lowest initial variance, it still exhibits noticeable variations in evaluation scores. Increasing the number of evaluation rounds helps reduce this variance, but does not eliminate it completely. Temperature 0.1 demonstrates a similar pattern of variance reduction with increased evaluation rounds, eventually achieving comparable stability to temperature 0.0. This suggests that the slight increase in temperature does not significantly compromise evaluation reliability for our specific task of assigning scores on a 1-10 scale. Therefore, we adopt temperature 0.1 for evaluation to maintain consistency with our summarization process.

In contrast, higher temperatures (0.5 and 1.0) show substantially larger variances that persist even with multiple evaluation rounds, indicating potential reliability issues for evaluation tasks. These findings reinforce our choice of temperature 0.1 for the evaluation process, as it maintains evaluation stability while avoiding the overly rigid scoring patterns observed at temperature 0.0 and the excessive variability at higher temperatures.

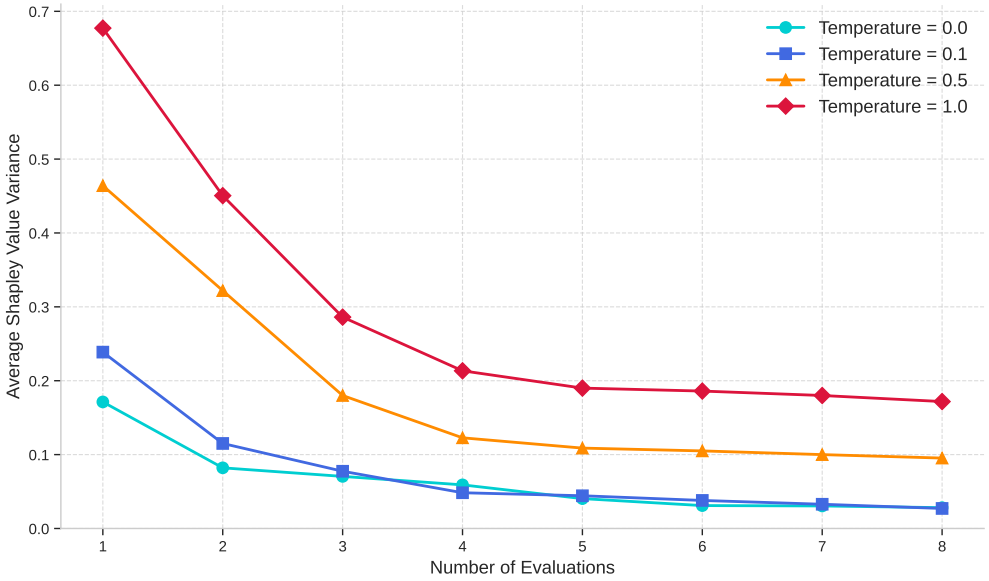


Figure A4: Impact of Temperature on Shapley Value Variance

C Variance Analysis of Shapley Value

Unlike typical Shapley values in traditional game theory applications, document valuation in supervised ML tasks and feature importance (as introduced in §4.1) is generally deterministic. In our LLM setting, however, the Shapley value is inherently random due to the variability in both the summarization and evaluation processes when using LLMs. As discussed, even with the temperature of GPT models set to 0, some degree of randomness remains in the output.

While achieving perfect determinism in LLM outputs is fundamentally challenging due to hardware-level computational variations (as discussed in §6), one can approach near-deterministic behavior through several technical strategies: (1) deploying models locally with controlled environments where random seeds and computational workflows are strictly regulated, including the use of high-precision floating-point operations (float64) to minimize rounding errors; (2) implementing constrained decoding strategies by combining nucleus sampling (top-p = 1.0) with top-k filtering (k = 1) to consistently select the highest probability token; and (3) applying specialized quantization techniques to minimize hardware-induced variations. Note that in the following analysis – the variance induced by both the summarization process $A(S)$ and the evaluation process $v(A(S))$, we omit the notation q in functions for simplicity, as q is fixed and its omission does not introduce ambiguity.

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

$$v(A(S)) = \mu(A(S)) + \varepsilon, \tag{A1}$$

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 assump-

tion and independent white noise, and based on the Shapley formula (5)—which is essentially a linear combination of v—the variance in Shapley can be expressed as the sum of the variances of v. Thus, in the following, we focus on analyzing the variance of $v(A(S))$:

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

Substituting Equation (A1) 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:

$$\text{Var}(v(A(S))) = \text{Var}(\mu(A(S))) + \text{Var}(\varepsilon). \tag{A2}$$

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 $\text{Var}(\varepsilon)$. Essentially, the observed $v(A(S))$ is noisy, and we can reduce this noise by averaging scores from multiple summarization and/or evaluation processes.

C.1 Experimental Analysis and Results

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 datasets from Amazon product reviews, each containing five reviews. For each dataset, we apply the Shapley value calculation across different subsets of reviews to explore how the summarization and evaluation processes contribute to the overall variance. The experimental process involves generating three summarizations for each subset, followed by three evaluation rounds for each summarization. This approach allows us to measure both the variance in summarizations and the evaluation noise.

The empirical variance for each subset is computed in three ways:

- **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.

- **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.

- **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.

The results of this experiment, presented in Table A2, 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%.

C.2 Variance Reduction by Multiple Evaluations in Shapley

The variance decomposition shown in Equation (A2) 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 summary multiple times to obtain an averaged score across evaluations, thereby reducing variance introduced by ε. We now aim to reduce the variance when calculating the Shapley value using the first strategy.

To determine the most cost-effective number of evaluations needed to minimize variance in the Shapley value, we conduct an experiment assessing how increasing the evaluation count impacts Shapley value variance. Using five distinct datasets, each containing five reviews, we fix the summarization to isolate variance attributable to the evaluation process. Shapley value variance is calculated from 10 replications. We conduct the experiments across different numbers of evaluation times ranging from 1 to 8. This approach allows us to analyze how increasing the number of evaluations improves Shapley value consistency and helps identify a balance between computational efficiency and variance reduction.

The results shown in Figure A5 indicate a diminishing variance trend as the evaluation times increase. The most significant variance reduction occurs between one and four evaluations, with the average variance dropping from around 0.25 to 0.05. Beyond four evaluations, the reduction becomes minimal, suggesting that taking an average of four evaluations strikes an effective balance between computational cost and variance reduction.

C.3 Cost-Effectiveness Analysis

In the previous part, we reduce the variance in Shapley values by averaging multiple evaluations with a single summarization. Now, we test variance under configurations of multiple evaluations, as well as summarization. Building on our earlier discussion of variance reduction through increased evaluation counts, we extend a similar analysis to examine the trade-off between computational cost in calculating Shapley values and performance regarding variance. This cost-effectiveness analysis aims to identify an optimal configuration that minimizes variance while keeping computational costs manageable.

Specifically, we conduct experiments 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

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
Overall Average	0.2457	0.1153	0.1304

Table A2: Performance score variance for different subsets. There are a total of $2^5 - 1 = 31$ different subsets.

(also from 1 to 4), we repeat the process six times to calculate the variance in Shapley value. We measure computational cost regarding the total computation time required for each configuration.

The results are displayed in Figure A6. 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.

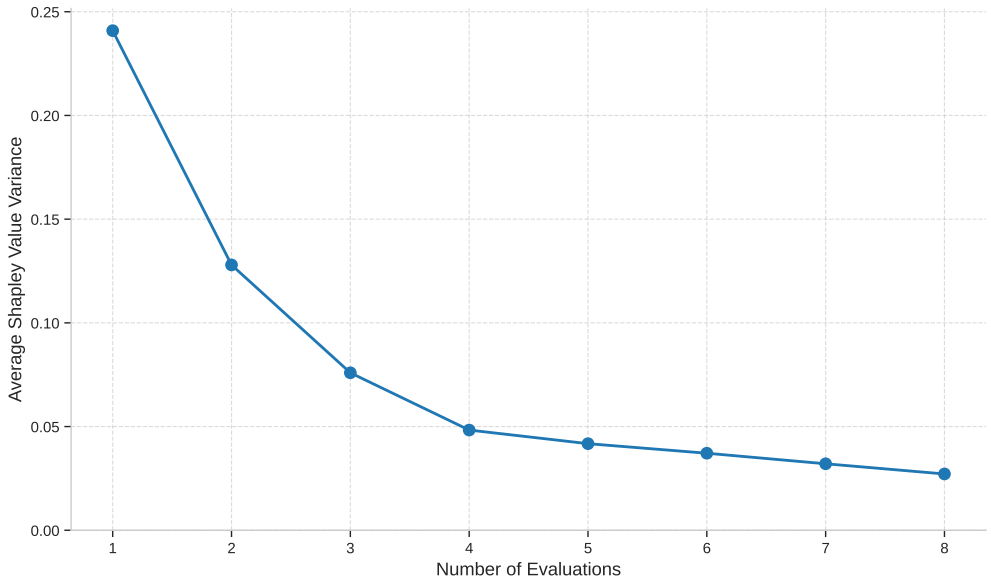


Figure A5: Impact of Evaluation Count on Shapley Value Variance

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.

The computational cost increases proportionally with the number of summarizations and evaluations. For example, computation time rises from approximately 54 seconds for a configuration with 1 summarization and 1 evaluation to about 295 seconds for a configuration with 4 summarizations and 4 evaluations. This proportional increase underscores the importance of balancing variance reduction against computational resources.

Considering practical applications where computational resources and time are constrained, selecting a configuration that optimally balances variance reduction and computation time is essential. Based on our findings, configurations with 1 summarization and 3 or 4 evaluations, or with 2 summarizations and 2 evaluations, achieve substantial variance reduction while maintaining reasonable computation times. For instance, a configuration with 1 summarization and 3 evaluations results in an average variance of approximately 0.0788 with a computation time of about 59 seconds. Increasing to 1 summarization and 4 evaluations further reduces the variance to approximately 0.0616 with a modest increase in computation time to about 64 seconds. Similarly, using 2 summarizations and 2 evaluations yields an average variance of approximately 0.0322 with a computation time of about 114 seconds.

In conclusion, the cost-performance analysis indicates that while increasing the number of summarizations and evaluations can effectively reduce the variance in Shapley value calculations, the benefits taper off beyond certain points. To balance accuracy and computational efficiency, we recommend using a summarization count of 1 with 3 or 4 evaluations, or a summarization count of 2 with 2 evaluations. These configurations sufficiently minimize variance while keeping computation times within practical limits, thereby enhancing the robustness and reliability of the Shapley value estimates without incurring excessive computational costs.

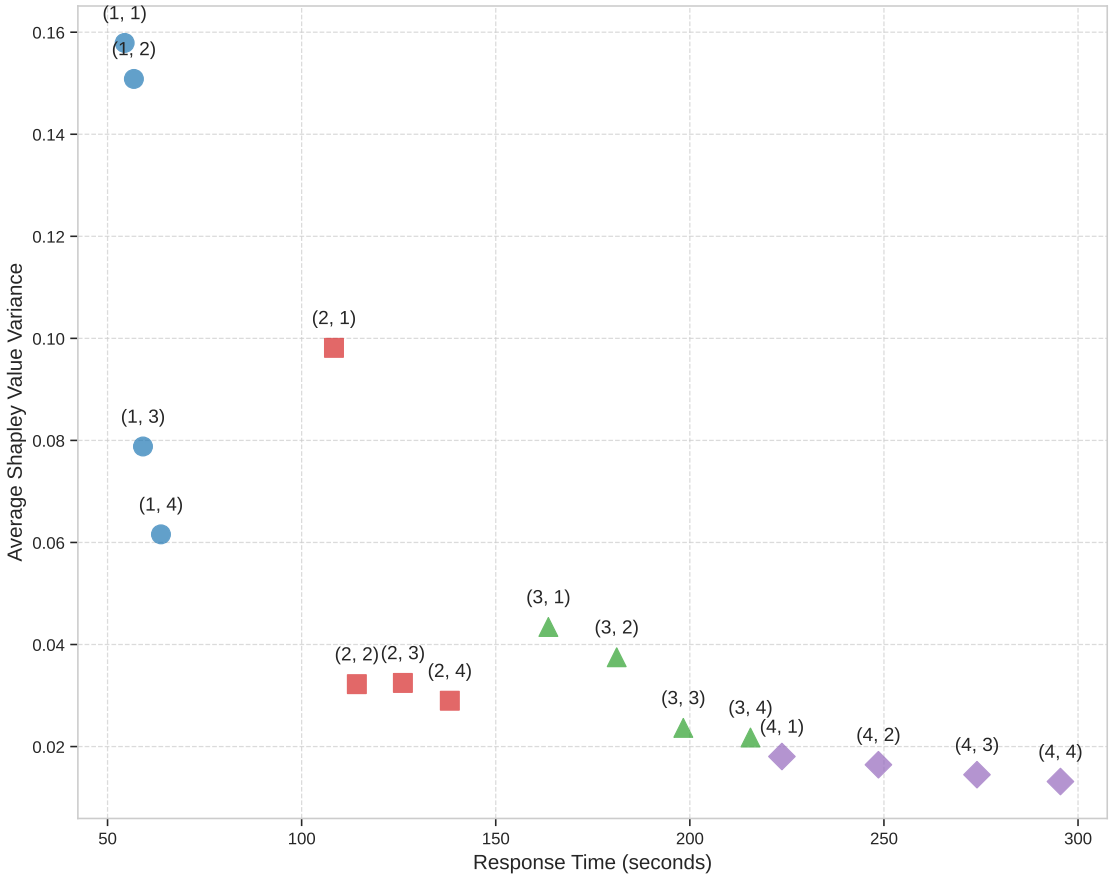


Figure A6: Cost-Effectiveness Analysis: This figure shows the relationship between response time (in seconds) and average Shapley value variance across different summarization and evaluation configurations. Each labeled point represents a configuration, with the first number indicating the number of summarizations and the second the number of evaluations. Point size reflects the number of evaluation steps, while color and shape represent the summarization count.

D Evaluation Model Robustness Analysis

To validate the robustness of our evaluation framework, we conducted an additional set of experiments using Claude-3.5-Sonnet as an alternative evaluation model to compare against the results obtained with GPT-4o. The objective of this analysis was to examine whether the evaluation outcomes remain consistent across different LLMs and thereby assess the generalizability of the proposed framework.

For each summary, we performed four independent evaluations using both GPT-4o(0806) and Claude-3.5-Sonnet(20241022), and the final score was calculated as the average of these four evaluations. To ensure fair comparison, we maintained identical experimental conditions across both models. Both GPT-4o and Claude-3.5-Sonnet were provided with the same prompts and instructed to evaluate summaries on a consistent 0-10 scale. We used a temperature setting of 0.1 for both models to balance output determinism with natural language generation. This averaging process and controlled setup were introduced to mitigate the potential variance caused by the inherent stochasticity of LLM outputs.

To quantify the alignment between the two models' evaluations, we employed two complementary metrics. The mean absolute difference (MAD) metric measures the magnitude of disagreement between the scores assigned by Claude-3.5-Sonnet and GPT-4o, providing a direct measure of score consistency across

models. We calculated MAD for both individual subset scores and final Shapley values to understand the differences at both evaluation and attribution levels. Additionally, we used the Pearson correlation coefficient to assess the consistency in relative rankings assigned by the two models, as this metric captures whether the models agree on the comparative quality of summaries even if their absolute scores differ.

The results revealed a mean MAD of 0.812 and a median MAD of 0.573 for subset scores, suggesting minor, yet expected, differences between Claude-3.5-Sonnet and GPT-4o. These differences are attributable to the intrinsic randomness in LLM outputs, rather than systematic biases in evaluation. Despite these variations, the correlation coefficient for subset scores demonstrated strong alignment, with a mean of 0.788 and a median of 0.837. For Shapley values, the observed differences were notably smaller, with a mean MAD of 0.179 and a median MAD of 0.168. The Pearson correlation coefficient for Shapley values was substantially higher, with a mean of 0.915 and a median of 0.936, indicating a very strong consistency between the two models in evaluating contributions.

These results underscore the robustness of our evaluation framework. Although small differences in absolute scores were observed, these differences are within the range expected from the stochastic nature of LLMs and do not compromise the relative rankings and the computed Shapley values. The high correlation coefficients across both subset scores and Shapley values suggest that our evaluation framework is generalizable and reliable across different state-of-the-art LLMs, such as GPT-4o and Claude. This consistency strengthens the validity of the proposed methodology and ensures its applicability to a broader range of evaluation settings.

E Additional Numerical Results of Cluster Shapley Algorithm

E.1 Computation time of the Clustering Step

We first analyze our adaptive clustering process (i.e., Algorithm 2) 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 1.5310^{-3} seconds, yielding an average time of 1.5310^{-9} seconds 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 §4.2, 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.

E.2 Cluster Shapley Parameter Selection

Figure A7 demonstrates the relationship between approximation accuracy and computational cost across different epsilon values in the Cluster Shapley algorithm. The epsilon parameter, ranging from 0.01 to 1.00, critically influences both clustering granularity and computational efficiency. For reference, a complete calculation without clustering requires evaluating all 255 possible unique subsets. As shown in the figure, the relationship between the number of unique subsets evaluated and Mean Absolute Error (MAE) exhibits clear patterns:

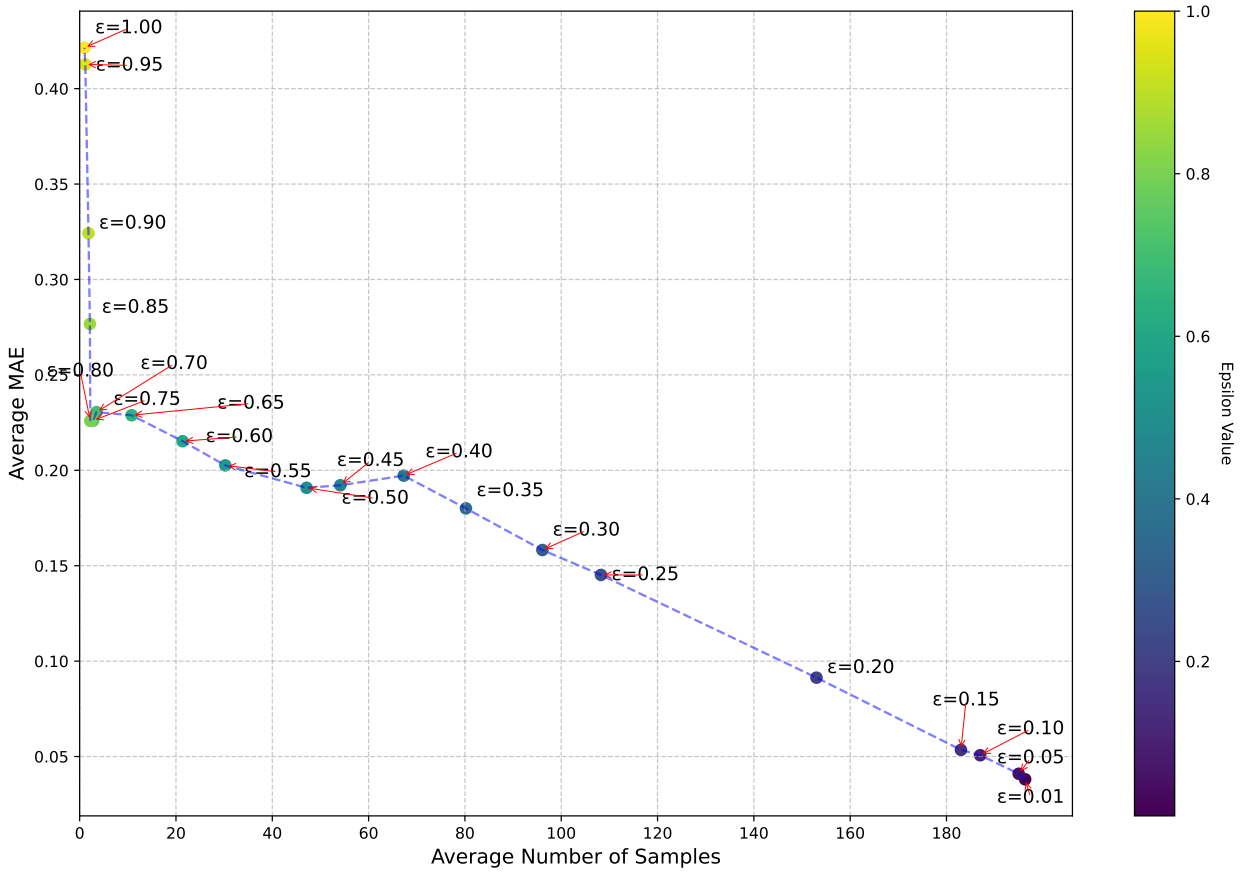


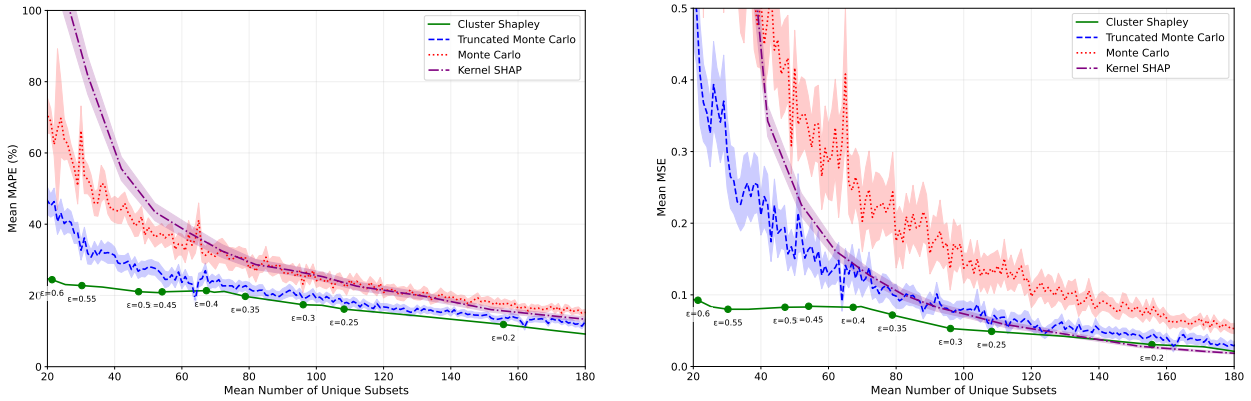
Figure A7: Cost-Performance Analysis of Cluster Shapley Algorithm. The plot shows the relationship between the average number of samples and the average MAE, with different epsilon values annotated. Points are colored according to their epsilon values, revealing the trade-off between computational cost and approximation accuracy. A higher epsilon value corresponds to a less strict similarity threshold in clustering, resulting in fewer clusters and, consequently, fewer required subsets but higher approximation error. Conversely, a lower epsilon value enforces stricter similarity criteria, leading to finer clustering granularity and thus more subsets, while achieving lower MAE.

At lower epsilon values (0.01-0.15), the algorithm maintains high accuracy (MAE ¡ 0.05) but requires evaluating a large number of unique subsets (180-196 out of 255 possible subsets). This is attributed to the formation of more granular clusters due to stricter similarity thresholds. As epsilon increases to the moderate range (0.20-0.50), we observe a gradual trade-off between computational efficiency and accuracy, with the number of required unique subsets reducing significantly while MAE increases moderately. Beyond epsilon values of 0.50, the computational cost continues to decrease sharply, with fewer than 50 unique subsets needed, but at the expense of rapidly deteriorating accuracy, with MAE exceeding 0.20.

This analysis quantifies the fundamental trade-off inherent in our clustering-based approach: epsilon values effectively control the balance between computational efficiency and approximation accuracy. The optimal choice of epsilon depends on specific application requirements, including computational constraints, accuracy demands, and task characteristics. For applications demanding high precision, lower epsilon (0.20-0.30) values are recommended, while scenarios prioritizing computational efficiency might benefit from moderate epsilon values (0.40-0.75), offering a balanced trade-off between accuracy and cost.

E.3 Numerical Results under MAPE and MSE Metrics

For completeness, we present the Cluster Shapley algorithm's performance under MAPE and MSE metrics in Figure A8. The X-axis represents the number of unique subsets or samples used by the algorithms, consistent with Figure 8 in §**??**. The implementation details of all algorithms remain the same, with the only difference being the Y-axis, which reflects the respective performance metrics.



(a) MAPE Performance

(b) MSE Performance

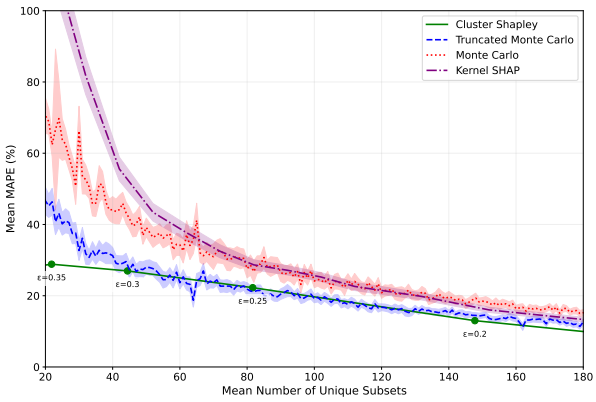
Figure A8: Performance evaluation of the Cluster Shapley algorithm under MAPE and MSE metrics as a complement to the MAE results reported in Figure 8 in §**??**. The X-axis indicates the number of unique subsets or samples used by the algorithms, while the Y-axis represents the respective performance metric.

E.4 Cluster Shapley using the Standard DBSCAN

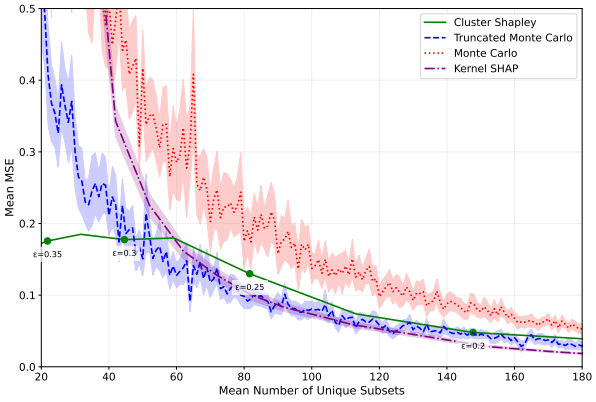
For completeness, we also report the performance of the Cluster Shapley algorithm using the standard DBSCAN rather than our proposed iterative DBSCAN in Figure A9. Unlike the iterative DBSCAN, the standard DBSCAN does not guarantee that all pairs of documents in the same cluster have embeddings with distances strictly smaller than ϵ as discussed in §4.2.

There are two key observations from the comparison. First, when using fewer unique subsets (corresponding to larger epsilon values), standard DBSCAN shows slightly worse performance than our method. This gap gradually narrows as more unique subsets are used. This pattern is particularly relevant to our objective: we aim to reduce computation time while maintaining reasonable accuracy to make Shapley value calculation feasible in real-world applications. The superior performance of our method in the computationally efficient regime (fewer unique subsets) is therefore crucial for practical implementation.

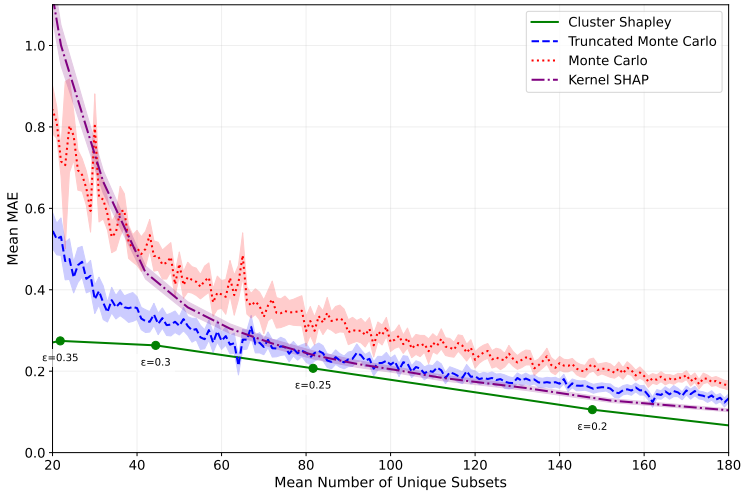
Second, we observe that standard DBSCAN requires different ϵ values than our method to achieve the same number of unique subsets. For instance, at around 20 unique subsets, our method uses $\epsilon = 0.60$ while standard DBSCAN requires $\epsilon = 0.35$; at 80 unique subsets, our method uses $\epsilon = 0.35$ while standard DBSCAN needs $\epsilon = 0.25$; finally, both methods converge to $\epsilon = 0.20$ at around 150 unique subsets. This discrepancy arises from standard DBSCAN's fundamental characteristics - it forms clusters based solely on local density, without enforcing strict intra-cluster distance constraints. Consequently, it requires smaller ϵ values to achieve the same level of clustering granularity as our method. This also explains why the performance gap between the two methods narrows with more unique subsets: as ϵ decreases, both methods naturally enforce stricter clustering criteria, leading to more similar cluster formations and, thus, more comparable performance.



(a) MAPE Performance



(b) MSE Performance



(c) MAE Performance

Figure A9: Performance comparison for Cluster Shapley using the standard DBSCAN. The X-axis represents the number of unique subsets or samples used by the algorithms.