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.

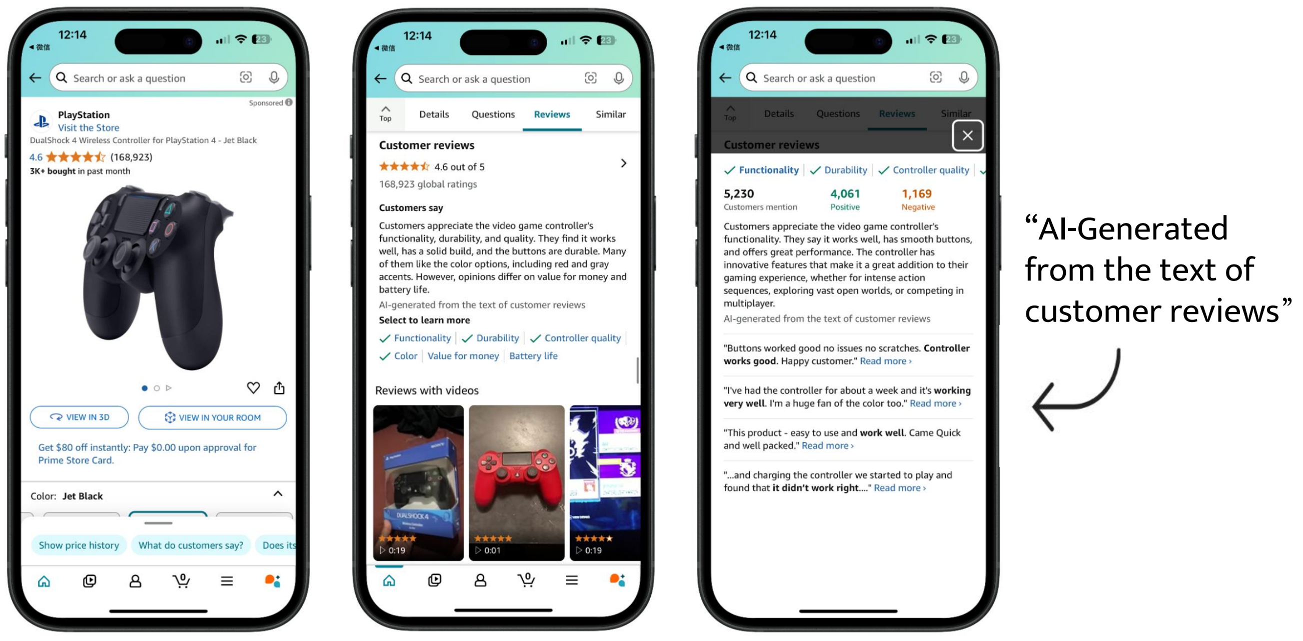
1 Introduction

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responses are grounded in up-to-date, relevant content and do not suffer from limitations such as hallucinations and outdated knowledge that plague off-the-shelf LLMs (Perplexity, 2024).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

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

The main advantage of LLM-based search and summarization tools is that they simplify the search process for users, who no longer have to click on multiple links and aggregate information from individual sources. This can increase customer satisfaction and lead to higher platform usage (Xu et al., 2023). Furthermore, these approaches keep customers on the platform's own website or interface; that is, consumers don't need to leave the search engine and visit other websites to gather information. This, in turn, can lead to increased time spent on the platform, resulting in more eyeballs and potentially higher revenues (Goodwin, 2024). For these reasons, the rise of LLM-based search and summarization has generally been seen as beneficial to both

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



DualShock 4 Wireless Controller for PlayStation 4 - Jet Black

4.6 ★★★★☆ (168,923)

3K+ bought in past month

Get $80 off instantly: Pay $0.00 upon approval for Prime Store Card.

Color: **Jet Black**

Customer reviews

★★★★☆ 4.6 out of 5

168,923 global ratings

Customers say

Customers appreciate the video game controller's functionality, durability, and quality. They find it works well, has a solid build, and the buttons are durable. Many of them like the color options, including red and gray accents. However, opinions differ on value for money and battery life.

AI-generated from the text of customer reviews

Select to learn more

✓ Functionality | ✓ Durability | ✓ Controller quality | ✓ Color | Value for money | Battery life

Customer reviews

✓ **Functionality** | ✓ Durability | ✓ Controller quality | ✓

5,230	4,061	1,169
Customers mention	Positive	Negative

Customers appreciate the video game controller's functionality. They say it works well, has smooth buttons, and offers great performance. The controller has innovative features that make it a great addition to their gaming experience, whether for intense action sequences, exploring vast open worlds, or competing in multiplayer.

AI-generated from the text of customer reviews

"Buttons worked good no issues no scratches. **Controller works good**. Happy customer." Read more ›

"I've had the controller for about a week and it's **working very well**. I'm a huge fan of the color too." Read more ›

"This product - easy to use and **work well**. Came Quick and well packed." Read more ›

"...and charging the controller we started to play and found that **it didn't work right**...." Read more ›

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 an Amazon AI-generated summary review of this product. Users can click "Select to learn more" to focus on specific aspects of interest. The right image shows AI-generated summaries for the selected aspect, displaying the source customer reviews with key information highlighted in bold.

platforms and consumers, at least in the short term.

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

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

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

creators and raises concerns about uncompensated use of their data. Indeed, several major publishers such as *The New York Times* have started to restrict or revoke AI access to their content, citing concerns that their contributions are being undervalued and misused without fair compensation (Grynbaum and Mac, 2023). Together, these changes pose challenges to the sustainability of the digital information economy. If content creators are not fairly compensated for their content, then they will have no incentive to provide content to platforms and search engines. This can lead to a general degradation in the quality of AI-powered summaries over time. Thus, while platforms may see short-term benefits from avoiding attribution or compensation to content creators and websites, this is not a sustainable equilibrium, in the long-term.

As a first step to address this problem, many AI firms have started making licensing deals with large content creators; e.g., recently OpenAI and Murdoch-owned empire of publications like *The Wall Street Journal* and *The New York Post* made a deal to allow OpenAI to use the content from these publications (Robertson, 2024). However, a key challenge in such arrangements is that it is often unclear how to value the body of content from different publishers and arrive at an equitable price for each publisher/contributor. This is due to a few fundamental issues related to value attribution across documents and queries. First, quantifying a document or publisher's contribution to a given summary is a non-trivial task – the value that a specific document adds to a summary (in response to a query) depends not only on the document's own relevance, quality, and reliability, but also on the extent to which the information it contains is unique, i.e., the marginal contribution of the document in comparison to other documents. Second, it is also unclear how to aggregate the a document's contributions across queries. For example, one document/publisher may be very valuable for a niche query while another may be moderately valuable for a very popular query. Thus, we need a principled way to correctly value a document's contribution both *within* and *across* queries.

Further, for a document valuation approach to be broadly applicable and practical, it needs to satisfy three properties: (1) be summarization procedure agnostic, (2) be evaluation process agnostic, and (3) be scalable and cost-effective. The first property is important because LLM-based summarization procedures vary across platforms (e.g., Google uses Gemini-based systems, whereas Perplexity allows the users to choose their base LLM for summarization). Further summarization techniques and LLMs continue to evolve. Thus, it is essential that any framework we develop is not specific to a single LLM or summarization technique. The second property implies that the framework should be agnostic to the procedure used to score the usefulness of the summary because evaluation methods vary across platforms. For example, search engines typically use implicit feedback, such as how long the user spends on the summary. In contrast, question-answering websites or review aggregators often have explicit feedback on the helpfulness of the summaries. Finally, the third property implies that the document valuation procedure should scale to real platforms and not be unreasonably costly in terms of time or money. Together, these properties help ensure that any document valuation procedure is general and scalable, allowing it to be applied across multiple business contexts.[4]

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

In this paper, we present a framework for *equitable document valuation* in the context of LLM-based summaries. Our framework addresses the challenges discussed above and satisfies the three properties noted earlier. Our solution concepts builds on the Shapley value framework from cooperative game theory (Shapley, 1953). First, given a query q and an LLM-based summary based on a specific set of documents, we use the Shapley value framework to distribute the total value generated by the summary (measured by its quality or usefulness) across this set of documents. Shapley values quantify how much each document contributes to the overall summary while accounting for interactions between documents such as redundancy and overlapping information. Formally, Shapley values ensure fair attribution through four properties: Efficiency (the total value is fully distributed across all documents), Symmetry (treat equally contributing documents identically), Null Document (assign zero value to non-contributing documents), and Linearity (additive consistency across multiple evaluations). The first three properties uniquely determine the Shapley value *within* the context of a given query, while the Linearity property allows us to aggregate Shapley values *across* queries. As such, this framework addresses both the challenges discussed above. Further, Shapley value calculation is a meta-algorithm, i.e., the exact procedures used for summarization and evaluating the summaries can be black-boxed in Shapley calculations. Thus, it naturally satisfies the first two desirable properties listed earlier.

However, a major hurdle in applying Shapley values to LLM summaries is its exponential computational complexity – if we have S_q relevant documents for query q, the number of summarizations and evaluations needed to calculate Shapley values is $2^{|S_q|} - 1$. Since most summaries use anywhere from 4–8 documents for summarization and even small platforms typically process millions of queries, the exact Shapley framework is impractical in real settings. To address this problem, many approximation algorithms have been proposed, such as, Monte Carlo (Mann and Shapley, 1960), Truncated Monte Carlo (Ghorbani and Zou, 2019), and Kernel SHAP (Lundberg and Lee, 2017). However, all these algorithms treat each document as independent entities and do not leverage the textual information of documents. As such, they are unlikely to be efficient.

We propose a novel approximation algorithm – Cluster Shapley – that leverages the textual information available to improve efficiency while preserving accuracy. The core idea of our approach is intuitive: documents with similar content should have comparable contributions to the final summary and, therefore, should receive similar Shapley values. Instead of treating documents as independent, our method utilizes LLM-generated embeddings to first identify and cluster similar documents. Then, it treats each cluster as a single meta-document for the purposes of the Shapley calculation. Finally, it distributes Cluster-level Shapley values equally across all the documents within a cluster. This approach significantly reduces the number of document combinations the LLM must summarize and evaluate without seriously compromising accuracy. Another key advantage of Cluster Shapley is its ability to flexibly trade off computational cost and accuracy via a tunable hyperparameter - clustering diameter ϵ. For tasks requiring high precision, a smaller ϵ can be selected, while tasks prioritizing speed can utilize a larger ϵ.

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

To assess performance, we compare our Cluster Shapley algorithm against three widely used approximation 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. When choosing a smaller clustering parameter $\epsilon = 0.20$, a higher accuracy can be achieved (MAE of 0.0913 and MAPE of 11.85%), while the computation time can still be reduced by 40%. 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.

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.

2 Related Literature

Our research relates to and contributes to multiple streams of work, including game theory, LLMs, and marketing. First, our work relates to the growing literature on LLM-based summarization. Recent research on summarization has shown that summaries produced by LLMs like GPT-4 achieve comparable or superior factual accuracy, coherence, and overall quality compared to human annotators in news summarization tasks (Pu et al., 2023). Indeed, text summarization has evolved from traditional extractive methods like TextRank (Mihalcea and Tarau, 2004) to more sophisticated abstractive approaches powered by LLMs. RAG frameworks (Lewis et al., 2020), which combine information retrieval with LLM generation, have shown significant improvements in factual accuracy and information currency (Gao et al., 2023; Jiang et al., 2023). Recent developments include GraphRAG (Edge et al., 2024), which enhances retrieval performance using graph-based representations. While summarization technologies continue to evolve rapidly, our work addresses the fundamental challenge of document valuation that persists across summarization approaches. The document valuation framework we develop in §6.2 is designed to be agnostic to specific summarization techniques, ensuring its applicability even as LLM and RAG technologies advance.

Next, our work builds on the game-theoretic concept of Shapley values (Shapley, 1953), which has gained significant traction in machine learning for quantifying feature importance (Lundberg and Lee, 2017) and data valuation (Jia et al., 2019; Ghorbani and Zou, 2019). 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.

Finally, our work relates to literature on news aggregators in marketing and economics. Intuitively, news aggregators function similarly to AI-based summarization, since users may consume content directly on the aggregator's platform without visiting the publishers' websites. A stream of research has examined whether this substitutive effect reduces traffic to publishers or whether aggregators can also serve as a discovery channel that increases exposure. For instance, Mayzlin and Yoganarasimhan (2012) find that blogs can build an audience by effectively promoting/linking to rival sites under certain conditions. Dellarocas et al. (2013) develop a game-theoretic model and show that content aggregators can both benefit and harm content creators and consumers. Jeon and Nasr (2016) investigate competition among online newspapers in the presence of aggregators. Empirically, both Calzada and Gil (2020) and Athey et al. (2021) find that news aggregators can exert a market-expansion effect, increasing visits to news outlets, especially for smaller publishers. Further, recent work by Amaldoss and Du (2023) explores how publishers might collaborate and compete with news aggregators, and Song and Manchanda (2023) empirically examine the effects of carrying news on user engagement with non-news content on social media. Given that AI-based summarization is functionally similar to content aggregators, our Cluster Shapley approach for valuing source documents can also be used by content aggregators and newspapers/content websites to formalize revenue-sharing arrangements.

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 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, a 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 presents a principled framework for addressing the problem of document valuation in LLM summaries, as introduced in §3. We begin in §4.1 by introducing the Shapley value, a game-theory concept for assigning value to individual documents in cooperative settings and explain its relevance to our context. While theoretically appealing, computing exact Shapley values is not practically applicable in most settings (including ours). To address this, §4.2 introduces an efficient approximation algorithm—Cluster Shapley—which leverages LLM-generated similarity embeddings to reduce the computational burden. Finally, in §4.3, we analyze the theoretical properties of this algorithm and provide guarantees on approximation error and computation complexity.

4.1 Shapley Value

We now introduce the Shapley value 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 now adapt and present the four properties of $\phi_i(q)$ within the context of our LLM-summarization problem to guarantee **equitable document valuation**:

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

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

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)).

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

$$\phi_i(q) = \phi_j(q), \quad \text{if} \quad v(q, A(q, S \cup \{i\})) = v(q, A(q, S \cup \{j\})) \;\; \forall \;\; S \subseteq D \setminus \{i, j\}. \tag{3}$$

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

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

4. **Linearity.** The values of document i under two separate queries q_1 and q_2, sum up to its value when evaluated using a performance score function that combines the individual performance score functions. Formally:

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

   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.

   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, $v(q_1, A(q_1, S)) + v(q_2, A(q_2, S)) + \cdots + v(q_n, A(q_n, S))$, where q is drawn from the distribution of queries $g(q)$. This approach is natural because queries typically arrive sequentially and are processed independently. The linearity property ensures that under our 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 important practical implications in our LLM setting since it provides a natural mapping between query-level revenues generated from a platform's consumers/users and batch-level licensing/subscription fees that the platform needs to pay to content providers. For instance, 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 revenues 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 revenues, 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 determine the document value function $\phi(q)$, Shapley value (Shapley, 1953); and no additional conditions are necessary.[5] This is a foundational result in cooperative game theory. We refer interested readers to Shapley's seminal paper for a formal proof, and henceforth only focus on how these properties apply to our context and their importance in our setting.

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.

Recall that for $i \notin S_q$, the Shapley value of i is $\phi_i(q) = 0$ because only documents in S_q are used for summarization. Documents outside S_q have no 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 as:

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

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

However, many of these evaluation score calculations are redundant since 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.

4.2 Approximating Shapley Value: Cluster Shapley Approach

Computational Challenge: While Shapley valuation is a theoretically appealing construct, evaluating Shapley values for source attribution presents a significant computational challenge even in moderate-sized settings. The computational cost associated with Shapley calculation exhibits exponential complexity – if we have S_q relevant documents for query q, the number of both summarization and evaluations scales needed to calculate Shapley values scale as $2^{|S_q|} - 1$, where $|S_q|$ is the number of relevant documents. This rapid growth in the number of calculations makes exact Shapley computation infeasible for large datasets, as the number of summarizations and evaluations quickly becomes overwhelming. Essentially, for each combination of documents, we need the LLM to generate a new summary and then perform an evaluation of that summary. While parallelization and batch processing can reduce latency, the overall computational burden remains substantial. As discussed in a later section §6.5, even for a query with eight relevant 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) are infeasible and we therefore need efficient approximation algorithms.

Researchers have proposed a number of algorithms designed to address the computational challenge associated with Shapley calculations. These approaches typically adopt a variety of sampling techniques to reduce the computational cost associated with Shapley calculation. One widely used method is Monte Carlo algorithm (Mann and Shapley, 1960), which estimates Shapley values by randomly sampling permutations and computing marginal contributions across these samples. While this approach reduces computational 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.

Key Idea: 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. Earlier embedding methods, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), are based on shallow neural networks and co-occurrence statistics, learning word-level embeddings by predicting surrounding context words or factorizing word co-occurrence matrices. These embeddings typically represent each word with a fixed vector, independent of context. In contrast, modern LLM-based embeddings, such as those produced by OpenAI's latest text-embedding models, are generated using Transformer-based architectures and are pretrained on massive text corpora via next-token prediction objectives. These newer embeddings are contextualized—meaning the vector for a word or sentence depends on its surrounding context—and are typically high-dimensional (e.g., 3072 dimensions in OpenAI's `text-embedding-3-large` model). Unlike generative LLMs designed for tasks like chat or text generation, embedding models are optimized to produce semantically meaningful representations suitable for a wide range of downstream tasks. LLM-based embedding vectors have been successfully applied to a wide variety of discriminative tasks, including text classification, document retrieval, sentiment analysis, and predicting the attractiveness of news headlines (Patil et al., 2023; Ye et al., 2025). In our setting, we leverage these embeddings to cluster similar documents before computing Shapley values, allowing us to reduce redundant calculations.

Our Approach: We outline our proposed Cluster Shapley Algorithm in Algorithm 1. Cluster Shapley begins with a preprocessing step, where for a given query q, we first determine the set of relevant documents $S_q \subseteq D$. This retrieval step ensures that only contextually relevant documents are considered; e.g., if the query pertains to political news, unrelated sports articles will be excluded from summarization. In §6.2, 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/Alpaca.

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

$$\text{cosine\_similarity}(e_i, e_j) = \frac{e_i \cdot e_j}{\|e_i\| \|e_j\|}. \tag{8}$$

This metric measures the cosine of the angle between two vectors in an inner product space, capturing how similar their directional components are. Higher cosine similarity values indicate greater textual similarity, meaning the embeddings of semantically similar documents are more aligned.

To facilitate clustering, we define a corresponding distance measure, $d(e_i, e_j)$, which is bounded within the range $[0, 1]$, given by:

$$d(e_i, e_j) = 1 - \text{cosine\_similarity}(e_i, e_j) = 1 - \frac{e_i \cdot e_j}{\|e_i\| \|e_j\|}. \tag{9}$$

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 Algorithm

```
Step 0: Inputs and preprocessing.
    Given a query $q$, retrieve the set of relevant documents $S_q$.
    For each document $i \in S_q$, obtain its embedding vector $e_i$.
    Set the clustering diameter (similarity threshold) $\epsilon > 0$.
Step 1: Document clustering with the distance constraint.
    Cluster the documents in $S_q$ into clusters $\{G_1, G_2, \ldots, G_m\}$ such that for any $i, j \in G_k,\ k \in [m] := \{1, 2, \ldots, m\}$, we have $d(e_i, e_j) \leq \epsilon$. Each cluster $G_k$ is a group of similar documents.
Step 2: Cluster-level Shapley value computation.
    Define a cluster-level value function $v_{\mathcal{G}}(T) := v\left(\bigcup_{G_k \in T} G_k\right)$ for $T \subseteq \{G_1, \ldots, G_m\}$.
    Compute the Shapley value $\hat{\phi}_{G_k}$ for each cluster $G_k$ based on $v_{\mathcal{G}}$.
Step 3: Document-level value allocation.
    For each document $i \in G_k,\ \forall k \in [m]$, assign the approximated Shapley value:
        $$\hat{\phi}_i = \frac{\hat{\phi}_{G_k}}{|G_k|}. \quad (10)$$
```

4.2.1 Step 1: Document Clustering with the Distance Constraint

We now discuss the document clustering step (Step 1). First, note that this step is agnostic to a specific clustering algorithm. The goal is to partition the documents into m non-overlapping clusters, i.e., $D = \bigcup_{k=1}^{m} G_i$ with $G_i \cap G_j = \emptyset$ for $i \neq j$. 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, standard clustering algorithms do not always enforce uniform intra-cluster proximity—that is, they may allow pairs of documents within the same cluster to be arbitrarily far apart, particularly in the presence of high-dimensional noise or uneven density. This flexibility, while useful in noisy settings, is problematic for our purpose: the theoretical performance of the Cluster Shapley algorithm relies on a Lipschitz continuity assumption (Assumption 1), which bounds the Shapley value differences between documents by their embedding distance. Without bounding the pairwise distance within each cluster, we cannot guarantee that documents in the same cluster will have similar Shapley values—violating the condition needed to ensure our approximation guarantees hold. Therefore, unlike the standard clustering algorithm, we impose a distance constraint that documents within the same cluster should be strictly close to each other, with a distance less than ϵ; that is, if $i, j \in G_k$, then $d(e_i, e_j) \leq \epsilon$. We also find that this constraint can improve the empirical performance of the Cluster Shapley algorithm; see more details in Web Appendix §G.3. Intuitively, a smaller ϵ results in more clusters (larger m), leading to a more accurate

Shapley estimation at the cost of increased computation. In the extreme case, we can set ϵ to be the smallest distance between any pair of documents, $\epsilon < \min_{i,j \in S_q, i \neq j} d(e_i, e_j)$, which yields clusters where each cluster contains only one document. In this case, our proposed algorithm reduces to the exact Shapley calculation. Therefore, tuning ϵ appropriately is essential, as it balances the trade-off between computational efficiency and the approximation error induced by clustering. We formalize this statement in §4.3.

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. 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, the 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 documents i, j, and k: if $d(e_i, e_j) \leq \epsilon'$ and $d(e_j, e_k) \leq \epsilon'$, DBSCAN will assign all three points to the same cluster through density-reachability, even if $d(e_i, e_k) > \epsilon'$. This transitive clustering property, while effective for many applications, can result in clusters where some document pairs exceed the 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 document embeddings and a predefined distance function d, and input this matrix into the DBSCAN algorithm. We set the minimum number of points per cluster to 1 (i.e., a DBSCAN hyperparameter $\texttt{MinPts} = 1$) to ensure no document is excluded as noise. However, standard DBSCAN does not guarantee that all documents within a cluster are within ϵ (the distance constraint we want to satisfy), nor does it ensure sufficient separation between clusters. To address this, we distinguish between two thresholds: the clustering diameter ϵ, which defines our desired upper bound on intra-cluster distances, and the DBSCAN neighborhood radius ϵ', which governs local density during clustering. Initially, we set $\epsilon' \leftarrow \epsilon$, run DBSCAN, and then check whether all document pairs within each cluster satisfy the global diameter constraint $d(e_i, e_j) \leq \epsilon$. If any cluster violates this condition, we iteratively tighten the local neighborhood by reducing ϵ' by a factor of $\alpha = 0.95$, i.e., $\epsilon' \leftarrow 0.95\epsilon'$, and rerun DBSCAN until all clusters satisfy the diameter constraint.

Lastly, we discuss the choice of the clustering diameter ϵ, a key hyperparameter that governs both the approximation error and the computational cost. Smaller values of ϵ lead to more accurate approximations of the Shapley value but incur higher computational cost, as formalized in §4.3. In practice, one can perform

Algorithm 2 Iterative Distance-Constrained DBSCAN

```
Input: Distance matrix $M$ with $M_{ij} = d(e_i, e_j)$. Hyperparameters: clustering diameter $\epsilon$ defined in
Cluster Shapley Algorithm, neighborhood radius $\epsilon'$, MinPts $= 1$, scaling factor $\alpha = 0.95$
Initialize the neighborhood radius $\epsilon' \leftarrow \epsilon$          ▷ Start with the original $\epsilon$
while true do          ▷ Iterate until all clusters satisfy the distance constraint
    Run the standard DBSCAN:
    • For each document $i$: find $\epsilon'$-neighborhood $N_{\epsilon'}(i) = \{j : M_{ij} \leq \epsilon'\}$.
    • If $|N_{\epsilon'}(i)| \geq$ MinPts, mark $i$ as a core point.
    • Connect core points that are within $\epsilon'$ 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 distance constraint, 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.
```

standard hyperparameter tuning: use a separate dataset – such as one from historical logs – to select the value of ϵ that best balances approximation accuracy and computational efficiency. This selected value can then be fixed for future use. Later in §7.2, we empirically show how ϵ governs the trade-offs between computational costs and approximation error in our application setting.

4.2.2 Steps 2 and 3: Cluster-level Shapley Value Computation and Document Value Allocation

In Step 2 of Cluster Shapley (Algorithm 1), we consolidate the documents within each cluster by concatenating them into a single *meta-document*, which serves as a representative unit for that cluster. Instead of computing Shapley values for individual documents, we calculate the Shapley value $\hat{\phi}_{G_k}$ for each cluster $k \in [m]$, significantly reducing the computational complexity. After forming these 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 techniques. In case we have a very large dataset or if there are a large number of clusters, approximation methods such as Monte Carlo sampling can be employed to further reduce computational costs while maintaining reasonable accuracy. We formally discuss this additional approximation in §4.3.3.

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

4.3 Theoretical Analysis of Cluster Shapley Algorithm

We now develop theoretical guarantees for our proposed Cluster Shapley algorithm. We begin by stating a mild assumption that semantically similar documents—i.e., those close in the embedding space—have similar contributions to the summarization output. This Lipschitz continuity assumption underpins the accuracy of our clustering-based approximation by ensuring that documents grouped into the same cluster have bounded differences in their Shapley values.

Under this assumption, we first analyze the approximation error introduced by clustering in §4.3.1. Specifically, we prove that the error in estimating the Shapley value of any document is at most $L\epsilon$, where ϵ is the clustering diameter chosen in the algorithm. This result highlights a trade-off: smaller ϵ leads to more accurate Shapley estimates, at the cost of increased computation.

Second, we characterize the computational complexity of the algorithm in §4.3.2. By reducing the number of elements from n documents to m clusters, the computation cost of Shapley value computation drops from $O(2^n)$ to $O(n^2 + 2^m)$, offering substantial gains in efficiency when m is small.

Third, for the scenario that m is still moderately large (so that 2^m complexity is burdensome), we extend the analysis to a Monte Carlo version of Cluster Shapley in §4.3.3, where cluster-level Shapley values are estimated using random permutations/samples. We derive high-probability error bounds and show that the computation complexity scales inversely with the square of the desired error level. This variant enables even greater scalability when the number of clusters m becomes moderately large.

Taken together, our analysis shows that Cluster Shapley is a principled, flexible algorithm that offers provable accuracy-efficiency trade-offs.

4.3.1 Approximation Error of Cluster Shapley

We first introduce the following mild assumption.

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

$$\Big| \big(v(S \cup \{i\}) - v(S)\big) - \big(v(S \cup \{j\}) - v(S)\big) \Big| \leq L\, d(e_i, e_j). \tag{11}$$

Under Assumption 1, if two documents lie in the same cluster ($d(e_i, e_j) \leq \epsilon$), their marginal contributions in any coalition differ by at most $L\epsilon$. Taking a weighted average over all coalitions (as in the Shapley formula), then implies that their Shapley values differ by at most $L\epsilon$ as well. Thus, documents grouped together by the clustering step are approximately symmetric in terms of their contribution. In particular, if $d(e_i, e_j)$ is small, i and j have nearly interchangeable effects on any summary. This also justifies the Cluster Shapley algorithm's strategy of treating cluster members as equal for value allocation in the last step. We now formalize the approximation error, accuracy, and complexity of our approach.

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

Proposition 1 (Convergence and Approximation Error Bound)**.** Under Assumption 1, the approximated Shapley values $\hat{\phi}$ in Equation (10), output by the Cluster Shapley algorithm, converge to the exact Shapley values ϕ as the clustering diameter ϵ approaches zero (i.e., as each cluster becomes an identical-document singleton). In particular, for any document i, the approximation error is bounded by:

$$|\hat{\phi}_i - \phi_i| \leq L\,\epsilon. \tag{12}$$

The proof can be found in Web Appendix §E.1. This proposition provides an approximation error bound $|\hat{\phi}_i - \phi_i| \leq L\epsilon$ for each document's approximated Shapley value as a function of the clustering diameter ϵ. Intuitively, this implies that the MAE of the Cluster Shapley value of a document i ($\hat{\phi}_i$ is upper bounded by the maximum distance between any two documents in the cluster times a constant L. We now further interpret this result and consider the following special case.

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

The proof can be found in Web Appendix §E.2. This corollary confirms that the algorithm is exact for clusters of truly similar documents. In realistic settings, documents in a cluster may not be perfectly identical in contribution, but as long as the within-cluster heterogeneity is small, the approximation will be accurate.

4.3.2 Computational Complexity Analysis

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

- **Document clustering complexity:** The algorithm must compute embeddings for n documents and then cluster them based on pairwise distances. Generating embeddings (using a pre-trained LLM embedding model) takes $O(n)$ operations. Computing the pairwise distance matrix naively costs $O(n^2)$ time (since there are $\binom{n}{2}$ pairs). The clustering itself using DBSCAN will typically require examining each point's neighbors; DBSCAN in worst-case can be $O(n^2)$, though it can approach $O(n \log n)$ on average (Ester et al., 1996). In summary, the clustering step is polynomial in n. We can reasonably approximate it as $O(n^2)$ in the worst case, which is manageable for moderate n. This phase is far more efficient than an exhaustive Shapley computation (which is exponential in n).

- **Cluster-level Shapley value computation complexity:** Once m clusters are formed, we must compute the Shapley values for these m meta-players. If we perform the exact Shapley computation at the cluster level, there are 2^m subsets in total (including empty and full), but since $v(\emptyset) = 0$ is trivial, one often writes $2^m - 1$ evaluations. Each such evaluation requires generating a summary from the union of documents in those clusters and scoring it. Thus, this step is exponential in m. In Big-O notation, the worst-case runtime for computing cluster Shapley exactly is $O(2^m)$.

Combining the two phases, the overall worst-case time complexity is $O(n^2) + O(2^m)$. It is important to note that the exponential complexity in m is unavoidable if using the exact calculation in Step 2 of the Cluster Shapley algorithm. Cluster Shapley does not magically circumvent the combinatorial explosion; rather, it reduces the problem size from n to m by leveraging redundancy in the document. In scenarios where n is large but the effective number of independent information sources is small (many documents are repeats or very similar), this yields a drastic speed-up. In the best case, if all n documents were near-duplicates of a few types, m would be small (say $m = O(\log n)$ or even $O(1)$), and then the overall algorithm runs in polynomial time $O(n^2 + 2^{O(\log n)}) = O(n^2 + n^c)$ (or even $O(n^2)$), which is efficient. In the worst case where no two documents are alike ($m = n$), the complexity $O(n^2 + 2^n)$ reverts to that of exact Shapley (plus the n^2 overhead).

4.3.3 Moderately Large m Scenario

Finally, we note that if m is still moderately large (so that 2^m summarizations and evaluations is burdensome), one can combine Cluster Shapley with other Shapley approximation methods. For example, instead of exact evaluation on all 2^m cluster subsets in Step 2, one could run a Monte Carlo sampling procedure or use a truncated permutation method to estimate the cluster Shapley values in polynomial time. This would further reduce computation at the cost of a small sampling error. We formally outline it in Algorithm 3 and give the following proposition about the approximation error and computation complexity.

Proposition 2 (Monte Carlo Cluster Shapley Error and Complexity)**.** Suppose Assumption 1 holds and the marginal contribution of any cluster is bounded by $V_{\max}$, the approximated Shapley values $\hat{\phi}$, the output by Algorithm 3, converge to the exact Shapley values ϕ as the clustering diameter $\epsilon \to 0$ and the number of permutations $N \to \infty$. In particular, for any document $i \in G_k$ and any cluster $k \in [m]$, with probability at least $1 - \eta$, the approximation error is bounded by:

$$|\tilde{\phi}_i - \phi_i| \le L\epsilon + \frac{V_{\max}}{|G_k|}\sqrt{\frac{\log(2/\eta)}{2N}}. \tag{13}$$

Additionally, to guarantee that the total approximation error per document does not exceed a threshold $\varepsilon_{\text{total}}$, it suffices to choose:

$$N \ge \frac{V_{\max}^2}{2|G_k|^2(\varepsilon_{\text{total}} - L\epsilon)^2} \log\left(\frac{2}{\eta}\right). \tag{14}$$

And the total computational complexity is $O(n^2 + Nm)$.

The proof is shown in Web Appendix §E.3. As shown in Proposition 2, when using Monte Carlo to

Algorithm 3 Cluster Shapley Algorithm with Monte Carlo Approximation

```
Step 0: Inputs and preprocessing.
    Given a query $q$, retrieve the set of relevant documents $S_q$.
    For each document $i \in S_q$, obtain its embedding vector $e_i$.
    Set the clustering diameter (similarity threshold) $\epsilon > 0$.
    Set the number of permutation samples $N$.
Step 1: Document clustering.
    Cluster the documents in $S_q$ into groups $\{G_1, G_2, \ldots, G_m\}$ such that for any $i, j \in G_k,\ k \in [m] := \{1, 2, \ldots, m\}$, we have $d(e_i, e_j) \leq \epsilon$. Each cluster $G_k$ is a group of similar documents.
Step 2: Cluster-level Shapley value approximation via Monte Carlo.
    Define a cluster-level value function $v_{\mathcal{G}}(T) := v\left(\bigcup_{G_k \in T} G_k\right)$ for $T \subseteq \{G_1, \ldots, G_m\}$.
    Approximate the Shapley value $\tilde{\phi}_{G_k}$ for each cluster $G_k$ based on $v_{\mathcal{G}}$ and $N$ i.i.d. sampled random permutations:
        $$\tilde{\phi}_{G_k} = \frac{1}{N} \sum_{n=1}^{N} \Big[ v_{\mathcal{G}}\Big( \bigcup_{G_j \in T_k^{(n)}} G_j \cup G_k \Big) - v_{\mathcal{G}}\Big( \bigcup_{G_j \in T_k^{(n)}} G_j \Big) \Big],$$
    where $T_k^{(n)}$ is the set of clusters that appear before $G_k$ in the $n$-th random permutation of all clusters.
Step 3: Document-level value allocation.
    For each document $i \in G_k,\ \forall k \in [m]$, assign the approximated Shapley value:
        $$\tilde{\phi}_i = \frac{\tilde{\phi}_{G_k}}{|G_k|}.$$
```

approximate the Shapley value of Clusters in step 2, it introduces an additional error term $\frac{V_{\max}}{|G_k|}\sqrt{\frac{\log(2/\eta)}{2N}}$ while reducing the computation cost from $O(n^2 + 2^m)$ to $O(n^2 + Nm)$. To maintain the same order of total approximation error, it suffices to have $N = O(\epsilon^{-2})$. This implies that the overall computation is reduced from exponential in the number of clusters m to polynomial, while still preserving approximation accuracy.

5 Application Setting: Amazon Review Dataset

We now present an application of our algorithm to a real setting. We use the publicly available Amazon Product Reviews dataset as the empirical context to demonstrate the performance of our document valuation approach. This dataset was collected by Hou et al. (2024) and has been 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, and 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.

To that end, many e-commerce platforms (including Amazon) have started adopting LLMs to retrieve and summarize the most relevant information for a consumer's specific query from the available set of reviews/user-generated content (see Figure 1). Customers can either see a summary from all reviews or query the system for a specific piece of information (e.g., ease of return) through Amazon's "Rufus" AI chatbot. For our analysis, we focus on query-based summaries, though our framework is quite general and can also be applied to the general summarization settings.

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

To compute the Shapley value of each individual review (within the context of a given product), we need to first specify the distribution of queries, $g(q)$, that consumers use when requesting summaries for this product. While this query distribution is not publicly available, for each product, we observe the set of popular attributes frequently mentioned by customers to construct the queries. 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 common aspects of the product that users are concerned about. This information allows us to craft a proxy distribution of user queries for each product that mimics the real distribution of queries. Specifically, for each product, we design two queries based on the top two attributes most frequently mentioned in its reviews, as shown in the last column of Table 1. Each query is assigned equal probability, contributing 50% to the valuation.[6]

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?
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?

Continued on next page

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

No.	Product	Number of Reviews	Designed Query
22	Dental Tool	1,343	1. How's the quality of the dental tool? 2. Does the dental tool do its job effectively?
23	Crystal Crowns	1,374	1. What's the quality of the crystal crowns? 2. How beautiful are the crystal crowns?
24	Blemish Formula	449	1. What's the quality of the blemish formula? 2. How effective is the blemish formula?

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

6 Implementation Details of Shapley Value

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 §6.1, we first introduce the RAG architecture, which describes how pre-trained LLMs can be augmented for search tasks with domain-relevant documents/articles. Next, in §6.2 and §6.3, we detail our summarization $A(\cdot)$ and evaluation $v(\cdot)$, respectively. Last, we discuss the implementation details for the exact Shapley and Cluster Shapley in §6.5.

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

- **Retriever (R):** When a search query comes in, the retriever locates and retrieves relevant information by identifying a set of documents that are relevant to the search query. Essentially, given a query q and a set of documents D, the retriever's goal is to identify a subset $S_q \subseteq D$ that is most relevant to the query.

- **Augmentation (A):** In this phase, the retrieved documents (S_q) are integrated with the original input (user query, q) to provide additional context for the generative model. This augmentation ensures that the response from the generative model is grounded in retrieved reliable information, enhancing both its accuracy and relevance.
- **Generator (G):** The generator, typically an LLM such as GPT or Claude, synthesizes the user's query (q) and the retrieved information (S_q) to produce a coherent response. By incorporating the retrieved documents, the LLM can generate outputs that go beyond its pre-trained knowledge, delivering more comprehensive and contextually appropriate responses.

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).

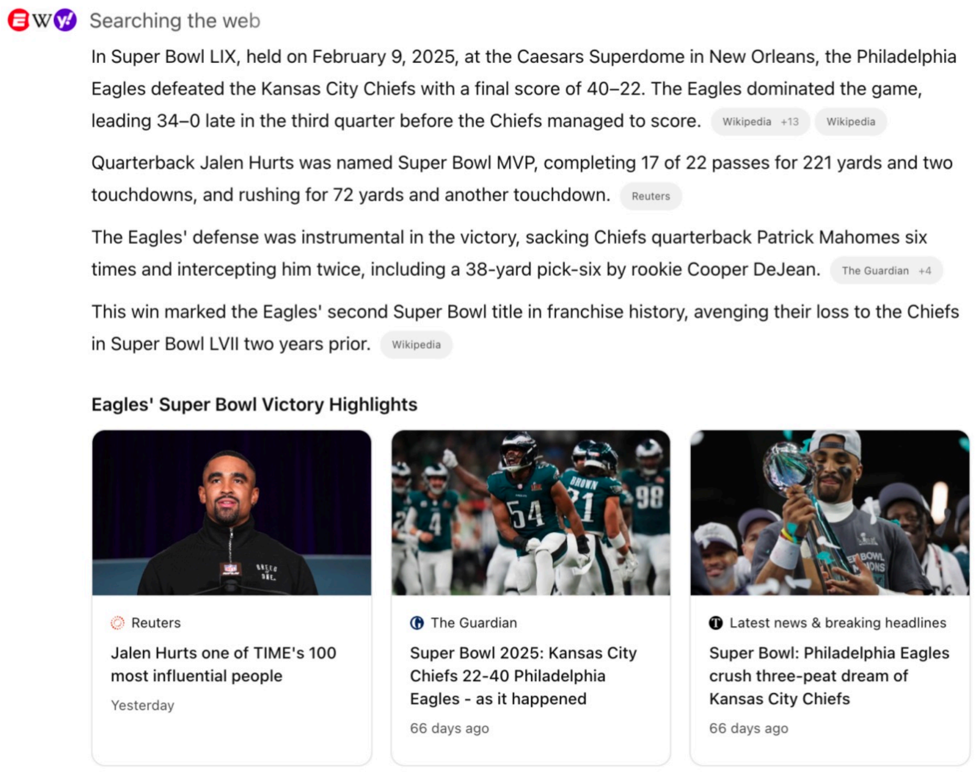
To illustrate the impact of RAG integration on LLMs, we provide an example with a query: "Who won the Super Bowl this year?" in Figure 2. In this case, the ChatGPT application (powered by GPT-4o) leverages web browsing to retrieve up-to-date information that is not present in the model's training data (GPT-4o has a knowledge cutoff of September 30, 2023). This example highlights how RAG enables LLMs to access and incorporate real-time information beyond their static training corpus.

6.2 Summarization of Relevant Amazon Reviews via RAG

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 from Amazon reviews and summarization tool $A(q, S_q)$ for any given query q. Figure 3 shows the overview of the four-step procedure.

- **Step 0: Generate Text Embeddings**

  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,


Searching the web

In Super Bowl LIX, held on February 9, 2025, at the Caesars Superdome in New Orleans, the Philadelphia Eagles defeated the Kansas City Chiefs with a final score of 40–22. The Eagles dominated the game, leading 34–0 late in the third quarter before the Chiefs managed to score. Wikipedia +13 Wikipedia

Quarterback Jalen Hurts was named Super Bowl MVP, completing 17 of 22 passes for 221 yards and two touchdowns, and rushing for 72 yards and another touchdown. Reuters

The Eagles' defense was instrumental in the victory, sacking Chiefs quarterback Patrick Mahomes six times and intercepting him twice, including a 38-yard pick-six by rookie Cooper DeJean. The Guardian +4

This win marked the Eagles' second Super Bowl title in franchise history, avenging their loss to the Chiefs in Super Bowl LVII two years prior. Wikipedia

Eagles' Super Bowl Victory Highlights

Reuters
Jalen Hurts one of TIME's 100 most influential people
Yesterday

The Guardian
Super Bowl 2025: Kansas City Chiefs 22-40 Philadelphia Eagles - as it happened
66 days ago

Latest news & breaking headlines
Super Bowl: Philadelphia Eagles crush three-peat dream of Kansas City Chiefs
66 days ago

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

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

  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 3, the user's query is, "I would like to know more details about the quality of the wireless controller." Here, the key semantic information is, "quality of the 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

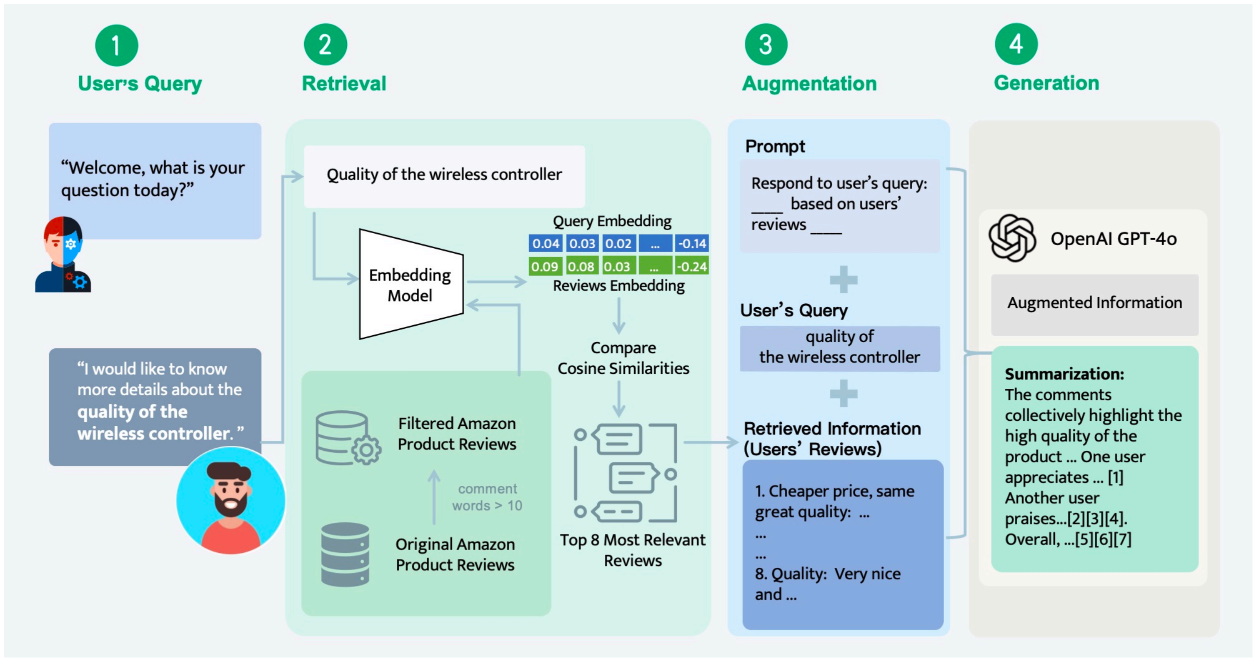

1
User's Query
2
Retrieval
3
Augmentation
4
Generation
"Welcome, what is your question today?"
"I would like to know more details about the quality of the wireless controller."
Quality of the wireless controller
Embedding Model
Query Embedding
0.04 0.03 0.02 ... -0.14
0.09 0.08 0.03 ... -0.24
Reviews Embedding
Compare Cosine Similarities
Filtered Amazon Product Reviews
comment words > 10
Original Amazon Product Reviews
Top 8 Most Relevant Reviews
Prompt
Respond to user's query: ____ based on users' reviews ____
User's Query
quality of the wireless controller
Retrieved Information (Users' Reviews)
1. Cheaper price, same great quality: ...
...
...
8. Quality: Very nice and ...
OpenAI GPT-4o
Augmented Information
Summarization: The comments collectively highlight the high quality of the product ... One user appreciates ... [1] Another user praises...[2][3][4]. Overall, ...[5][6][7]

Figure 3: Architecture of our LLM-based search and summarization tool for Amazon Product Reviews. This flowchart illustrates the architecture of an AI-powered search engine designed for processing and summarizing reviews about the quality of DualShock 4 Wireless Controller. The process starts with the user query, where a specific question about the quality is posed. In the retrieval phase, the query's key semantic information, "the quality of the wireless controller", is embedded and compared to filtered Amazon product reviews using cosine similarity. The system then retrieves the top 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.

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, see 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. We choose $|S_q| = 8$ as a fixed value for all queries, though in practice, the retrieval process can be more complex.[7]

[7]For example, 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.

We restrict ourselves to the eight most relevant reviews ($|S_q| = 8$) 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 next step). Second, giving irrelevant context to the generative model can worsen the quality of summaries. 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 summarize from 5 to 8 documents, with an average of 5.28 documents per query (Danny, 2024). Thus, by restricting ourselves to eight reviews, we provide the generative model with the most relevant documents while maintaining information diversity and providing users with a manageable summary. Of course, in settings where there exists a larger number of diverse and informative documents, it is easy to increase this number, and in the limit, it is possible to simply include all the reviews as context.

- **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 designed prompt in OpenAI's GPT-4o model (`GPT-4o-2024-08-06`). The prompt can be found in Figure A6 in Web Appendix §B. The prompt instructs the model to analyze the filtered reviews, exclude irrelevant information, and then generate a summary focusing solely on content related to the query.
- **Step 4: Generate the summary $A(q, S_q)$**
  Given the user query (q) and the contextual information from the relevant reviews (S_q), the LLM (GPT-4o) produces a grounded summary by retrieving and synthesizing evidence from the provided documents. Although the LLM retains knowledge from its pretraining corpus, the response in this step is guided primarily by the contextual input delivered through the augmentation prompt (see Figure A6). The output summary cites supporting reviews in square brackets (e.g., [2]), enabling users to trace each statement back to specific source documents. If a review is not relevant, it is explicitly marked as such by the model, e.g., ("[4] is not related to the query").

6.3 Evaluation of Summarized Amazon Reviews

Next, we describe the implementation of $v(\cdot)$, a function that evaluates the quality of generated summaries. To operationalize this, we design a prompt for GPT-4o that serves as the performance scoring function. The full prompt is provided in Figure A7, located in Web Appendix §B. This prompt takes summaries as inputs and outputs a performance score. The LLM evaluates each summary's informativeness based on its "Information Coverage," reflecting how well the summary captures key aspects of the product reviews.[8] Each summary is rated on a scale from 0 to 10[9] with higher scores indicating a more comprehensive and accurate reflection of relevant information. The LLM is instructed to prioritize clarity and relevance, emphasizing key

[8]Nevertheless, as discussed in §3, our framework is agnostic to the exact evaluation tool used, and other approaches can be used.
[9]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.

details. If a summary includes irrelevant content—such as the explicitly marked phrase "[X] is not related to the query".

6.4 Stochasticity in Summarization and Evaluation

LLM outputs are inherently stochastic due to their probabilistic nature in the next token prediction. Even when the temperature hyperparameter in GPT-4o is set to zero,[10] both the summarization process $A(q, S)$ and the evaluation process $v(q, A(q, S))$ yield non-deterministic outputs due to hardware-level computational variability. We empirically evaluate this effect and explore ways to manage it.

To manage this stochasticity, we experiment with different temperature settings and set the temperature to 0.1 for both summarization and evaluation prompts. This choice is guided by two metrics: semantic consistency, which measures the stability of meaning across repeated generations, and lexical diversity, which reflects variation in word usage. As shown in Web Appendix C, temperature 0.1 provides a strong balance between output stability and expressive flexibility. Prior work has also criticized temperature 0 for producing rigid and less coherent outputs (Holtzman et al., 2020).

In Web Appendix D, we further decompose the variance and find that summarization accounts for around 53% of the total variance, while evaluation contributes around 47%. Then, to further reduce the influence of randomness of LLM prompts on the variance of Shapley value, we choose to conduct four independent evaluations for each summary and use their average as the final evaluation score. This reduces evaluation variance without incurring excessive computational cost. We discuss the experiment results in Web Appendix D.2.

6.5 Implementation Details for Shapley Calculation

We now discuss the implementation details for both Exact Shapley and Cluster Shapley.

6.5.1 Exact Shapley Implementation

For each product setting, using the distribution of queries and inputs calculate the exact Shapley values using the formula in Equation (5). Table 2 presents the Shapley values for the top eight most relevant reviews in response to the query, "How is the quality of the wireless controller?" for the first product. Other reviews not contributing to this query receive a Shapley value of zero.

Review #3 has the highest Shapley value (1.83), as it directly compares the controller's quality to other versions and emphasizes functionality, aligning well with the prompt's emphasis on "Information Coverage" for quality details. Similarly, Review #7 (1.61) and Review #2 (1.58) score highly for addressing quality explicitly—#7 in a positive tone and #2 by highlighting durability compared to off-brand controllers. Review #5 (1.44) also performs well by underscoring the superior quality of the original controller versus knockoff brands. Review #4 (1.25) is somewhere in the middle, highlighting the good quality but without additional information relevant to the query. The lower-scoring reviews, including Review #1 (0.59) and Review #6

[10]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.

(0.53), just generally mention the great quality and cheap price, lacking specific details. Reviews #8 has the lowest value of 0.17, as it emphasizes aspects like shipping and being good for gifts, which are less relevant to the query's focus on controller quality, though the title mentions the "Quality", which makes it relevant to the query.

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

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

Next, we discuss the 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,[11] costing approximately $1.30 in OpenAI API fees per query. As we can see, this cost can become prohibitive in both time and money as the number and variety of queries scales up.[12]

To illustrate the scalability challenge, consider Perplexity AI, a leading LLM-based search engine. As of April 2024, Perplexity CEO -Srinivas (2024) confirmed that the platform serves over 400 million queries per month. Applying exact Shapley valuation to every query under such volume would imply more than $520 million in monthly compute cost ($1.3 per query × 400 million queries), primarily due to the high price

[11]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 use Tier 2 access.

[12]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, the 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 GPT-4o throughout the paper.

of GPT-based APIs. According to the AI/ML API inference pricing (AI/ML API, 2025), switching from `GPT-4o-2024-08-06` to `Llama-3-8B-Instruct-Lite` reduces API costs by approximately 98.4%, with GPT-4o costing \$1.313 per million tokens compared to just \$0.021 per million tokens for Llama 3. However, even with Llama 3, the cost to calculate exact Shapley values would still exceed \$8.3 million (\$520 million per month x (1- 98.4%)) per month, the cost of which is too high compared to Perplexity's annualized revenue of \$100 million as of March 2025 (Srinivas, 2025). Such computational expenses create a significant gap between theoretical document valuation frameworks and practical implementation, highlighting the need for an efficient approximation algorithm like our proposed Cluster Shapley algorithm.

6.5.2 Cluster Shapley Implementation

For the implementation of our proposed Cluster Shapley algorithm, as described in Algorithm 1, the first step is to specify the clustering diameter hyperparameter ϵ. In our numerical comparison in §7.2, we evaluate a spectrum of ϵ values to illustrate the trade-off between approximation error and computational time. Specifically, we explore ϵ values ranging from 0.01 to 1.00 in increments of 0.025. As discussed in §4.2.1, one can apply standard hyperparameter tuning procedures to select an appropriate ϵ. To assess the robustness of this choice, we conduct an additional experiment in §7.4, where we randomly split the test dataset into two subsets—one for tuning ϵ and the other for implementation and evaluation. The results demonstrate that the Cluster Shapley algorithm is fairly robust to the choice of ϵ.

After specifying the clustering diameter ϵ, we apply Algorithm 2 to perform document clustering. The computational cost of our adaptive clustering algorithm is negligible in our setting, especially when compared to the cost of LLM-based summarization and evaluation. We analyze and report the computation time of the clustering step using Algorithm 2 in Web Appendix §F.

To illustrate the power of clustering in our algorithm, Figure 4 presents the clustering results using Algorithm 2 for a sample query. 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, we append all the documents within a cluster, treat each cluster as a meta-document, and obtain cluster-level exact Shapley values. At this stage, the summarization and evaluation steps follow exactly the same prompts as those used in the exact Shapley calculation, except that we use the meta-documents at the cluster level as the input for summarization. Finally, we distribute the cluster-level Shapley values equally back to the individual documents within the clusters. As shown in the example in Figure 4, documents within the same cluster have similar exact Shapley values, and our Cluster Shapley approximation achieves high accuracy. For instance, reviews 1 and 6 are in the same cluster with similar exact Shapley values of $\phi_1 = 0.59$ and $\phi_6 = 0.53$, and the same approximated Shapley value $\hat{\phi}_1 = \hat{\phi}_6 = 0.58$. As shown in Table 2, both reviews 1 and 6 emphasize that the price is cheap but only mention that the product is great without details. The mean absolute error (MAE) between the exact and approximated Shapley values across all reviews in this example is only 0.04, demonstrating that our algorithm successfully approximates the Shapley value with

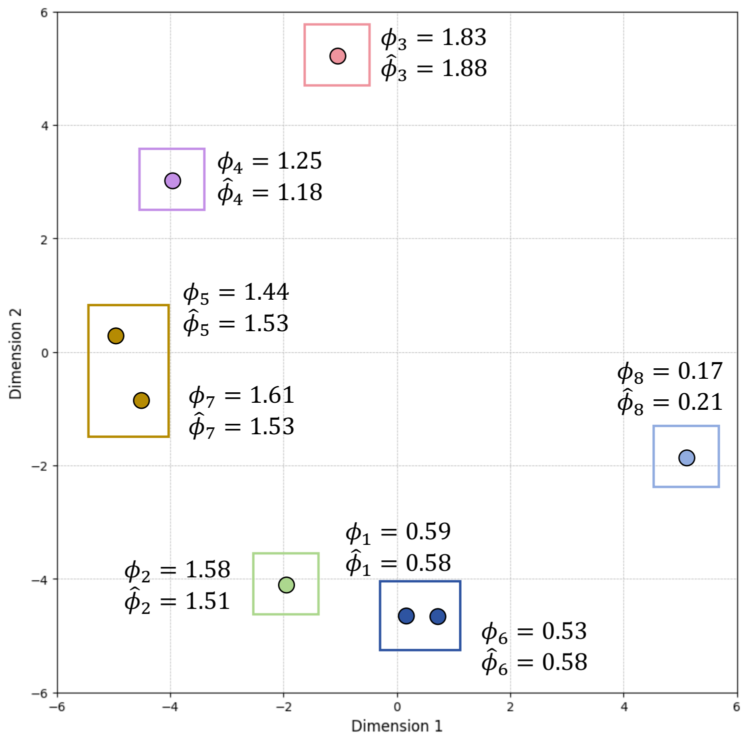


Figure 4: Clustering result of Top 8 relevant Amazon reviews for the query "How is the quality of the wireless controller?" We use 3072-dimensional OpenAI embeddings for the clustering. However, we use PCA to reduce the embedding dimension to 2 for better visualization. Dots represent the reviews, and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

a low error and reduces the computation cost. It also validates our theory that semantically similar documents tend to have comparable marginal contributions to the summary, and thus similar Shapley values.

7 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. In §7.4, we conduct a series of robustness checks.

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 popular 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 one permutation P_i^π, calculates its marginal contribution, i.e., $v(q, A(q, P_i^\pi \cup \{i\})) - v(q, A(q, P_i^\pi))$. Shapley value can then be approximated using the sample average of marginal contributions over all sampled permutations. As the number of permutation samples increases, the approximation error decreases, but the computational cost grows linearly. In our numerical experiments, we progressively increase the number of permutations to show the trade-off between accuracy and efficiency.

- **Truncated Monte Carlo:** The 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 is more likely to have higher overlapping information with other documents, reducing its marginal contribution.

  We briefly summarize the algorithm here and refer readers to Ghorbani and Zou (2019) for details. This algorithm randomly samples a permutation of reviews and sequentially calculates performance scores, v, by adding reviews in the permutation order. Since these scores are increasing, the algorithm truncates the computation by assigning zero marginal contributions to the remaining reviews when the gap between the current score and the maximum score (10 in our setting) is smaller than a pre-specified threshold, called *performance tolerance*. It basically means that when adding the remaining reviews, their marginal contributions are always smaller than this threshold. Thus, this algorithm simply assigns zero marginal contribution instead of calculating the negligible marginal value. 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 tested increasing numbers of samples to evaluate the performance of the method in different computational budgets. Kernel SHAP has been widely used across domains, including NLP for transformer-based model interpretation (Kokalj et al., 2021), finance for credit risk analysis (Famà et al., 2024), healthcare for clinical decision support (Li et al., 2022), and marketing for optimizing content engagement (Kong et al., 2023).

7.2 Numerical Comparison 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 (for all 255 possible subsets) and fix the evaluation score for each summary by averaging four evaluations, as detailed in §6.3 and §6.4. 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 6. The Y-axis represents the Mean Absolute Error (MAE) of the Shapley values, averaged across all test instances and reviews, which serves as a measure of the approximation error for each algorithm. Results under performance metrics including Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE), exhibit similar trends and can be found in Web Appendix §G.2. The X-axis represents the number of unique subsets used by the algorithms. Here, a "unique subset" refers to a distinct (non-replicated) subset of reviews used in the algorithm. For Cluster Shapley, this number represents the distinct subsets that emerge after clustering, where each cluster is treated as a new meta-review. For Monte Carlo, Truncated Monte Carlo, and Kernel SHAP, while these methods can sample the same subset multiple times, we count only the unique subsets encountered during sampling to ensure fair computational comparison. For example, if the same subset appears multiple times in these algorithms, we only evaluate it once and cache its evaluation score for reuse. Because the computation time of clustering in the Cluster Shapley algorithm and the regression step in Kernel SHAP are negligible compared to summarization and evaluation prompts, the actual computation cost of all algorithms 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.

Figure 6'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 6. 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

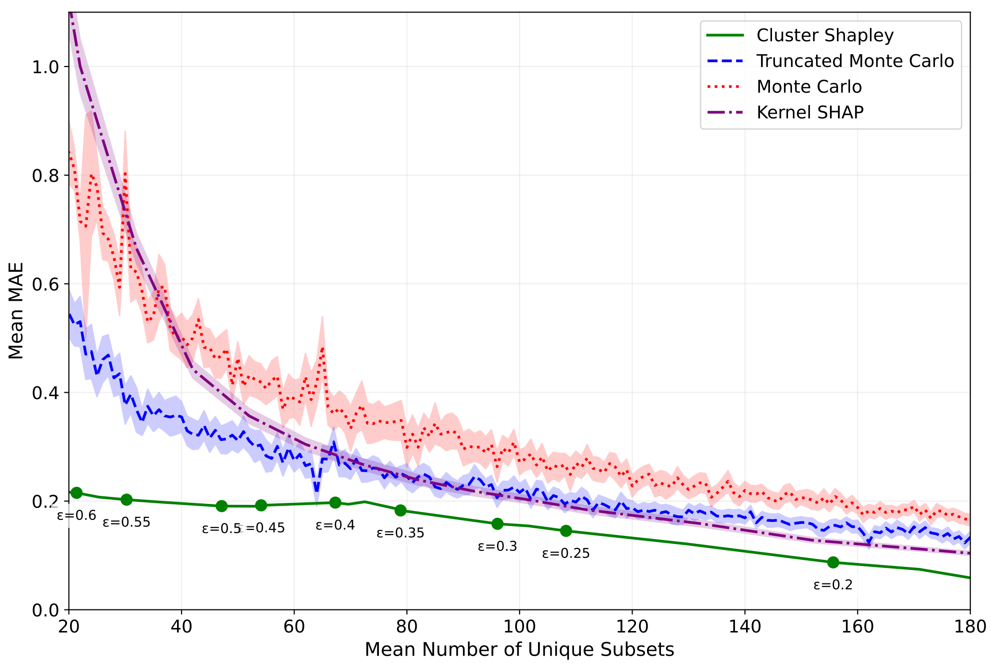


Figure 5: Performance of Shapley value approximation methods. The X-axis represents the number of unique subsets used by the algorithms, averaged across all test queries and reviews. The Y-axis represents the Mean Absolute Error (MAE) of the Shapley values, averaged across all test queries and reviews.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

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.

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.

Clustering Diameter (ϵ)	**MAE**	**MSE**	**MAPE**	**Cost Reduction**
0.01	0.0381	0.0148	7.47%	23.01%
0.10	0.0507	0.0184	8.62%	26.67%
0.20	0.0913	0.0441	11.85%	40.00%
0.30	0.1617	0.1723	17.16%	62.39%
0.40	0.1972	0.1499	21.35%	73.61%
0.50	0.1908	0.1074	21.05%	81.52%
0.60	0.2152	0.1038	24.43%	91.62%
0.70	0.2305	0.1636	26.49%	98.63%
0.80	0.2259	0.2123	26.33%	99.13%

Table 3: Approximation error (averaged over all documents) and computation time reduction of Cluster Shapley under varying ϵ. The last column, cost reduction, is calculated as the percentage reduction in the number of unique subsets used compared to all 255 subsets used by the exact Shapley. Note that when calculating MAPE, we add a small constant (0.1, approximately 10% of the mean Shapley value) to the denominator to prevent near-zero Shapley values from inflating the error.

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 3.5 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 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.

7.3 Cluster Shapley with Kernel SHAP Approximation

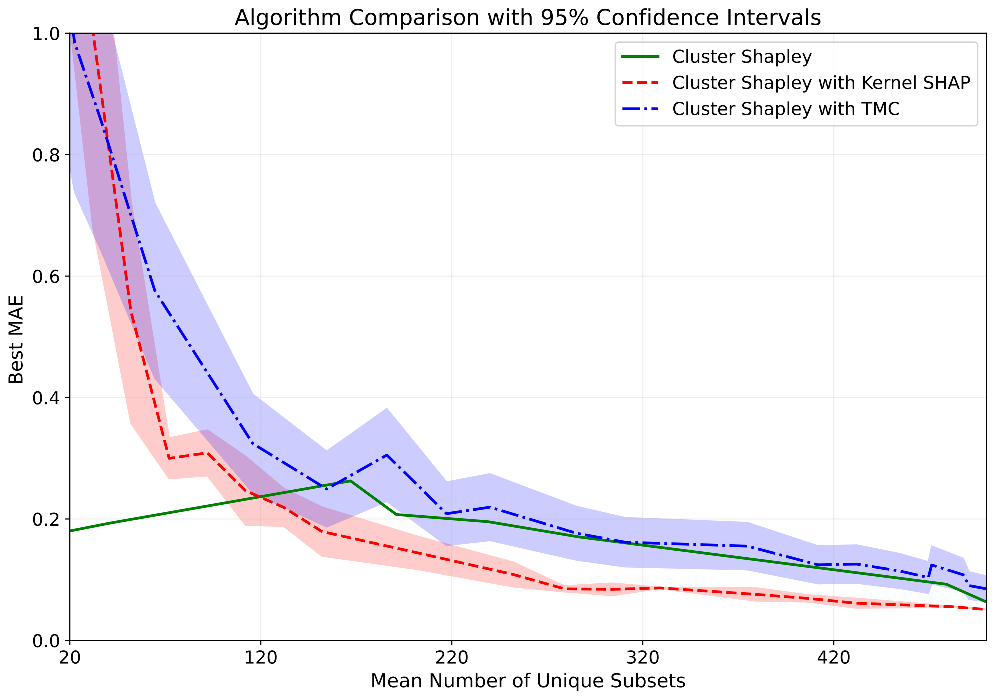


Figure 6: Performance comparison between Cluster Shapley and Cluster Shapley with Kernel SHAP. The X-axis represents the number of unique subsets evaluated by each method, averaged across all test queries and reviews. The Y-axis shows the Mean Absolute Error (MAE) of the estimated Shapley values. Points on the curves correspond to different parameter settings: clustering diameter ϵ for Cluster Shapley, and sample sizes for Kernel SHAP. 95% CIs for Cluster Kernel SHAP are computed through 10 replications of the algorithms.

While our proposed Cluster Shapley algorithm significantly reduces computational complexity by decreasing the number of unique subsets requiring summarization and evaluation, real-world scenarios could involve more extensive information sources. Although typical LLM applications rarely utilize a large number of documents simultaneously, scenarios such as extensive consumer reviews or complex information queries might necessitate processing larger document sets. Therefore, evaluating our algorithm's performance under these expanded conditions is essential.

To further enhance computational efficiency under scenarios involving numerous documents, we integrate Kernel SHAP, a widely used approximation method that estimates Shapley values through sampling, into our cluster-based approach, resulting in the Cluster Kernel SHAP method. By applying Kernel SHAP at the cluster level, we leverage clustering to minimize redundancy, allowing us to efficiently handle larger numbers of clusters.

Additionally, we examine another sampling-based approach, Cluster with Truncated Monte Carlo, to provide a comparative analysis with Cluster Kernel SHAP and the original Cluster Shapley method. This comparison helps elucidate the strengths and limitations of these approaches under varying computational and data constraints.

For the experiments, we employed the following steps. First, we randomly selected four distinct queries from our previously analyzed dataset of 48 queries, each consisting of 10 comments. This setup realistically simulates scenarios involving substantial information loads, where exact Shapley value computation requires evaluating 1,023 unique subsets, involving nearly 300,000 tokens, thus underscoring the computational intensity inherent in such analyses.

Second, we generated embeddings for the selected comments using OpenAI's text-embedding-3-large model. The embeddings were clustered using DBSCAN with cosine similarity-based adaptive distance constraints. We systematically tested various epsilon values (ranging from 0.01 to 1.0) to determine optimal clustering parameters, observing that lower epsilon values ensured higher internal similarity within clusters. Specifically, an epsilon value of 0.05 resulted in highly similar comment clustering, maintaining high precision within clusters and controlling the number of clusters effectively, averaging approximately 8.75 clusters.

Third, fixing epsilon at 0.05, we conducted Kernel SHAP calculations using various sample sizes, progressively increasing from 5 up to 2000. This comprehensive range allowed us to observe how approximation accuracy improved with increasing sample sizes and stabilized beyond certain thresholds.

Our experimental results illustrate that Cluster Kernel SHAP significantly reduces computational costs. Specifically, under the selected epsilon (0.05), the method consistently achieved low approximation errors from using 150 unique subsets, substantially decreasing computational requirements compared to Cluster Shapley calculations. Notably, while Cluster Kernel SHAP achieved a mean MAE of 0.1 using approximately 250 unique subsets, Cluster Shapley required around 450 unique subsets to achieve the same accuracy. This performance demonstrates that Cluster Kernel SHAP is particularly effective in scenarios involving extensive information sets, ensuring high accuracy with reduced computational overhead. In contrast, the Cluster with Truncated Monte Carlo method exhibited initial improvements in MAE performance relative to Cluster Kernel SHAP, but showed considerable inherent randomness as more subsets were evaluated. From approximately 50 subsets onward, the performance of Cluster with Truncated Monte Carlo deteriorated compared to Cluster Kernel SHAP, ultimately failing to outperform the basic Cluster Shapley method significantly.

Overall, this extension significantly enhances the practicality and scalability of our Cluster Shapley framework, offering a balanced solution for accuracy and computational efficiency in real-world applications involving larger document sets.

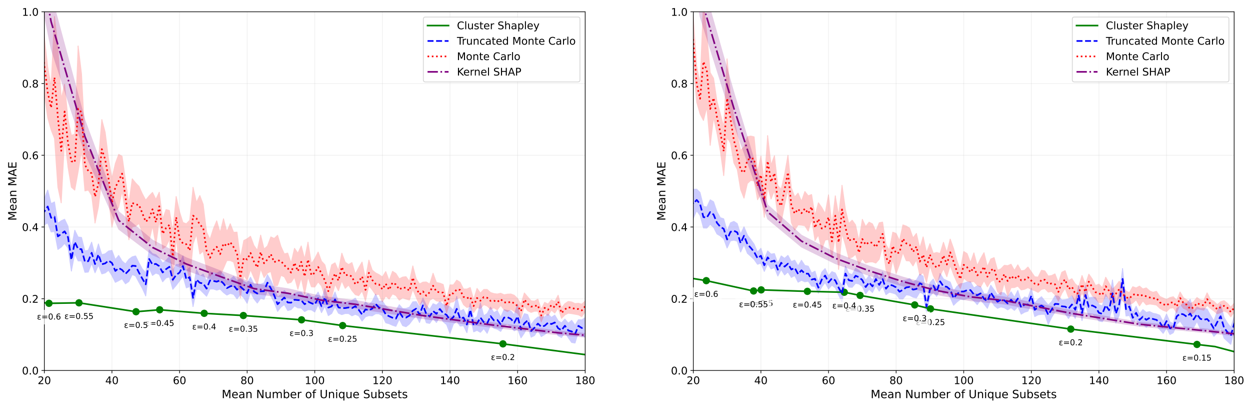
7.4 Robustness Checks

We now present a series of robustness checks on various aspects of our approach.

- In the main analysis, we use the same GPT-4o for both summarization and for evaluating the summaries. This can potentially introduce biases because LLMs tend to give higher scores to their own summaries. To address this, we conduct robustness checks using a different model, Claude, for evaluation. We see that Claude yields similar evaluation results and thus similar Shapley values. Details of this robustness check are shown in Web Appendix §G.1.
- For completeness, we also compare the approximation error of different algorithms using alternative metrics, including MSE and MAPE, and find that the comparative results are consistent with those shown

in Figure 6. Additional details of this robustness check are provided in Web Appendix §G.2.

- In the main analysis, we use our proposed adaptive DBSCAN (Algorithm 2) to enforce tight distance constraints within clusters. To examine the impact of this design choice, we conduct a robustness check using the standard (non-adaptive) DBSCAN algorithm. We find that while standard DBSCAN performs reasonably well, its performance is consistently inferior to our proposed adaptive version. Details of this comparison are provided in Web Appendix §G.3.
- To assess the robustness of our results with respect to the choice of test queries and the stability of hyperparameter tuning, we conduct a sample-splitting robustness check. Specifically, we randomly divide the 48 test queries into two equal subsets: Split 1 and Split 2, each containing 24 queries. We then replicate the main numerical comparison in Figure 6 separately for each split. The results, shown in Figure 7, confirm two key insights: (1) the comparative performance of different algorithms remains consistent across the two query subsets, and (2) the choice of the clustering diameter ϵ is straightforward and stable in our setting—one can use one subset for hyperparameter tuning and the other for evaluation without performance degradation.



(a) Split 1 (24 queries)

(b) Split 2 (24 queries)

Figure 7: Robustness check using random query splits.

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

AI/ML API. Ai/ml api inference pricing, 2025. URL https://aimlapi.com/ai-ml-api-pricing.

Wilfred Amaldoss and Jinzhao Du. How can publishers collaborate and compete with news aggregators? *Journal of Marketing Research*, 60(6):1114–1134, 2023.

Susan Athey, Markus Mobius, and Jeno Pal. The impact of aggregators on internet news consumption. Technical report, National Bureau of Economic Research, 2021.

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.

Joan Calzada and Ricard Gil. What do news aggregators do? evidence from google news in spain and germany. *Marketing Science*, 39(1):134–167, 2020.

Sarah Carroll. Will google's ai overview kill web traffic?, January 2025. URL https://www.forumone.com/insights/blog/will-googles-ai-overview-kill-web-traffic/. Accessed: 2025-04-17.

Ashley Chang. How i became yelp elite in 48 days, August 2023. URL https://ashleychangg.medium.com/how-i-became-yelp-elite-in-48-days-367abaa3879. Accessed: 2025-04-17.

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.

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.

Chrysanthos Dellarocas, Zsolt Katona, and William Rand. Media, aggregators, and the link economy: Strategic hyperlink formation in content networks. *Management science*, 59(10):2360–2379, 2013.

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.

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

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

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

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

Danny Goodwin. Google ceo says ai overviews are increasing search usage, April 2024. URL https://searchengineland.com/google-ceo-says-ai-overviews-are-increasing-search-usage-439983. Accessed: 2025-04-21.

Google. AI overviews – search anything, effortlessly, 2025. URL https://search.google/ways-to-search/ai-overviews/. Accessed: 2025-04-17.

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.

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.

Doh-Shin Jeon and Nikrooz Nasr. News aggregators and competition among newspapers on the internet. *American Economic Journal: Microeconomics*, 8(4):91–114, 2016.

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.

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

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

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

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

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

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

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

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.

Dina Mayzlin and Hema Yoganarasimhan. Link to success: How blogs build an audience by promoting rivals. *Management Science*, 58(9):1651–1668, 2012.

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.

Microsoft. Copilot search in bing, 2025. URL https://www.microsoft.com/en-us/bing/copilot-search/?form=MA13XW&cs=82041551. Accessed: 2025-04-17.

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

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

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.

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

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.

Perplexity AI. Perplexity ai - answer engine, 2025. URL https://www.perplexity.ai. Accessed: 2025-04-17.

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

Reddit Help. What is karma?, November 2024. URL https://support.reddithelp.com/hc/en-us/articles/204511829-What-is-karma. Accessed: 2025-04-17.

Elizabeth Reid. Generative AI in search: Let google do the searching for you, May 2024. URL https://blog.google/products/search/generative-ai-google-search-may-2024/. Accessed: 2025-04-19.

Katie Robertson. Openai strikes a deal to license news corp content, May 2024. URL https://www.nytimes.com/2024/05/22/business/media/openai-news-corp-content-deal.html. Accessed: 2025-04-17.

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.

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

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

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

URL https://www.eweek.com/news/google-ai-overviews-smb-impact/. Accessed: 2025-04-17.

Yu Song and Puneet Manchanda. Does carrying news increase engagement with non-news content on social media platforms? *Available at SSRN 4490982*, 2023.

Aravind Srinivas. Perplexity serving over 100 million queries per week, April 2024. URL https://x.com/AravSrinivas/status/1849813363850649959.

Aravind Srinivas. Perplexity has crossed $100m in annualized revenue, March 2025. URL https://www.linkedin.com/posts/aravind-srinivas-16051987_perplexity-has-crossed-100m-in-annualized-activity-7310678232079495170-38ik/.

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

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

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

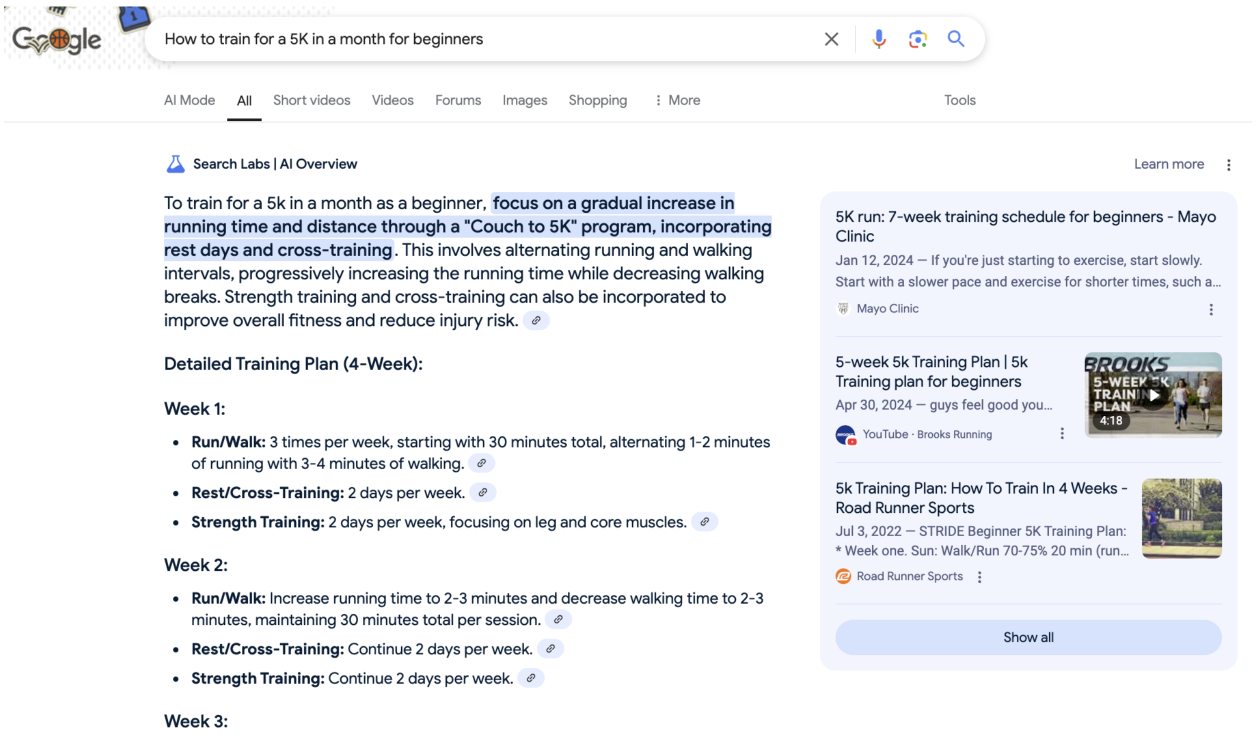
Shuo Zhang, Boci Peng, Xinping Zhao, Boren Hu, Yun Zhu, Yanjia Zeng, and Xuming Hu. Llasa: Large language and e-commerce shopping assistant. *arXiv preprint arXiv:2408.02006*, 2024.

Web Appendix

A Illustration Examples of AI-Generated Summaries

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

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



To train for a 5k in a month as a beginner, **focus on a gradual increase in running time and distance through a "Couch to 5K" program, incorporating rest days and cross-training**. This involves alternating running and walking intervals, progressively increasing the running time while decreasing walking breaks. Strength training and cross-training can also be incorporated to improve overall fitness and reduce injury risk.

Detailed Training Plan (4-Week):

Week 1:

- **Run/Walk:** 3 times per week, starting with 30 minutes total, alternating 1-2 minutes of running with 3-4 minutes of walking.
- **Rest/Cross-Training:** 2 days per week.
- **Strength Training:** 2 days per week, focusing on leg and core muscles.

Week 2:

- **Run/Walk:** Increase running time to 2-3 minutes and decrease walking time to 2-3 minutes, maintaining 30 minutes total per session.
- **Rest/Cross-Training:** Continue 2 days per week.
- **Strength Training:** Continue 2 days per week.

Week 3:

5K run: 7-week training schedule for beginners - Mayo Clinic
Jan 12, 2024 — If you're just starting to exercise, start slowly. Start with a slower pace and exercise for shorter times, such a...
Mayo Clinic

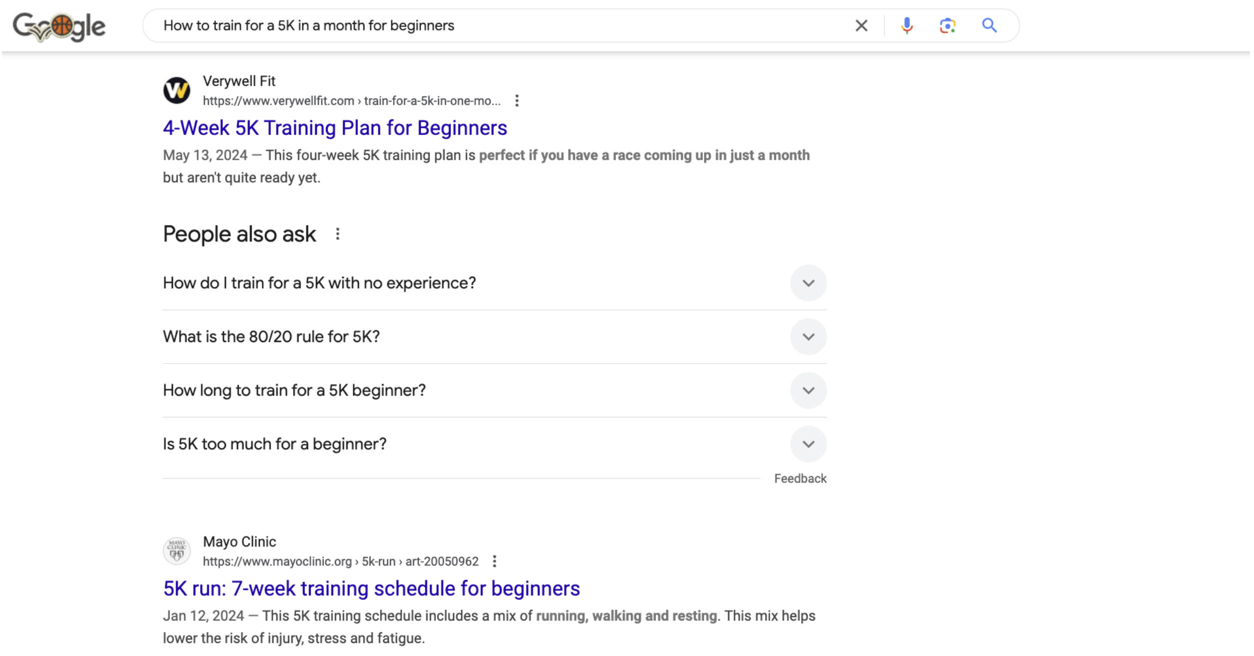
5-week 5k Training Plan | 5k Training plan for beginners
Apr 30, 2024 — guys feel good you...
YouTube · Brooks Running

5k Training Plan: How To Train In 4 Weeks - Road Runner Sports
Jul 3, 2022 — STRIDE Beginner 5K Training Plan: * Week one. Sun: Walk/Run 70-75% 20 min (run...
Road Runner Sports

Figure A1: Google AI Overview

The next example, shown in Figure A4, comes from Reddit, a Q&A forum. Reddit is currently piloting a new RAG-enhanced Q&A assistant called *Reddit Answers* (Reddit, 2025). Using the same query, "How to train for a 5K in a month for beginners?", Reddit Answers generates a summarized response based on user-generated content from relevant Reddit threads. Similar to Google's AI Overview, the summary is accompanied by links to the original responses, though the content is limited to the Reddit platform.

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



How to train for a 5K in a month for beginners

Verywell Fit
https://www.verywellfit.com › train-for-a-5k-in-one-mo...

4-Week 5K Training Plan for Beginners

May 13, 2024 — This four-week 5K training plan is **perfect if you have a race coming up in just a month** but aren't quite ready yet.

People also ask

How do I train for a 5K with no experience?

What is the 80/20 rule for 5K?

How long to train for a 5K beginner?

Is 5K too much for a beginner?

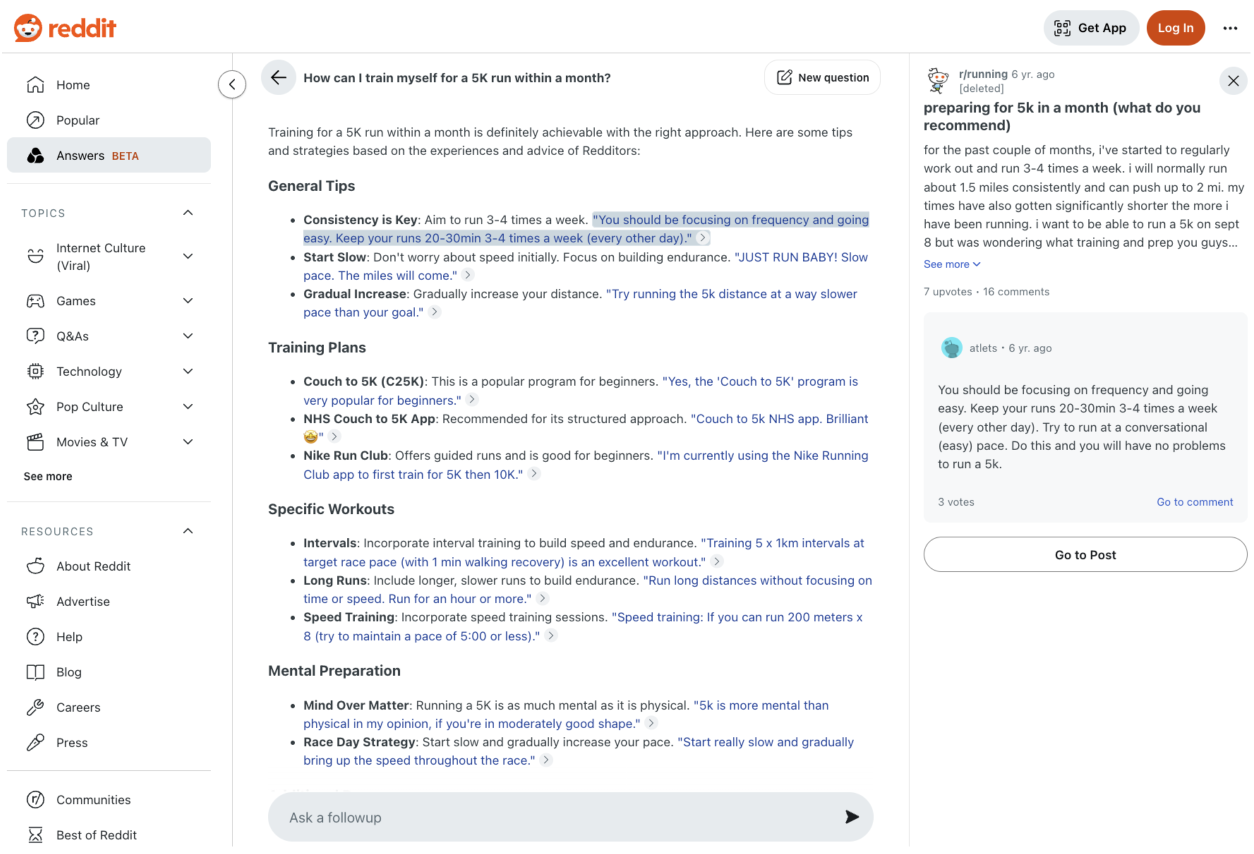
Feedback

Mayo Clinic
https://www.mayoclinic.org › 5k-run › art-20050962

5K run: 7-week training schedule for beginners

Jan 12, 2024 — This 5K training schedule includes a mix of **running, walking and resting**. This mix helps lower the risk of injury, stress and fatigue.

Figure A2: Google Search without AI Overview



How can I train myself for a 5K run within a month?

Training for a 5K run within a month is definitely achievable with the right approach. Here are some tips and strategies based on the experiences and advice of Redditors:

General Tips

- **Consistency is Key**: Aim to run 3-4 times a week. "You should be focusing on frequency and going easy. Keep your runs 20-30min 3-4 times a week (every other day)."
- **Start Slow**: Don't worry about speed initially. Focus on building endurance. "JUST RUN BABY! Slow pace. The miles will come."
- **Gradual Increase**: Gradually increase your distance. "Try running the 5k distance at a way slower pace than your goal."

Training Plans

- **Couch to 5K (C25K)**: This is a popular program for beginners. "Yes, the 'Couch to 5K' program is very popular for beginners."
- **NHS Couch to 5K App**: Recommended for its structured approach. "Couch to 5k NHS app. Brilliant 🤩"
- **Nike Run Club**: Offers guided runs and is good for beginners. "I'm currently using the Nike Running Club app to first train for 5K then 10K."

Specific Workouts

- **Intervals**: Incorporate interval training to build speed and endurance. "Training 5 x 1km intervals at target race pace (with 1 min walking recovery) is an excellent workout."
- **Long Runs**: Include longer, slower runs to build endurance. "Run long distances without focusing on time or speed. Run for an hour or more."
- **Speed Training**: Incorporate speed training sessions. "Speed training: If you can run 200 meters x 8 (try to maintain a pace of 5:00 or less)."

Mental Preparation

- **Mind Over Matter**: Running a 5K is as much mental as it is physical. "5k is more mental than physical in my opinion, if you're in moderately good shape."
- **Race Day Strategy**: Start slow and gradually increase your pace. "Start really slow and gradually bring up the speed throughout the race."

Ask a followup

r/running 6 yr. ago
[deleted]

preparing for 5k in a month (what do you recommend)

for the past couple of months, i've started to regularly work out and run 3-4 times a week. i will normally run about 1.5 miles consistently and can push up to 2 mi. my times have also gotten significantly shorter the more i have been running. i want to be able to run a 5k on sept 8 but was wondering what training and prep you guys...

See more

7 upvotes · 16 comments

atlets · 6 yr. ago

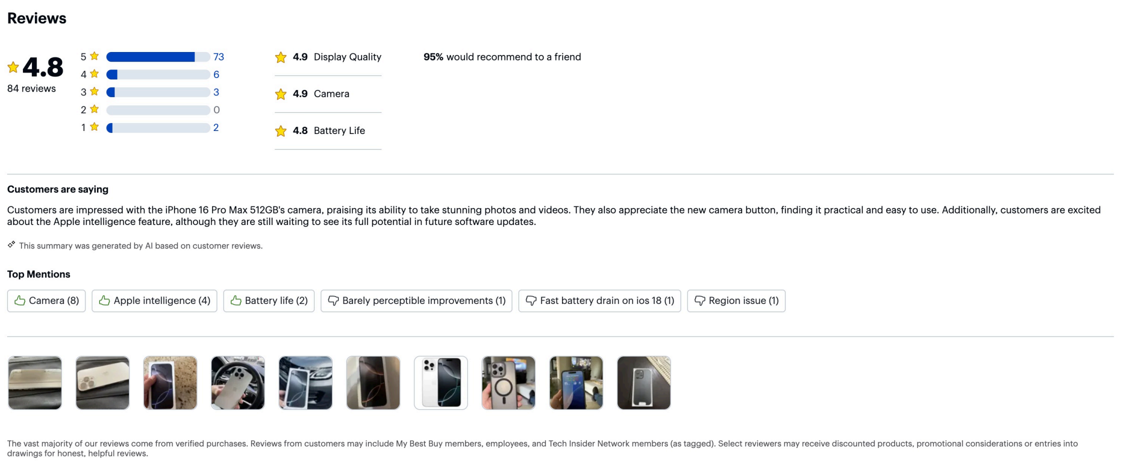
You should be focusing on frequency and going easy. Keep your runs 20-30min 3-4 times a week (every other day). Try to run at a conversational (easy) pace. Do this and you will have no problems to run a 5k.

3 votes Go to comment

Go to Post

Figure A3: Reddit AI-Generated Answer

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



Reviews

★ **4.8**
84 reviews

5 ★	73
4 ★	6
3 ★	3
2 ★	0
1 ★	2

★ **4.9** Display Quality

★ **4.9** Camera

★ **4.8** Battery Life

95% would recommend to a friend

Customers are saying

Customers are impressed with the iPhone 16 Pro Max 512GB's camera, praising its ability to take stunning photos and videos. They also appreciate the new camera button, finding it practical and easy to use. Additionally, customers are excited about the Apple intelligence feature, although they are still waiting to see its full potential in future software updates.

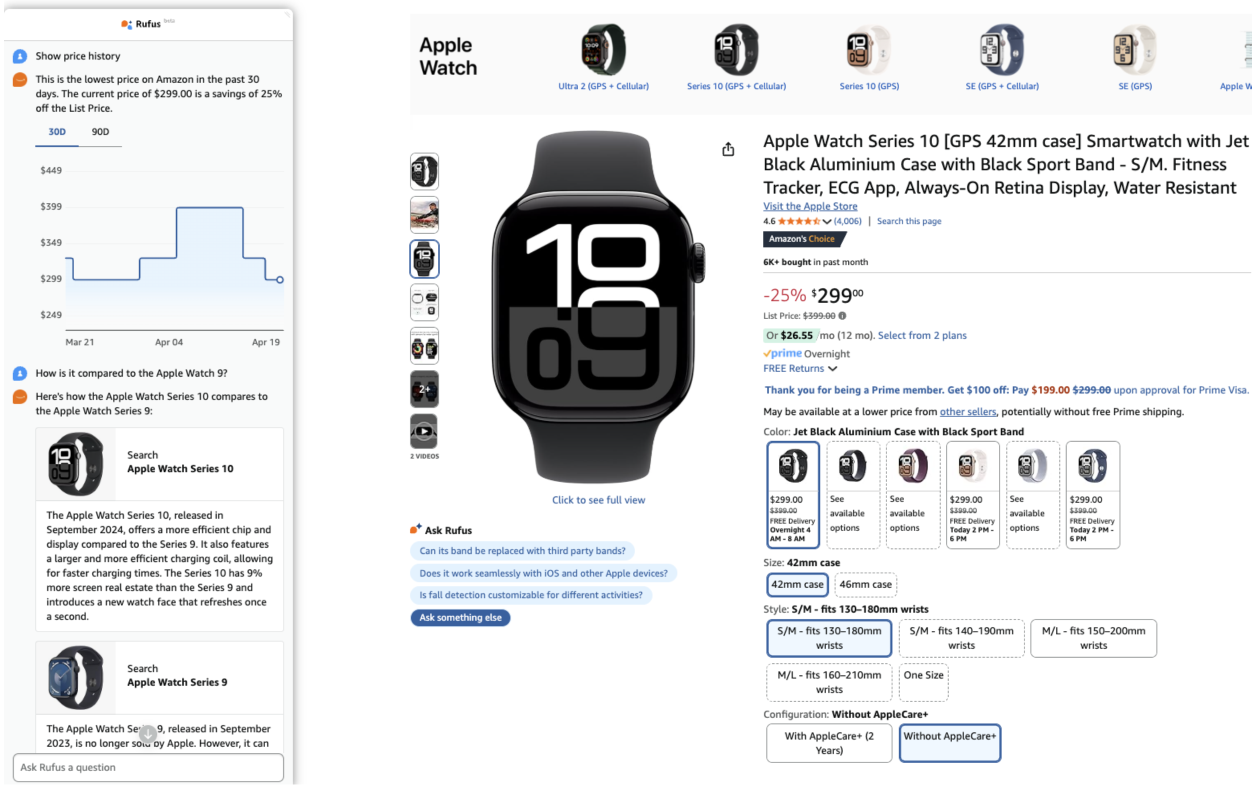
This summary was generated by AI based on customer reviews.

Top Mentions

Camera (8) · Apple intelligence (4) · Battery life (2) · Barely perceptible improvements (1) · Fast battery drain on ios 18 (1) · Region issue (1)

The vast majority of our reviews come from verified purchases. Reviews from customers may include My Best Buy members, employees, and Tech Insider Network members (as tagged). Select reviewers may receive discounted products, promotional considerations or entries into drawings for honest, helpful reviews.

Figure A4: Best Buy AI-Generated Product Review



Rufus beta

Show price history

This is the lowest price on Amazon in the past 30 days. The current price of $299.00 is a savings of 25% off the List Price.

How is it compared to the Apple Watch 9?

Here's how the Apple Watch Series 10 compares to the Apple Watch Series 9:

Search
Apple Watch Series 10

The Apple Watch Series 10, released in September 2024, offers a more efficient chip and display compared to the Series 9. It also features a larger and more efficient charging coil, allowing for faster charging times. The Series 10 has 9% more screen real estate than the Series 9 and introduces a new watch face that refreshes once a second.

Search
Apple Watch Series 9

The Apple Watch Se[illegible] 9, released in September 2023, is no longer so[illegible] by Apple. However, it can

Ask Rufus a question

Apple Watch

Ultra 2 (GPS + Cellular) · Series 10 (GPS + Cellular) · Series 10 (GPS) · SE (GPS + Cellular) · SE (GPS) · Apple W

Apple Watch Series 10 [GPS 42mm case] Smartwatch with Jet Black Aluminium Case with Black Sport Band - S/M. Fitness Tracker, ECG App, Always-On Retina Display, Water Resistant

Visit the Apple Store

4.6 ★★★★☆ (4,006) | Search this page

Amazon's Choice

6K+ bought in past month

-25% $299.00

List Price: ~~$399.00~~

Or **$26.55** /mo (12 mo). Select from 2 plans

prime Overnight

FREE Returns

Thank you for being a Prime member. Get $100 off: Pay $199.00 ~~$299.00~~ upon approval for Prime Visa.

May be available at a lower price from other sellers, potentially without free Prime shipping.

Color: **Jet Black Aluminium Case with Black Sport Band**

$299.00 ~~$399.00~~ FREE Delivery **Overnight 4 AM - 8 AM**	See available options	See available options	$299.00 ~~$399.00~~ FREE Delivery **Today 2 PM - 6 PM**	See available options	$299.00 ~~$399.00~~ FREE Delivery **Today 2 PM - 6 PM**

Size: **42mm case**

[✓ 42mm case] 46mm case

Style: **S/M - fits 130–180mm wrists**

[✓ S/M - fits 130–180mm wrists] S/M - fits 140–190mm wrists · M/L - fits 150–200mm wrists · M/L - fits 160–210mm wrists · One Size

Configuration: **Without AppleCare+**

With AppleCare+ (2 Years) [✓ Without AppleCare+]

Click to see full view

2 VIDEOS

Ask Rufus

Can its band be replaced with third party bands?

Does it work seamlessly with iOS and other Apple devices?

Is fall detection customizable for different activities?

Ask something else

Figure A5: Amazon AI-Powered Shopping Aisstant - Rufus

B Prompt Designs

We now provide our designed GPT-4o prompts for both summarization ($A(q, S)$) and evaluation ($v(q, A)$) steps, outlined in Figure A6 and Figure A7 respectively.

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

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

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

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

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

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

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

Figure A6: Prompt for GPT-4o to analyze the relevant reviews to the original query and generate a summary. The prompt specifies citation rules and explicitly requires noting if reviews are irrelevant. The structured output is generated and formatted in JSON, consisting of two fields: "key" for indexing the source reviews and "summary" for the final generated text, complete with appropriate citations.

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

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

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

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

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

Figure A7: Prompt for GPT-4o to evaluate the extent to which a summary accurately and comprehensively captures the key product attributes as requested in the query.

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

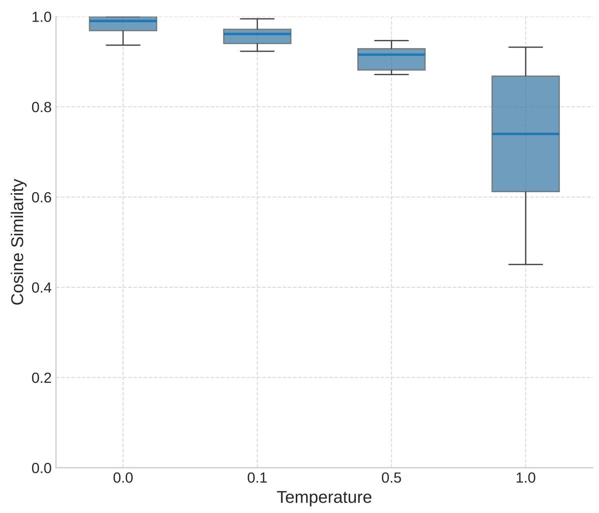
Due to the inherent stochastic nature of LLM outputs, we now investigate how the temperature setting in GPT-4o can affect the outcomes from summarization and evaluation prompts.

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

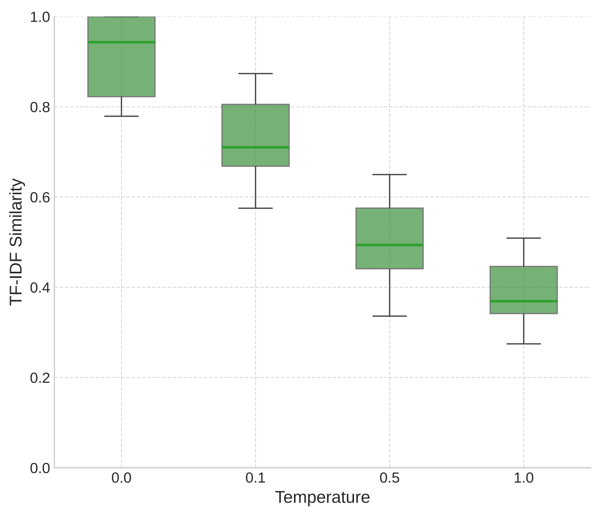
Detailed results for the summarization and evaluation experiments are presented in Web Appendix C.1 and Web Appendix C.2, respectively. For the summarization, we select a temperature of 0.1 to balance output consistency and expression richness. For the evaluation, we also choose a temperature of 0.1.

C.1 Temperature for Summarization Prompt

Because the outputs of summarization prompts are text information, to gauge the performance, we use two metrics: (1) We assess semantic consistency through embedding-based cosine similarity, which measures the degree to which different summaries preserve the same underlying meaning regardless of specific word choices; (2) We examine lexical diversity using TF-IDF (Term Frequency–Inverse Document Frequency)



(a) Effect of Temperature on Summarization Semantic Consistency



(b) Effect of Temperature on Lexical Diversity

Figure A8: Analysis of Temperature's Effect on Summary Generation. (a) Cosine similarity of embeddings measures semantic consistency, where higher values indicate stronger preservation of meaning across generated summaries. (b) TF-IDF similarity reflects lexical choice patterns, where lower values indicate more diverse vocabulary usage in the generated text, demonstrating the trade-off between consistency and diversity at different temperatures.

similarity measures, which quantify the variation in vocabulary and phrasing across multiple generations. While semantic consistency measures reliability in meaning preservation, lexical diversity reflects the model's creativity in expression. Figure A8 visualizes the effect of temperature settings on both metrics, and Table **??** presents the detailed statistical results across all temperature configurations.

As shown in Figure A8(a), even at temperature 0.0, where theoretically deterministic behavior is expected, the model exhibits slight variations in output (mean cosine similarity = 0.9820, std = 0.0217), confirming the presence of hardware-level computational variability. As temperature increases, we observe a decrease in semantic consistency, with mean similarities of 0.9576 (temp = 0.1), 0.8545 (temp = 0.5), and 0.7339 (temp = 1.0). While the decline in semantic consistency from temperature 0.0 to 0.1 is modest (approximately 2.5%), this minor trade-off shows significant improvements in lexical diversity, as demonstrated in the TF-IDF analysis in Figure A8(b). Specifically, when transitioning from temperature 0.0 to 0.1, we observe a beneficial decrease in TF-IDF similarity from 0.9102 to 0.7244, indicating substantially more diverse vocabulary usage while maintaining semantic integrity. This optimal balance point at temperature 0.1 enables richer and more nuanced expression through varied word choices, while preserving the essential meaning of the content with high semantic consistency. However, at higher temperatures, both metrics indicate potential instability in the generation process. The substantial increase in semantic standard deviation (from 0.0217 at temp = 0.0 to 0.1696 at temp = 1.0) suggests increasingly unpredictable semantic variations, while the further decrease in TF-IDF similarity (0.5013 at temp = 0.5 and 0.3845 at temp = 1.0) indicates excessive vocabulary variation. These patterns at higher temperatures could potentially compromise both semantic reliability and textual coherence, reinforcing our choice of temperature 0.1 as the optimal setting for balancing semantic preservation with expressive diversity.

C.2 Temperature for Evaluation Prompt

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

Using the same setup as in our summarization analysis, we fix the generated summaries and repeat the evaluation process under different temperature settings. This isolates the variance caused solely by the evaluation prompt. As shown in Figure A9, even at temperature 0.0, evaluation outputs exhibit variability due to hardware-level computational noise, similar to what we observed in summarization.

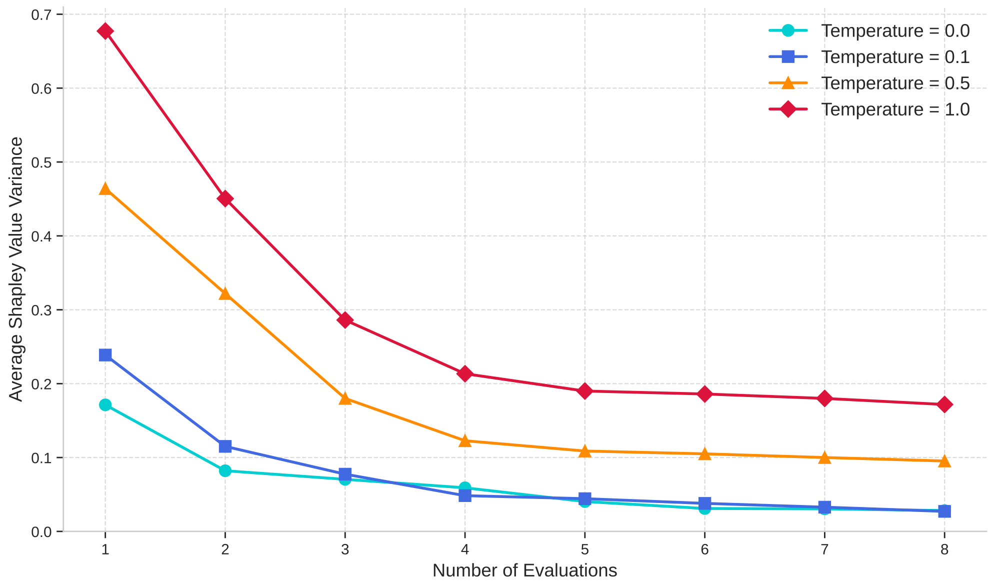


Figure A9: Shapley value variance under different temperatures and different replications of evaluation prompts. The X-axis represents the number of replications of evaluations. The Y-axis is the variance of the average Shapley value over replications.

Among the tested settings, temperature 0.0 yields the lowest Shapley variance but still shows noticeable fluctuations. Temperature 0.1 displays slightly higher variance but achieves similar stability after averaging the evaluation score over multiple replications, making it a practical choice that balances consistency and alignment with our summarization process. In contrast, higher temperatures (0.5 and 1.0) result in substantially larger variances, suggesting diminished reliability.

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

D Variance Analysis of Shapley Value

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

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

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

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

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

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

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 $\text{Var}(\varepsilon)$.

D.1 Empirical Variance in Summarization and Evaluation

To validate this variance decomposition and gauge the magnitudes of these variances, we conduct an experiment to quantify the variance contributions from both the summarization and evaluation stages.

We select five distinct queries from Amazon product reviews, each containing five reviews, which is the same setting as in Web Appendix §C. For each query, we calculate the Shapley value to explore how the summarization and evaluation processes contribute to the overall variance. Specifically, for each subset of reviews in a query, we replicate both summarization and evaluation prompts three times to calculate the following empirical variance.

- **Total Variance** $\text{Var}(v(A(S)))$ — This variance represents the overall variability of the evaluation scores for a given subset, considering all summarization and evaluation rounds. For each subset of reviews, we generate multiple summarizations and perform several evaluation rounds for each summarization. The total variance is calculated as the variance of all the evaluation scores across these rounds, capturing the combined effects of both summarization and evaluation.
- **Summarization Variance** $\text{Var}(\mu(A(S)))$ — This variance reflects the variability introduced during the summarization process. After generating multiple summaries for each subset, we compute the mean evaluation score for each summary. The summarization process variance is then calculated as the

variance of these mean evaluation scores across different summarizations. This captures how much the content of the summaries themselves contributes to the overall variability in evaluation scores, independent of the evaluation noise.

- **Evaluation Variance** Var(ε) — This variance isolates the variability introduced during the evaluation process. For each summarization, we evaluate the subset multiple times. The evaluation noise variance is computed as the average variance of the scores within each summarization round, reflecting the inconsistency of GPT-based evaluations across the same summary. In other words, it measures how much the scores fluctuate due to noise in the evaluation model rather than changes in the summaries themselves.

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

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

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

D.2 Reduce the Variance of Shapley Value

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 summarization multiple times to obtain an averaged score across evaluations, thereby reducing variance introduced by ε.

However, more summarization and evaluation replications mean higher computation cost, although with lower variance in Shapley. To determine the most cost-effective approach, we test variance under various configurations, including multiple evaluations and summarization. Specifically, we conduct the experiment using a single query, "the delivery speed of the card," with four selected reviews. For each combination of summarization counts (ranging from 1 to 4) and evaluation counts (also from 1 to 4), we repeat the process six times to calculate the variance in Shapley value. We measure computational cost in terms of the total response time of the GPT-4o API required for each configuration.

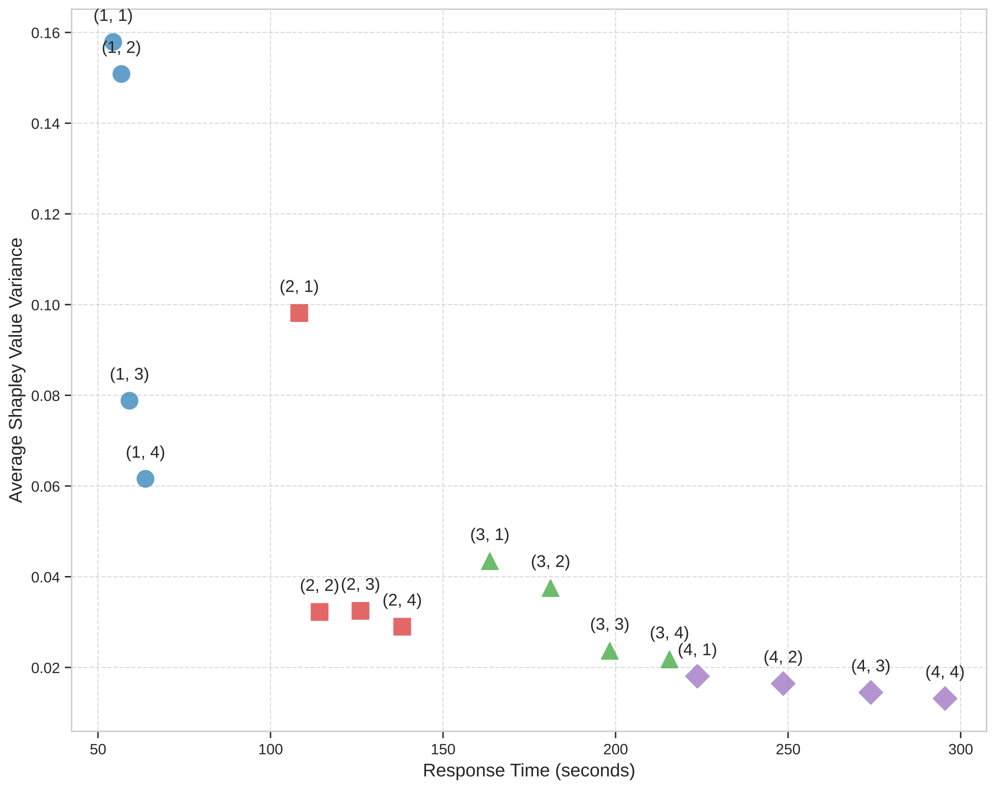


Figure A10: Cost-Effectiveness Analysis. The X-axis is the API response time (in seconds), and the Y-axis is the average Shapley value variance. Each labeled point represents a configuration, with the first number indicating the number of summarizations and the second the number of evaluations.

The results are displayed in Figure A10. We can observe that increasing both the summarization and evaluation counts reduces the average variance of the Shapley values. Notably, the most significant decrease in variance occurs when moving from lower to moderate counts of summarizations and evaluations. For example, increasing the evaluation count from 1 to 3 while keeping the summarization count at 1 reduces the average variance from approximately 0.1579 to 0.0788. Similarly, increasing the summarization count from 1 to 2 with an evaluation count of 2 decreases the average variance from approximately 0.1508 to 0.0322.

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

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

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

As shown in Figure A11, the variance in Shapley values decreases as the number of evaluation repetitions increases, but the rate of reduction diminishes over time. The most notable improvement occurs between one and four evaluations, where the average variance drops from approximately 0.25 to 0.05. Beyond four repetitions, additional variance reduction becomes marginal. These findings suggest that averaging over four evaluations provides an effective trade-off between computational cost and variance reduction. Accordingly, we adopt this configuration in our main numerical experiments.

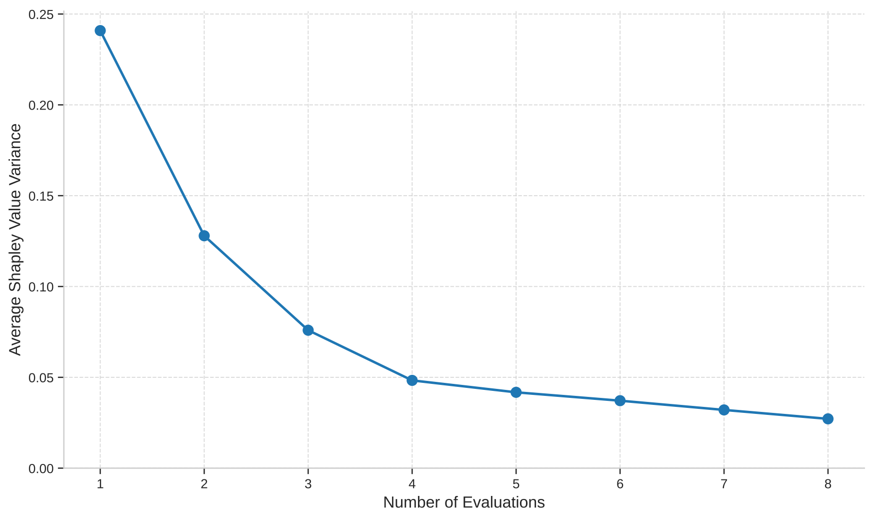


Figure A11: Reduction in Shapley Value Variance with Increased Evaluations

E Proofs

In this appendix, we present the proofs for the propositions in the main text.

E.1 Proof for Proposition 1

Proof. Without loss of generality, we assume the document i is in the cluster G_k. By the clustering construction, for any two documents i, j in the same cluster G_k, we have $d(e_i, e_j) \leq \epsilon$. Then by Assumption

1, for every coalition S not containing i or j, the marginal contributions of i and j differ by at most $L\epsilon$. That is:

$$\left|[v(S \cup \{i\}) - v(S)] - [v(S \cup \{j\}) - v(S)]\right| \leq L\epsilon. \tag{A3}$$

Now consider the exact Shapley values ϕ_i and ϕ_j. Using the permutation definition in Equation (6), $\phi_i - \phi_j$ can be expressed as the difference in i and j's marginal contributions to S, and then weighted average over all coalitions S. Because each such difference is bounded by $L\epsilon$ as shown in Equation (A3), it follows that $|\phi_i - \phi_j| \leq L\epsilon$. In other words, all members of a cluster have Shapley values within a range of at most $L\epsilon$, i.e., for any i and j in G_k,

$$|\phi_i - \phi_j| \leq L\epsilon. \tag{A4}$$

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

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

We now claim that:

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

To see this, define a function $\psi(G_k) := \sum_{j \in G_k} \phi_j$. This defines an allocation of value to clusters based on the sum of Shapley values in the original game. Since the original Shapley allocation is efficient,

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

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

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

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

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

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

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

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

$$|\hat{\phi}_i - \phi_i| \le \max\{\phi_{\max} - \phi_i,\ \phi_i - \phi_{\min}\} \le \phi_{\max} - \phi_{\min}. \tag{A5}$$

But from the Equation (A4), we know $\phi_{\max} - \phi_{\min} \le L\epsilon$. Hence $|\hat{\phi}_i - \phi_i| \le L\epsilon$ as claimed in Equation (12). In the limit $\epsilon \to 0$, every document will eventually stand alone (since there is a finite minimum nonzero distance between any two distinct document embeddings in S_q). In that extreme case, the algorithm reproduces the exact Shapley values with zero error. Thus $\hat{\phi}_i \to \phi_i$ for all i. □

E.2 Proof for Corollary 1

Proof. Given $\phi_{\max} - \phi_{\min} = \delta$ and as shown in the proof of Proposition 1 (see Equation (A5)), we have $|\hat{\phi}_i - \phi_i| \le \delta$ for each i. In the symmetric case ($\delta = 0$ by the definition of the symmetry property), then we have $|\hat{\phi}_i - \phi_i| = 0$. □

E.3 Proof for Proposition 2

Proof. The total error $|\tilde{\phi}_i - \phi_i|$ can be decomposed into two parts:

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

From Proposition 1, we already have $|\hat{\phi}_i - \phi_i| \le L\epsilon$.

Now consider the estimation of $\hat{\phi}_{G_k}$ using N random permutations. Let X_t denote the marginal contribution of G_k in the t-th permutation. Then $\tilde{\phi}_{G_k} = \frac{1}{N}\sum_{t=1}^{N} X_t$. Since each $X_t \in [0, V_{\max}]$, by Hoeffding's inequality:

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

To ensure the deviation is at most δ with probability at least $1-\eta$, it suffices to set:

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

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

$$|\tilde{\phi}_i - \hat{\phi}_i| = \frac{|\tilde{\phi}_{G_k} - \hat{\phi}_{G_k}|}{|G_k|} \le \frac{\delta}{|G_k|}.$$

Combining the bounds yields:

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

To guarantee $|\tilde{\phi}_i - \phi_i| \le \varepsilon_{\text{total}}$, solve:

$$\frac{V_{\max}}{|G_k|}\sqrt{\frac{\log(2/\eta)}{2N}} \le \varepsilon_{\text{total}} - L\epsilon,$$

which gives the required bound on N:

$$N \ge \frac{V_{\max}^2}{2|G_k|^2(\varepsilon_{\text{total}} - L\epsilon)^2}\log\left(\frac{2}{\eta}\right).$$

The clustering phase requires $O(n^2)$ operations, and each permutation for Monte Carlo contributes $O(m)$ evaluations. So N permutations contribute $O(Nm)$, and the total complexity is $O(n^2 + Nm)$. $\square$

F Computation time of the Clustering Step

We now analyze our adaptive clustering (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 1.53×10^{-3} second, yielding an average time of 1.53×10^{-9} second per iteration. While we acknowledge that measurements at such small time scales can be subject to system-level variations due to factors such as CPU scheduling, memory access patterns, and hardware-specific optimizations, these results consistently demonstrate that the iteration overhead is orders of magnitude smaller than the LLM operations in our proposed Cluster Shapley Algorithm Step 2 in §4.2, 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.

G Robustness Check

We now present detailed results and discussion for several aspects of the robustness check mentioned in §7.4.

G.1 Robustness Check using Claude for Evaluation

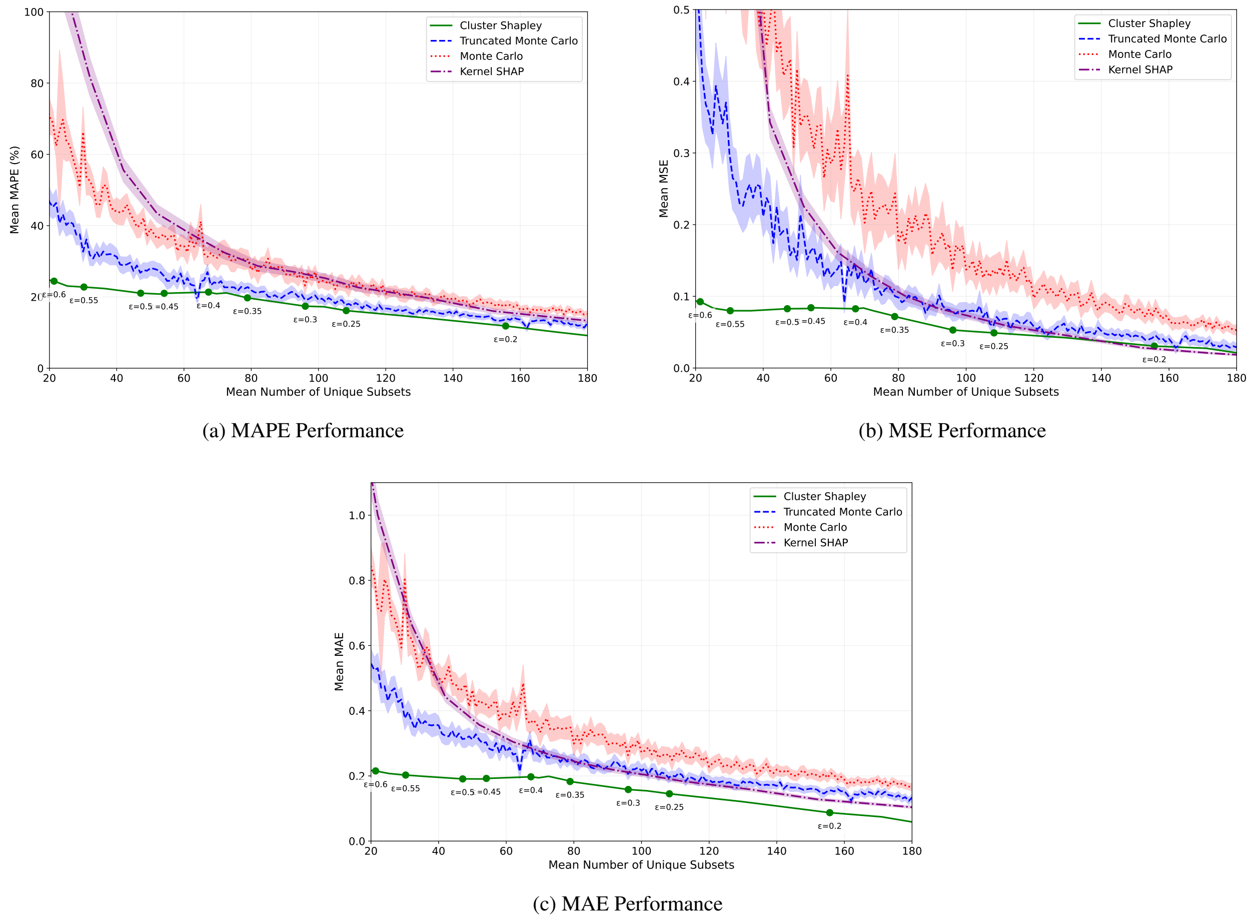
In the main study, we use the same GPT-4o for both summarization and for evaluating the summaries, which may introduce biases because LLMs tend to favor their own summaries. To address this, we conduct a robustness check here using `Claude-3.5-Sonnet` as an alternative evaluation model to compare against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

To ensure a fair comparison, we use the same prompt (as shown in Figure A7) and the same temperature setting of 0.1 for evaluation in Claude-3.5. For each summary generated by GPT-4o, we perform four independent evaluations using both GPT-4o and Claude-3.5, and take the average over four replications as the final evaluation score, to mitigate the variance caused by the inherent stochasticity of LLM outputs. To quantify the alignment between the two LLM models' evaluations, we conduct the Pearson correlation test on the relative rankings of summaries assigned by the two models, as this ranking metric captures whether the models agree on the comparative quality of summaries even if their absolute scores differ. We also do the correlation test on Shapley values to see the impact of using different LLM models for evaluation on the final Shapley values.

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

G.2 Performance under MAPE and MSE Metrics

For completeness, we evaluate the performance of the Cluster Shapley algorithm using alternative error metrics—MAPE and MSE—and compare it to other benchmark methods in Figure A12. For ease of comparison, we replicate the results from Figure 6 as panel (c) in Figure A12. As shown, Cluster Shapley consistently outperforms the benchmarks across all three metrics.



(a) MAPE Performance

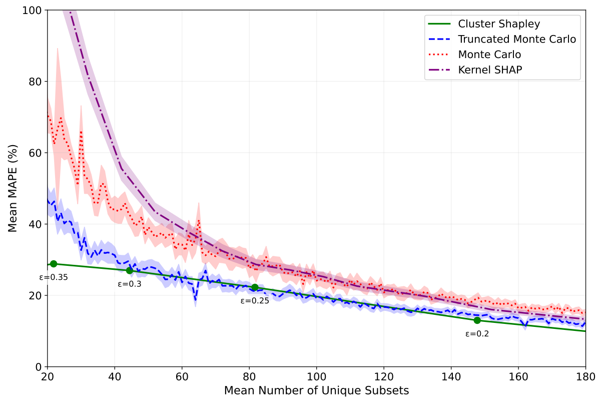
(b) MSE Performance

(c) MAE Performance

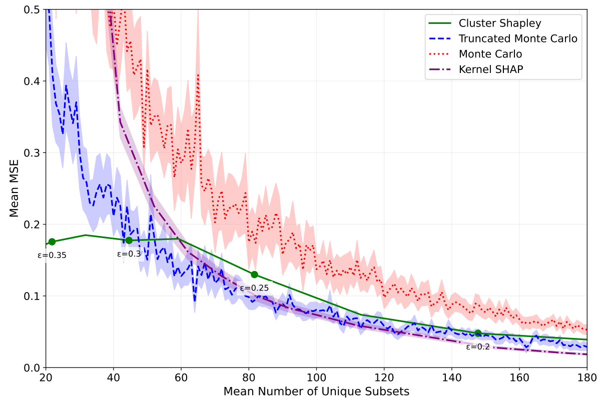
Figure A12: Performance comparison of algorithms under MAPE, MSE, and MAE measures. Subfigure (c) replicates Figure 6. The X-axis represents the number of unique subsets used by the algorithms, averaged across all test queries and reviews. The Y-axis represents the Mean error measures of the Shapley values, averaged across all test queries and reviews.

G.3 Cluster Shapley using the Standard DBSCAN

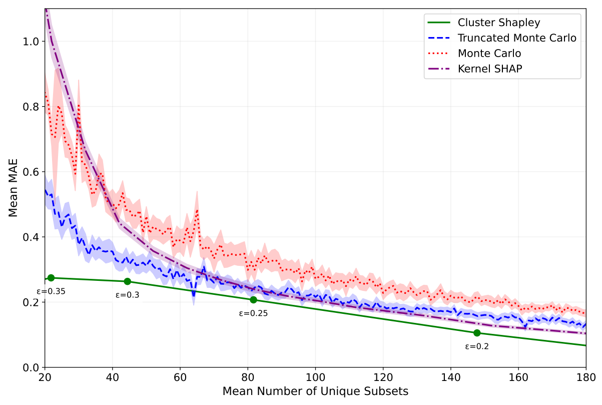
We now report the performance of the Cluster Shapley algorithm using the standard DBSCAN algorithm, instead of our proposed iterative variant, in Figure A13. As discussed in §4.2, the standard DBSCAN does not guarantee that all pairs of documents within the same cluster have embedding distances strictly smaller than ϵ, which may affect approximation quality.



(a) MAPE Performance



(b) MSE Performance



(c) MAE Performance

Figure A13: Performance comparison of Cluster Shapley using **standard DBSCAN** in Step 2. Note that the performance curves for the other three algorithms are identical to those in Figure A12. Each point on the Cluster Shapley curve corresponds to a different neighborhood radius ϵ'.

Comparing Figure A13 (which uses standard DBSCAN for clustering) with Figure A12 (which uses our proposed adaptive DBSCAN, i.e., Algorithm 2), we observe that Cluster Shapley achieves consistently better performance with adaptive clustering across all three approximation error measures. Moreover, when using more than 60 unique subsets, Cluster Shapley with standard DBSCAN performs significantly worse than both Truncated Monte Carlo and Kernel Shapley, highlighting the importance of enforcing tighter clustering constraints through the adaptive procedure.

References

Reddit. Reddit answers (currently in beta), 2025. URL https://support.reddithelp.com/hc/en-us/articles/32026729424916-Reddit-Answers-Currently-in-Beta. Accessed: 2025-04-19.