TokenShapley: Token Level Context Attribution with Shapley Value

Yingtai Xiao[1], Yuqing Zhu[1], Sirat Samyoun[1], Wanrong Zhang[1], Jiachen T. Wang[2], Jian Du[1],

[1]TikTok Inc., [2]Princeton University,

Abstract

Large language models (LLMs) demonstrate strong capabilities in in-context learning, but verifying the correctness of their generated responses remains a challenge. Prior work has explored attribution at the sentence level, but these methods fall short when users seek attribution for specific keywords within the response, such as numbers, years, or names. To address this limitation, we propose TokenShapley, a novel token-level attribution method that combines Shapley value-based data attribution with KNN-based retrieval techniques inspired by recent advances in KNN-augmented LLMs. By leveraging a precomputed datastore for contextual retrieval and computing Shapley values to quantify token importance, TokenShapley provides a fine-grained data attribution approach. Extensive evaluations on four benchmarks show that TokenShapley outperforms state-of-the-art baselines in token-level attribution, achieving a 11–23% improvement in accuracy.

1 Introduction

In context learning (ICL) (Brown et al., 2020; Dong et al., 2024) has emerged as a powerful capability of large language models (LLMs) (Achiam et al., 2023; Dubey et al., 2024; Guo et al., 2025). LLMs can perform a wide range of tasks (for example, question answering (Choi et al., 2018), solving math problems) by learning from the context information users provide. ICL is particularly valuable in scenarios where data collection and retraining are costly or impractical (for example, medical diagnosis (Wang et al., 2023)).

However, responses generated by large language models (LLMs) are not inherently grounded in truth, as these models can sometimes hallucinate (Zhang et al., 2023), producing factually incorrect or misleading information. This can lead to serious risks, particularly in high-stakes applications (e.g., healthcare diagnosis). As such, fact-checking

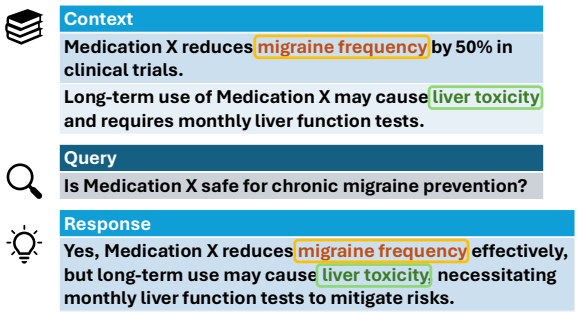

Figure 1: An example of context attribution. Here the response contains multiple information. Doing sentence-level attribution may provide incomplete evidence. A better way is to provide token-level attribution.

(Quelle and Bovet, 2024) and context attribution (Li et al., 2023a) is crucial to ensure that LLM responses are both interpretable and verifiable. Consider an LLM that recommends medication based on a medical database (Figure 1). A user may want to get evidence from key tokens, such as "migraine frequency" and "liver toxicity", that capture comprehensive evidence of both benefits and risks, which might be overlooked with a single attributed sentence in the context. By pinpointing the tokens behind a response and providing supporting evidence, users can more easily assess its reliability.

Approaches to attribution in LLMs generally fall into three categories (Li et al., 2023a). 1) **Direct Generated Attribution** (Sun et al., 2022; Weller et al., 2023). These methods prompt LLMs to generate self-attributions to improve response accuracy. However, researchers have found that LLMs often struggle to provide reliable evidence, particularly for domain-specific questions (Gravel et al., 2023). 2) **Post-Retrieval Answering** (Asai et al., 2023; Li et al., 2023b). These approaches involve additional model updates or training to improve response accuracy. However, the requirement for direct access to the model's parameters can introduce additional computational overhead. 3) **Post-Generation At-**

Findings of the Association for Computational Linguistics: ACL 2025, pages 3882–3894

tribution (Gao et al., 2022; Huo et al., 2023; Chen et al., 2023; Cohen-Wang et al., 2024). These methods utilize search engines, document retrieval systems, or surrogate models to retrieve evidence based on input questions and generated answers. They primarily attribute information at the sentence level, which lacks fine-grained attribution when a sentence contains multiple pieces of information. See Figure 1 for an example.

To address the limitations of existing attribution methods, we aim to propose a new attribution method that 1) Provides accurate evidence. 2) Eliminates the need for additional tuning with LLMs. 3) Enables fine-grained attribution. This motivates us to adopt the Shapley value (Shapley, 1953), which fairly attributes contributions by distributing credit among input components based on their marginal impact. Shapley values have been widely used for training data attribution (Jia et al., 2019a,b; Wang et al., 2024). In supervised learning tasks, when a K-nearest neighbors (KNN) model is used for attribution, Shapley values can be computed in polynomial time (Wang et al., 2024). However, applying KNN for attribution in LLMs remains challenging due to the vague notion of "labels" among tokens.

We observe that KNN-LM (Khandelwal et al., 2019; Li et al., 2024) enhances LLM generation by leveraging a KNN model, which stores (key, value) pairs for retrieval, where the key represents the prefix of a given token and the value is the token itself. Building on this idea, we introduce TokenShapley, a novel token-level attribution method that employs a KNN model specifically for attribution. The flowchart of TokenShapley is shown in Figure 2, with further details provided in Section 4.

In summary, our contributions are

- We propose a token-level attribution algorithm TokenShapley, allowing accurate, efficient and fine-grained attribution.
- We introduce an efficient strategy for exact token-level Shapley value calculation that leverages embedding-based retrieval and weighted KNN search, eliminating the need for Monte Carlo sampling.
- TokenShapley improves token-level attribution accuracy by 11–23% on Verifiability-Granular and QuoteSum, achieves perfect score on KV Retrieval, and raises Natural Questions precision by about 3.2%.

2 Related work

We provide additional related work not discussed in the previous section.

External Data Attribution Many prior work focus on tracing model's prediction back to the training data, this includes methods such as influence functions (Koh and Liang, 2017; Hammoudeh and Lowd, 2022; Grosse et al., 2023; Mlodozeniec et al., 2024). (Slobodkin et al., 2024; Sancheti et al., 2024) provides citations to sentences in the response from external documents.

Context Attribution (Cohen-Wang et al., 2024) learns a linear surrogate model that approximates the external context's behavior towards the generated response. They partition the context into sentences and learn a score of each sentence towards the response. (Park et al., 2023) propose the linear data modeling score, which measures the extent to which attribution scores can predict the effect of excluding a random subset of sources. (Phukan et al., 2024; Ding et al., 2024; Qi et al., 2024) utilize hidden-space embeddings of tokens in both the context and response to compute similarity scores. The tokens in the context with the highest scores are then identified as attributions for the corresponding tokens in the response.

3 Problem Setup and Background

In this section, we will start with the problem setup of context attribution. The notation used in this paper is listed in Table 6 in Appendix A .

Problem Setup We consider the in-context learning scenario in which a language model generates a response to a query based on a given context, such as in search engines like Bing Chat, which answers users' questions using information from news webpages. After seeing the response, a user may wonder: Is the generated response correct? Which part of the context led the model to produce this response? Our goal is to pinpoint the specific tokens from the context that drive the model's response, thereby enabling precise token-level attribution.

Formally, given a query q and a corpus of text data (context) with N tokens, $D = (z_1, \cdots, z_N)$, the language model generates a response R with T tokens $R = (\hat{z}_1, ..., \hat{z}_T)$. For each response token $\hat{z} \in R$, we aim to identify the most contributive tokens $z \in D$. To achieve this, we compute a score $\phi_{z \to \hat{z}}$ to quantify the contribution of z to $\hat{z}$. Tokens z with the highest scores $\phi_{z \to \hat{z}}$ are selected as the attribution for the token $\hat{z}$. We simply write ϕ_z

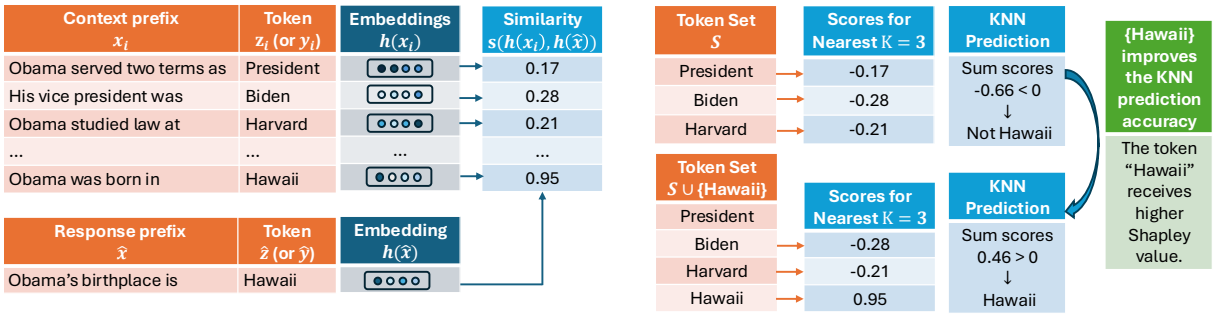

Figure 2: An illustration of TokenShapley. Given a response token $\hat{z}$, we first compute similarity scores using the datastore. To determine the Shapley value of a context token z_i, we select all possible subsets $S \subseteq D \backslash \{z_i\}$. We then run a KNN classifier on both S and $S \cup \{z_i\}$ to evaluate whether adding z_i improves the accuracy of predicting $\hat{z}$. If the accuracy increases, z_i is assigned a higher Shapley value.

when $\hat{z}$ is clear from the context. A widely used scoring function is the Shapley Value (Shapley, 1953), which we will introduce in the following section.

3.1 Shapley Value

The Shapley value (SV) (Shapley, 1953) is a classic concept from game theory to attribute the total gains generated by the coalition of all players. Formally, given a utility function $v(\cdot)$ (e.g., the model's prediction accuracy) and a dataset D, the Shapley value assigns a score to each data point $z \in D$ representing its contribution of z to $\phi_z(v)$.

To compute this contribution, one utility function is evaluated with the data point z present and another utility function with z excluded, denoted as $v(S \cup z) - v(S)$, where S is any possible subset of the dataset. The difference in utility functions represents the marginal contribution of z. The Shapley value is then the weighted average of these differences across all possible subsets S.

Definition 1. *Given a utility function $v(\cdot)$ and a dataset D with N data points, the Shapley value of a data point $z \in D$ is defined as*

$$\phi_z(v) = \frac{1}{N} \sum_{k=1}^{N} \sum_{\substack{S \subseteq D_{-z} \\ |S|=k-1}} \frac{v(S \cup \{z\}) - v(S)}{\binom{N-1}{k-1}} \tag{1}$$

Here $D_{-z} = D \backslash \{z\}$. When the utility function $v(\cdot)$ is clear from the context, we use ϕ_z as a shorthand to represent the Shapley value of data point z.

Equation 1 shows that computing the exact Shapley value for a data point z requires evaluating the utility function $v(\cdot)$ for all possible subsets $S \subseteq D \backslash \{z\}$. The number of evaluations grows exponentially with respect to N (the size of D), making the exact calculation computationally hard.

Fortunately, recent research (Ghorbani et al., 2022; Liang et al., 2020, 2021; Courtnage and Smirnov, 2021) has shown that the exact Shapley value for K-nearest neighbors (KNN)-based machine learning models can be computed with polynomial-time complexity.

3.2 KNN-Shapley

KNN-Shapley (Jia et al., 2019b) was originally introduced to address data attribution in the training set, which uses KNN to retrieve training samples. KNN-Shapley aims to determine the contribution of each training data point to the model prediction. In this section, we provide background information and highlight the challenges of applying KNN-Shapley to context attribution.

We use a K-nearest neighbors classifier to illustrate how KNN-Shapley works. Given a validation data point $\hat{z} = (\hat{x}, \hat{y})$, where $\hat{x}$ is the feature and $\hat{y}$ is the label, a KNN classifier predicts $\hat{y}$ via a weighted majority vote of the top-K neighbor's labels from the training set D. The weight assigned to a data point $z_i = (x_i, y_i) \in D$ is based on the similarity metric $s(\cdot, \cdot)$ between the feature x_i and the validation point feature $\hat{x}$.

(Jia et al., 2019b) defines the utility function as the likelihood of predicting the correct label $\hat{y}$. Let $x_j^{(S)}$ refer to j-th nearest feature in the set S with respect to $\hat{x}$, and $y_j^{(S)}$ be the label for $x_j^{(S)}$. Let $K^{(S)} = \min(K, |S|)$ be the number of voting points (in case S has less than K points). The utility function can be expressed as:

$$v(S;\hat{z}) = \frac{\sum_{j=1}^{K^{(S)}} s(x_j^{(S)}, \hat{x}) \mathbb{I}[y_j^{(S)} = \hat{y}]}{\sum_{j=1}^{K^{(S)}} s(x_j^{(S)}, \hat{x})} \quad (2)$$

When the weights $s(\hat{x}, x_j^{(S)})$ are 1, the utility function gives the portion of voting points that correctly predict the label $\hat{y}$. The utility function of S only depends on the top-K neighbors that are within S.

For weighted KNN, (Wang et al., 2024) demonstrates that by discretizing and slightly tweaking the utility function and using dynamic programming, it is possible to compute the exact Shapley values in $O(N^2)$ runtime. The utility function for weighted KNN is written as

$$v(S;\hat{z}) \quad (3)$$
$$= \mathbb{I}\left[\hat{y} \in \operatorname*{argmax}_{w \in \mathcal{C}} \sum_{j=1}^{K^{(S)}} s(x_j^{(S)}, \hat{x}) \mathbb{I}[y_j^{(S)} = w]\right]$$

Here $\mathcal{C}$ represents the space of classes, the number of labels is $|\mathcal{C}|$. This utility function measures whether the weighted majority vote predicts the true label $\hat{y}$. Compared to the equation 2, this utility function omits the normalization term.

The limitation of the KNN-Shapley method is that the utility function is primarily designed for supervised learning tasks. It relies on labeled data, where the correctness of a prediction serves as the utility function. However, in large language model generation task, defining an appropriate utility function remains an open challenge (Jia et al., 2019a; Wang et al., 2024), making it unclear how to compute an exact Shapley value in polynomial time. One of our key contributions is introducing a novel approach to creating "labels" within the generation process. Before presenting the details in Section 4, we first provide background on the KNN Language Model (Khandelwal et al., 2019), which gives the intuition behind our design.

3.3 KNN-LM

The primary challenge of applying KNN-Shapley in language models (LMs) is the absence of an integrated KNN mechanism. Can we leverage such mechanism to enhance model's generation capabilities? KNN-LM (Khandelwal et al., 2019; Li et al., 2024) offers a viable solution. It builds a KNN model on a datastore constructed from a text corpus and utilizes it to improve model performance.

Let $x_t = (z_1, \cdots, z_{t-1})$ be the prefix tokens and $y_t = z_t$ be the next token. The LM generates y_t based on the probability distribution $p(y_t|x_t)$. The KNN-LM enhances a pre-trained language model with a KNN retrieval mechanism, which processes a text collection in a single forward pass. The resulting context-target pairs are stored in a key-value datastore. During the inference, the datastore is queried to provide additional information to the LMs. Next we describe how this datastore is constructed.

3.3.1 Datastore

Let $h(\cdot)$ be the hidden state representation function that maps the prefix tokens $x_t = (z_1, \cdots, z_{t-1})$ to a LM hidden layer representation. For a token in the text corpus $y_t = z_t \in D$ and its prefix $x_t = (z_1, \cdots, z_{t-1})$, the (key, value) pair (k_t, v_t) is set such that the key k_t is the hidden layer representation $h(x_t)$ and the value v_t is the target token y_t. The datastore consists of all key-value pairs generated from the dataset D.

$$(\mathcal{K}, \mathcal{V}) = \{(h(x_t), y_t) | z_t \in D\} \quad (4)$$

3.3.2 Inference

During inference, given the prefix tokens $\hat{x}$, the model generates the target token according to $p_{LM}(\hat{y}|\hat{x})$. It also gives the hidden state representation $h(\hat{x})$. The model queries the datastore with $h(\hat{x})$ to retrieve its K-nearest neighbors $\mathcal{N}$ according to a similarity function $s(\cdot, \cdot)$.

$$p_K(\hat{y}|\hat{x}) \propto \sum_{\substack{(h(x_t), y_t) \\ \in \mathcal{N}}} s(h(x_t), h(\hat{x})) \mathbb{I}[y_t = \hat{y}] \quad (5)$$

The similarity function used is the RBF kernel function, $s(h, h') = \exp(-\gamma \|h - h'\|_2^2)$, where $\gamma > 0$ is a parameter and h, h' are two vectors. The final KNN-LM distribution is a weighted combination of the nearest neighbor distribution p_{KNN} and the model distribution p_{LLM} with a parameter $0 \leq \lambda \leq 1$, $p(y|x) = \lambda p_K(y|x) + (1 - \lambda) p_{LM}(y|x)$.

We observe that the probability distribution in Equation 5 is closely related to the utility function in Equations 3, as they both measure the likelihood of predicting the label $\hat{y}$ with the KNN model. This motivates us to create a KNN model in the LM generation. In this setting, we say the (feature, label) pair for a data point $z_t \in D$ is a (prefix, token) pair which is represented by (x_t, y_t). We

can then apply the weighted KNN-Shapley method using a utility function similar to Equation 3. We provide the details of the algorithm in the next section.

4 Algorithm

The goal of context attribution is to identify the most contributive sources within the context D that affect the model's response R. Specifically, we aim to assign scores to each token in the context $z_i \in D$, reflecting its contribution to a given token in the response $\hat{z}_t \in R$.

Recall that, in order to run the KNN-Shapley value algorithm which gives exact Shapley values in polynomial run time, we need to design a KNN model in language model generation. In Section 4.1 we follow the design of KNN-LM to create a datastore with (key, value) pairs for the KNN model. Then in Section 4.2, we design the utility function $v(\cdot)$ for efficient KNN-Shapley value. In Section 4.3, we briefly review the core technique used in (Wang et al., 2024). Lastly, we show how to accumulate Shapley values for attribution from tokens to tokens. Algorithm 1 shows the pseudo-code for the proposed TokenShapley algorithm.

Algorithm 1 TokenShapley

Require: Key-value data store cache $\mathcal{C}$. Embedding model $h(\cdot)$. Context $D = (z_1, \cdots, z_m)$. Query q. Response $R = (\hat{z}_1, \cdots, \hat{z}_T)$. Token $\hat{z}_t \in R$ of interest.

Ensure: Shapely value of each token z_i in context D w.r.t. token $\hat{z}_t$: $\phi_{z_i \to \hat{z}_t}$.

1: Get the "feature" $\hat{x}_t \leftarrow q + R[1:t]$.
2: Get the "label" $\hat{y}_t \leftarrow \hat{z}_t$.
3: Get the hidden representation $h(\hat{x}_t)$.
4: Search in the data store cache $\mathcal{C}$ using $h(\hat{x}_t)$ as the key to find the nearest K neighbors $(x_1, \cdots, x_K)$.
5: Calculate the utility function $v(\cdot)$ according to Equation 7.
6: Run Weighted KNN Shapley algorithm to get the Shapley value for z_i (w.r.t. token $\hat{z}_t$).
7: **return** $\{\phi_{z_i \to \hat{z}_t}\}_{i=1}^N$.

4.1 Build Datastore

Suppose there are N tokens in the context $D = (z_1, \cdots, z_N)$. For the t-th token z_t, the prefix of z_t is $x_t = (z_{t_0}, z_{t_0+1}, \cdots, z_{t-1})$. Here t_0 is the starting index for the prefix, for example, we can set t_0 as the index for the starting token at the current sentence of z_t. In this setting, we say a data point $z_t \in D$ has a (feature, label) pair which is represented by the (prefix, token) pair $z_t \to (x_t, y_t)$.

We use the encoder $h(\cdot)$ of the embedding model to encode each prefix sentence x_t into a hidden states $h(x_t)$. Let y_t be the next token following x_t in the context D. Given a token $z_t \in D$, we define the key-value pairs in datastore as below

$$(\mathcal{K}, \mathcal{V}) = \{(h(x_t), y_t) | z_t \in D\} \tag{6}$$

The size of the datastore is proportional to the total number of tokens within D.

4.2 Define Utility Function

During inference, the language model outputs the next token distribution $f_{LM}(\hat{y}_t|\hat{x}_t)$ along with the hidden state $h(\hat{x}_t)$. We use $h(\hat{x}_t)$ as a query to search the datastore $(\mathcal{K}, \mathcal{V})$ and retrieve the K-nearest neighbors based on the distance in the hidden space. Next, we apply weighted KNN-Shapley (Wang et al., 2024) (see Sec 3.2) to calculate Shapley value for selected neighbors. This method relies on evaluating the utility function of all possible subset S within $(\mathcal{K}, \mathcal{V})$.

$$v(S; \hat{z}_t) \tag{7}$$
$$= \begin{cases} 1, & \sum_{j=1}^{K^{(S)}} s(h(\hat{x}_t), h(x_j^{(S)})) \mathbb{I}[y_j^{(S)} = \hat{y}_t] \\ & \geq \sum_{j=1}^{K^{(S)}} s(h(\hat{x}_t), h(x_j^{(S)})) \mathbb{I}[y_j^{(S)} \neq \hat{y}_t] \\ 0, & otherwise. \end{cases}$$

The utility function evaluates whether the KNN predictor correctly predicts the generated token $\hat{y}_t$. If the majority vote says the label should be $\hat{y}$, the utility function will be 1. If the majority vote favors the label (not $\hat{y}$), the utility function will be 0. It's a special case of Equation 3, here it's a binary classification problem. The similarity function used is the RBF kernel function, $s(h, h') = \exp(-\gamma \|h - h'\|_2^2)$, where $\gamma > 0$ is a parameter and h, h' are two vectors. As a special case, when $K = 1$, the utility function is

$$v(S; \hat{z}_t) = \mathbb{I}[y_1^S = \hat{y}_t] \tag{8}$$

The utility function 8 gives value 1 if and only if the nearest data point to $\hat{z}_t \to (\hat{x}_t, \hat{y}_t)$ is $z_1^{(S)} \to (x_1^{(S)}, y_1^{(S)})$ and the token $y_1^{(S)}$ is exactly the same as $\hat{y}_t$. Note that we only need to calculate the KNN once for each label $\hat{y}_t$, if S contains no data points selected by the KNN, the utility function $v(S; \hat{z}_t)$ will always be 0, so we don't need to update it.

4.3 Calculate Shapley Values

In this section, we briefly describe how to calculate the Shapley value using the algorithm in (Wang et al., 2024) to make it self-contained.

Lemma 2. *(Wang et al., 2024) For any data point $z_i \in D$ and any subset $S \subseteq D \setminus \{z_i\}$, the marginal contribution has the expression as follows:*

$$v(S \cup \{z_i\}) - v(S) \tag{9}$$
$$= \begin{cases} 1 & \text{if } y_i = \hat{y}, \text{Cond}_{KNN}, \text{Cond}_{0\to1} \\ -1 & \text{if } y_i \neq \hat{y}, \text{Cond}_{KNN}, \text{Cond}_{1\to0} \\ 0 & \text{Otherwise} \end{cases}$$

Here, Cond_{KNN} is the condition when z_i is within the K nearest neighbors of $\hat{z}$ among $S \cup z_i$. If z_i is not one of the neighbors, adding z_i to the set S will have no effect on the utility function, and the difference will be 0. $\text{Cond}_{0\to1}$ is the condition when adding z_i changes the majority vote in Equation 7 from a label (not $\hat{y}$) to $\hat{y}$. This can only happen when z_i's token y_i is the same as the label, i.e., $y_i = \hat{y}$. Similarly, $\text{Cond}_{1\to0}$ is the condition when adding z_i changes the majority vote in Equation 7 from label $\hat{y}$ to a different label (not $\hat{y}$). This can only happen when z_i's token differs from the label, i.e., $y_i \neq \hat{y}$. The exact expressions for these conditions are provided in Appendix B.

Given the expression in Equation 9, the Shapley value calculation can be efficiently computed using the following theorem.

Theorem 3. *(Wang et al., 2024) Let $\mathcal{G}_{i,\ell}$ denote the count of subsets $S \subseteq D \setminus \{z_i\}$ of size ℓ that satisfy (1) Cond_{KNN}, and (2) $\text{Cond}_{0\to1}$ if $y_i = \hat{y}$, or $\text{Cond}_{1\to0}$ if $y_i \neq \hat{y}$. Then for a weighted KNN binary classifier using the utility function $v(S; \hat{x})$ in Equation 7, the Shapley value of a data point z_i can be expressed as:*

$$\phi_{z_i\to\hat{z}}(v) = \frac{2\mathbb{I}[y_i = \hat{y}] - 1}{N} \sum_{\ell=0}^{N-1} \frac{\mathcal{G}_{i,\ell}}{\binom{N-1}{\ell}} \tag{10}$$

The computation of all $\mathcal{G}_{i,\ell}$ can be done in $O(N^2)$ time complexity, and the time complexity for calculating all Shapley values is $O(N^2)$.

The computation of all $\mathcal{G}_{i,\ell}$ is a counting problem that can be efficiently solved using dynamic programming. From Equation 9 and Cond_{KNN}, we observe that if z_i is not among the K nearest neighbors, its contribution is zero. Therefore, we can further enhance efficiency by selecting only the M nearest neighbors and updating the Shapley value exclusively for the data points within these neighbors ($K < M < N$).

4.4 Accumulate Attribution Scores

In the previous sections, we demonstrated how to compute the Shapley value for attributing from a single context token z_i to a response token $\hat{z}$. A natural extension is to determine the attribution from a set of context tokens $z_i \in S$ (e.g., a sentence in the context) to a set of response tokens $\hat{z} \in \hat{S}$ (e.g., the entire response). This can be achieved by summing the individual Shapley value scores.

Step 1 Recall that $\phi_{z_i\to\hat{z}}(v)$ measures the contribution of a data point $z_i \in D$ for a specific token $\hat{z}$. To compute the Shapley value over the token set $\hat{S}$ of interest, we leverage the linearity property of the Shapley value. Specifically, this is done by summing the individual Shapley values for each validation point. $\phi_{z_i\to\hat{S}}(v) := \sum_{\hat{z}\in\hat{S}} \phi_{z_i\to\hat{z}}(v)$.

Step 2 We sum up the Shapley values in the context token sets S to get the attribution from S to $\hat{S}$. $\phi_{S\to\hat{S}}(v) := \sum_{z_i\in S} \phi_{z_i\to\hat{S}}(v)$. This gives us the following formula, $\phi_{S\to\hat{S}}(v) := \sum_{z_i\in S} \sum_{\hat{z}\in\hat{S}} \phi_{z_i\to\hat{z}}(v)$.

5 Experiments

In this section, we compare TokenShapley with several baselines across four datasets. We use a single A100 GPU with 80 GB memory for experiments. For parameters in TokenShapley, we set $K = 1$ and $M = 10$ in all experiments.

5.1 Baselines

We consider the following baselines on model explanations and attributions.

Hidden Space Similarity (HidSim for short) (Phukan et al., 2024) measures similarity by comparing hidden space embeddings of tokens in the context and response. For each token in the response, the most similar tokens in the context are identified as its attribution.

Text Embedding Similarity (EmbSim for short): We use a "text-embedding-3-small" model from OpenAI to generate embeddings for each source and the generated response. The cosine similarity between the source embedding and the response embedding is used as the attribution score for the source.

ContextCite (ConCite for short) (Cohen-Wang et al., 2024) trains a linear surrogate model to trace

model responses back to their contexts. It assigns a weight to each source (i.e., sentence) as an attribution score, where higher scores indicate greater contributions to the model's response. ConCite is primarily designed for sentence-level attribution, as training a linear surrogate model for each source is computationally expensive for token-level attribution.

5.2 Open Source LLMs

We list the three open-source large language models used in the experiment. When the context is clear, we omit model parameters—for instance, 'Llama-3' refers to Llama-3-8B as used in the experiments. 1) **Llama-3-8B**. The Hugging Face model name is "meta-llama/Meta-Llama-3.1-8B-Instruct". 2) **Mistral-7B**. The Hugging Face model name is "mistralai/Mistral-7B-Instruct-v0.3". 3) **Yi-6B**. The Hugging Face model name is "01-ai/Yi-6B-Chat".

5.3 Datasets

We use five datasets for evaluation.

QuoteSum (Schuster et al., 2023). The context comprises multiple passages, with the answer to the question provided. Several key words are manually marked and labeled with the index of the passage from which their attribution originates.

Verifiability-Granular (Phukan et al., 2024). The context consists of a list of passages, along with a given question and answer. Several key tokens are manually marked, and the task is to identify the passage most relevant to these tokens. The ground truth passage is also manually labeled.

KV Retrieval (Liu et al., 2024). The context is a sequence of (key, value) (randomly generated), and the query is to find the value corresponding to a key, the task is to just search and recall information from the context.

Natural Questions (Kwiatkowski et al., 2019). In this dataset, the context is a Wikipedia page, a question is asked and a labeled answer is given. We use the labeled answer (original text from the Wiki page) as the true attribution source. We select answers that consist solely of text (marked as paragraphs in the HTML tags).

CNN Dailymail (Nallapati et al., 2016). The Dataset is an English-language dataset containing news articles as written by journalists at CNN and the Daily Mail. We prompt the language model to briefly summarize the news article.

5.4 Metric

For the first three datasets, we use prediction accuracy as the evaluation metric, since ground-truth attribution labels are available. For the Natural Questions dataset, we report precision, recall, and F1 score. For the CNN/DailyMail dataset, we adopt the log probability drop metric (Cohen-Wang et al., 2024), which measures the decrease in log probability of generating the same response when the attributed sentences are removed from the context.

5.5 Experiment on QuoteSum

QuoteSum	Llama-3	Mistral	Yi
HidSim *	-	89.95	89.24
HidSim	83.91	80.73	75.59
TokenShapley	**92.51**	**92.20**	**91.27**
EmbSim with OpenAI text embedding			53.38

Table 1: Attribution Accuracy on QuoteSum dataset with 1319 examples. HidSim * quoted the results from (Phukan et al., 2024).

In our experiments on the QuoteSum dataset, attribution accuracy varied noticeably across the baselines. Our TokenShapley method consistently outperformed the others, scoring over 91% on all models. TokenShapley works well with QuoteSum because the dataset is designed with clearly marked short answers in the summaries, where each answer is tied to a specific passage via bracketed indices. Our method focuses on individual tokens that form these short answers, and results in higher performance. We also observe that HidSim suffer from slight misalignments in token boundaries and hidden state similarities, resulting in lower accuracy. EmbSim's poorer performance arises from the differences in embedding spaces between the response and source, which makes it less sensitive to detailed token-level information.

5.6 Experiment on Verifiability-Granular

The results in Table 2 show that TokenShapley outperforms both HidSim and EmbSim across all models. TokenShapley achieves the highest accuracy, with improvements of 4.5% over HidSim and 10.7% over EmbSim for Llama-3, and similarly for other models. This shows that token-level attribution is more effective for this dataset, as it better captures fine-grained relationships between the response and passages. On the other hand, Em-

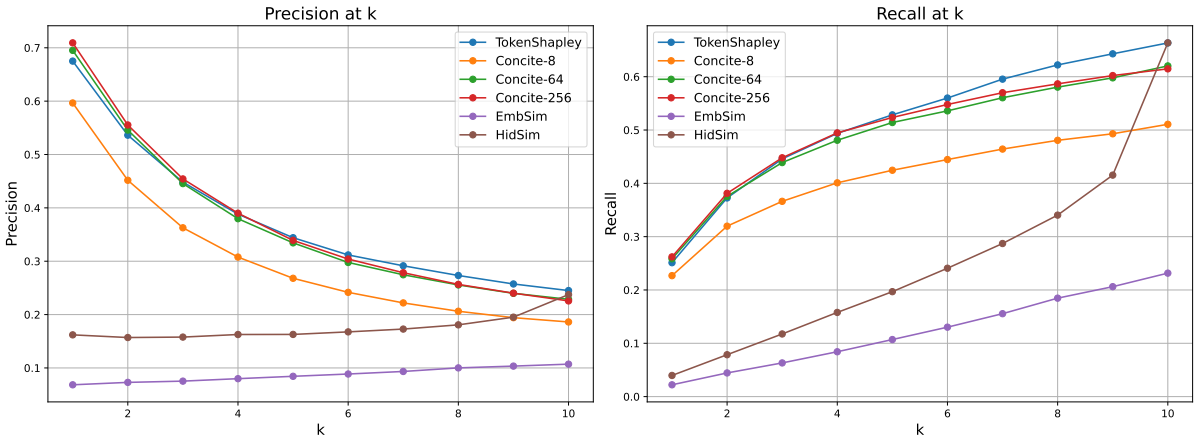

Figure 3: Precision and recall of top-k attributed sentences compared with ground truth attribution. Precision is defined as (number of match) / (k), recall is defined as (num of match) / (number of ground truth sentences).

VeriGran	Llama-3	Mistral	Yi
HidSim *	-	77.71	77.61
HidSim	70.77	71.28	61.54
TokenShapley	**82.23**	**84.77**	**84.77**
EmbSim with OpenAI text embedding			70.05

Table 2: Attribution Accuracy on Verifiability-Granular dataset with 197 examples. HidSim * quoted the results from (Phukan et al., 2024).

bSim's sentence-level approach is less sensitive to the detailed contextual relevance, and thus results in lower accuracy. HidSim, despite providing consistent results doesn't fully leverage the power of token-level analysis. Our intuition behind this is because it relies on average token representations rather than token-by-token analysis, which is overcome by our method.

5.7 Experiment on KV Retrieval

KVData	Llama-3	Mistral
ConCite (β=8)	82.80	80.80
ConCite (β=64)	100	100
HidSim	90.98	73.35
TokenShapley	**100**	**100**
EmbSim OpenAI text embedding		57.81

Table 3: Attribution Accuracy on KV Retrieval dataset with 500 examples. β is the number of random samples used for attribution in ConCite .

The results in Table 3 show that ConCite with β=64 and TokenShapley both reached perfect accuracy for both Llama-3 and Mistral. HidSim performed moderately well. In contrast, the off-the-shelf text embedding method (EmbSim) only achieved 57.81% accuracy. We exclude the Yi model from our evaluation due to its 4K context length limit, while the dataset requires approximately 5K tokens. This dataset is a simple task for non-LLM algorithms because a naive exact match can achieve 100% accuracy. We add this experiment to check if an attribution algorithm can also get 100% accuracy. But we can see that similarity based methods fail to find all the true values.

5.8 Experiment on Natural Question

Figure 3 shows the performance of various attribution methods on the Natural Questions dataset. 1000 examples are selected from the dataset. Overall, TokenShapley performs similarly well with ConCite when $k < 5$ and it performs consistently better when $k \geq 5$, which is likely due to its token-level attribution that better captures subtle contributions in complex documents. In contrast, methods like HidSim and EmbSim rely on overall hidden state or embedding similarities, and thus it missed fine-grained token contributions. Moreover, we see that TokenShapley works similarly with ConCite , yet it improves performance when more sentences are selected, indicating its advantage in capturing broader contextual information. Table 4 reports the precision, recall, and F1 scores for various methods when $k = 5$. TokenShapley outperforms the other baselines, indicating its superior effectiveness in identifying the correct attribution.

NQ	Precision	Recall	F1
ConCite -8	0.268	0.425	0.329
ConCite -64	0.334	0.514	0.405
ConCite -256	0.339	0.523	0.412
HidSim	0.163	0.197	0.178
EmbSim	0.084	0.107	0.094
TokenShapley	**0.344**	**0.528**	**0.417**

Table 4: Precision, recall and F1 on Natural Question dataset with 1000 examples. 8, 64, 256 are the number of random samples used for attribution in ConCite . Here we select top 5 candidates, $k = 5$.

CNN Dailymail	Llama-3	Mistral
ConCite (β=8)	0.88	0.91
ConCite (β=64)	**1.38**	**1.48**
HidSim	0.72	0.30
TokenShapley	1.01	1.33
EmbSim OpenAI text embedding		0.46

Table 5: Log Probability Drop on CNN Dailymail dataset with 1000 examples. β is the number of random samples used for attribution in ConCite .

5.9 Experiment on CNN Dailymail

Note that ConCite directly optimizes the log probability drop metric, which explains its strong performance when $\beta = 64$. As shown in Table 5, TokenShapley achieves comparable results to ConCite and even outperforms ConCite with $\beta = 8$. In contrast, HidSim results in much smaller drops (0.72 for Llama-3 and 0.30 for Mistral), and the off-the-shelf OpenAI text embedding similarity (EmbSim) performs poorly, with a drop of only 0.46. These results indicate that TokenShapley effectively identifies truly influential context, whereas simple similarity-based methods fail to capture attribution quality.

6 Discussion on Computational Cost

Our method involves two main computational components: (1) offline datastore construction, and (2) runtime KNN-Shapley inference.

Datastore Construction. This step is performed once and offline. For each token in the context dataset, we cache its prefix embedding, resulting in a datastore whose size is linear in the number of context tokens. For example, constructing the datastore for the NaturalQuestions dataset requires approximately 100KB of memory per example (computed as embedding dimension 768 × number of tokens). Importantly, we only store one Wikipedia page per question, rather than the full Wikipedia corpus, which greatly reduces the memory footprint. **Runtime Inference.** At inference time, the KNN-Shapley computation retrieves top-K neighbors from the datastore to assess token-level attribution. When $K = 1$, the computational complexity is $\mathcal{O}(n)$, where n is the number of tokens in the input context. In general, for $K > 1$, the complexity becomes $\mathcal{O}(n^2)$, though n remains small in practice (e.g., <512 tokens). We further reduce overhead by precomputing and storing datastore embeddings.

Given the small per-example memory cost and the offline nature of the datastore construction, the overall computational and memory overhead of our method is minimal. In addition, the modest datastore size allows it to be stored efficiently in memory or on low-cost cloud storage, making our approach practical for real-world applications.

7 Conclusion

In this work, we introduced TokenShapley, a novel token-level attribution method that integrates Shapley value-based data attribution with KNN-based retrieval, leveraging a precomputed datastore for efficient contextual retrieval. By computing Shapley values to quantify token importance, TokenShapley enables a more precise and interpretable attribution of language model outputs.

8 Limitations

One limitation of TokenShapley is the additional computational overhead required to create the datastore. However, this process only needs to be performed once and can be done offline. Future research may explore ways to enhance the efficiency of datastore creation. Another limitation is that the current method does not extend to text-to-image generation. TokenShapley relies on exact token-to-token matching, but identifying such correspondences in images remains an open challenge. We leave this for future work. One potential risk of TokenShapley is that datastore creation may expose user information, potentially compromising privacy. Future work should focus on developing privacy-preserving techniques for context attribution.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jifan Chen, Grace Kim, Aniruddh Sriram, Greg Durrett, and Eunsol Choi. 2023. Complex claim verification with evidence retrieved in the wild. *arXiv preprint arXiv:2305.11859*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. 2024. Contextcite: Attributing model generation to context. *arXiv preprint arXiv:2409.00729*.

Christie Courtnage and Evgueni Smirnov. 2021. Shapley-value data valuation for semi-supervised learning. In *Discovery Science: 24th International Conference, DS 2021, Halifax, NS, Canada, October 11–13, 2021, Proceedings 24*, pages 94–108. Springer.

Qiang Ding, Lvzhou Luo, Yixuan Cao, and Ping Luo. 2024. Attention with dependency parsing augmentation for fine-grained attribution. *arXiv preprint arXiv:2412.11404*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Y Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. 2022. Rarr: Researching and revising what language models say, using language models. *arXiv preprint arXiv:2210.08726*.

Amirata Ghorbani, James Zou, and Andre Esteva. 2022. Data shapley valuation for efficient batch active learning. In *2022 56th Asilomar Conference on Signals, Systems, and Computers*, pages 1456–1462. IEEE.

Jocelyn Gravel, Madeleine D'Amours-Gravel, and Esli Osmanlliu. 2023. Learning to fake it: limited responses and fabricated references provided by chatgpt for medical questions. *Mayo Clinic Proceedings: Digital Health*, 1(3):226–234.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. 2023. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Zayd Hammoudeh and Daniel Lowd. 2022. Identifying a training-set attack's target using renormalized influence estimation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1367–1381.

Siqing Huo, Negar Arabzadeh, and Charles Clarke. 2023. Retrieving supporting evidence for generative question answering. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 11–20.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. 2019a. Efficient task-specific data valuation for nearest neighbor algorithms. *Proceedings of the VLDB Endowment*.

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

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Dongfang Li, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Ziyang Chen, Baotian Hu, Aiguo Wu, and Min Zhang. 2023a. A survey of large language models attribution. *arXiv preprint arXiv:2311.03731*.

Minghan Li, Xilun Chen, Ari Holtzman, Beidi Chen, Jimmy Lin, Wen tau Yih, and Xi Victoria Lin. 2024. Nearest neighbor speculative decoding for llm generation and attribution. *Preprint*, arXiv:2405.19325.

Xiaonan Li, Changtai Zhu, Linyang Li, Zhangyue Yin, Tianxiang Sun, and Xipeng Qiu. 2023b. Llatrieval: Llm-verified retrieval for verifiable generation. *arXiv preprint arXiv:2311.07838*.

Weixin Liang, Kai-Hui Liang, and Zhou Yu. 2021. Herald: An annotation efficient method to detect user disengagement in social conversations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3652–3665.

Weixin Liang, James Zou, and Zhou Yu. 2020. Beyond user self-reported likert scale ratings: A comparison model for automatic dialog evaluation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1363–1374.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Bruno Mlodozeniec, Runa Eschenhagen, Juhan Bae, Alexander Immer, David Krueger, and Richard Turner. 2024. Influence functions for scalable data attribution in diffusion models. *arXiv preprint arXiv:2410.13850*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*.

Anirudh Phukan, Shwetha Somasundaram, Apoorv Saxena, Koustava Goswami, and Balaji Vasan Srinivasan. 2024. Peering into the mind of language models: An approach for attribution in contextual question answering. *arXiv preprint arXiv:2405.17980*.

Jirui Qi, Gabriele Sarti, Raquel Fernández, and Arianna Bisazza. 2024. Model internals-based answer attribution for trustworthy retrieval-augmented generation. *arXiv preprint arXiv:2406.13663*.

Dorian Quelle and Alexandre Bovet. 2024. The perils and promises of fact-checking with large language models. *Frontiers in Artificial Intelligence*, 7:1341697.

Abhilasha Sancheti, Koustava Goswami, and Balaji Vasan Srinivasan. 2024. Post-hoc answer attribution for grounded and trustworthy long document comprehension: Task, insights, and challenges. *arXiv preprint arXiv:2406.06938*.

Tal Schuster, Adam D Lelkes, Haitian Sun, Jai Gupta, Jonathan Berant, William W Cohen, and Donald Metzler. 2023. Semqa: Semi-extractive multi-source question answering. *arXiv preprint arXiv:2311.04886*.

Lloyd S Shapley. 1953. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317.

Aviv Slobodkin, Eran Hirsch, Arie Cattan, Tal Schuster, and Ido Dagan. 2024. Attribute first, then generate: Locally-attributable grounded text generation. *arXiv preprint arXiv:2403.17104*.

Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. 2022. Recitation-augmented language models. *arXiv preprint arXiv:2210.01296*.

Jiachen T Wang, Prateek Mittal, and Ruoxi Jia. 2024. Efficient data shapley for weighted nearest neighbor algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 2557–2565. PMLR.

Sheng Wang, Zihao Zhao, Xi Ouyang, Qian Wang, and Dinggang Shen. 2023. Chatcad: Interactive computer-aided diagnosis on medical image using large language models. *arXiv preprint arXiv:2302.07257*.

Orion Weller, Marc Marone, Nathaniel Weir, Dawn Lawrie, Daniel Khashabi, and Benjamin Van Durme. 2023. " according to...": Prompting language models improves quoting from pre-training data. *arXiv preprint arXiv:2305.13252*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

A Table of Notation

B Details for Weighted KNN Shapley

We define the weights in Equation 7 in a more compact form,

$$\omega_j^{(S)}(\hat{x}) := \begin{cases} s(h(\hat{x}), h(x_j^{(S)})), & y_j^{(S)} = \hat{y} \\ -s(h(\hat{x}), h(x_j^{(S)})), & y_j^{(S)} \neq \hat{y} \end{cases} \tag{11}$$

Then Equation 7 becomes

$$v(S; \hat{x}_t) = \mathbb{I}\left[\sum_{j=1}^{K^{(S)}} \omega_j^{(S)} \geq 0\right] \tag{12}$$

Table 6: Table of Notation

D :	Dataset, i.e. the context.
S :	A subset of D.
$\mathcal{C}$:	Datastore cache.
z :	Data point in the dataset i.e. a token in the context.
(x, y):	(feature, label) pair for z.
$z \to (x, y)$:	The mapping from token to (feature, label) pair.
$\hat{z}, \hat{x}, \hat{y}$:	Token, feature, label in response.
$v(\cdot)$:	The utility function for Shapley value.
$\phi_z(v)$:	The Shapley value for a data point z.
$h(\cdot)$:	Hidden state function.
$s(\cdot, \cdot)$:	Similarity function.
N :	The number of points in D.
K:	Select the nearest K neighbors.
$K^{(S)}$:	$\min(K, \lvert S \rvert)$
$x_j^{(S)}$:	The j-th closest prefix to $\hat{x}$ in S
M :	The number of data points used for updating Shapley in each KNN.

For a data point $z_i = (x_i, y_i)$, let

$$\omega_i = s(h(\hat{x}), h(x_i)) \tag{13}$$

Then the three conditions in Equation 9 have the following expressions.

$$\begin{aligned} &\text{Cond}_{KNN} := z_i \text{ is within} \\ &K \text{ nearest neighbors of } \hat{x} \text{ among } S \cup \{z_i\} \end{aligned} \tag{14}$$

$$\text{Cond}_{0\to 1} := \tag{15}$$
$$\begin{cases} \sum_{z_j \in S} \omega_j^{(S)}(\hat{x}) \in [-\omega_i, 0), & \text{if } |S| \le K-1 \\ \sum_{j=1}^{K-1} \omega_j^{(S)}(\hat{x}) \in \left[-\omega_i, -\omega_K^{(S)}(\hat{x})\right), & \text{if } |S| \ge K \end{cases}$$

$$\text{Cond}_{1\to 0} := \tag{16}$$
$$\begin{cases} \sum_{z_j \in S} \omega_j^{(S)}(\hat{x}) \in [0, \omega_i), & \text{if } |S| \le K-1 \\ \sum_{j=1}^{K-1} \omega_j^{(S)}(\hat{x}) \in \left[-\omega_K^{(S)}(\hat{x}), \omega_i\right), & \text{if } |S| \ge K \end{cases}$$

C Implementation Details

We implemented the baseline method HidSim (Phukan et al., 2024) for text attribution following their paper. Table 7 presents the accuracy results for different parameters demonstrating this baseline's influence on model performance on Verifiability dataset. The method involved extracting hidden layer representation for attributing text spans. First, each relevant text passage, with questions and summaries, was combined, tokenized, and passed through the model to provide intermediate representations. We chose embeddings from the middle layer, as it provides a balanced representation of the syntactic and semantic information from the model. Second, for each span extracted from the summaries, we identify the annotated text passages and extract its indexes and texts. The indexes are used as ground truth, while texts were used to generate an embedding of the span using the same model. Next, we computed cosine similarity scores between these span embeddings and embeddings of the individual tokens within the passages. Then we identify the top-K tokens with the highest similarity score as anchor tokens, following the paper. Finally, for each anchor token, we created a window of tokens centering (having a size of W) on these anchor tokens. For each window, we compute the similarity score with respect to the span embedding and choose the window with maximum similarity. We use its index as the predicted location of the attributed text and compare it with the ground truth index extracted earlier.

Table 7: Experimental Results for Different Parameter Settings on Verifiability Dataset

Model	$K = 15$		$K = 25$	
	$W = 10$	$W = 15$	$W = 10$	$W = 15$
Llama-3	69.2	70.8	69.7	70.8
Mistral-7B	69.2	69.7	71.3	69.7
Yi-6B	61.0	60.5	61.5	60.0

It must be mentioned that the parameters top-k and window size are chosen by design choice, as the paper (Phukan et al., 2024) doesn't mention these parameters configurations. Given the middle layer provides a balanced representation of the texts, we chose a higher value of top-K to ensure sufficient anchor tokens are chosen. Moreover, we chose a value of moderate window size that ensures attributing the span correctly for all examples, including those having broader contexts. We tuned the parameters top-K and window size and investigated its performance on model accuracy. For example, increasing the window size from 10 to 15 tokens improved the Meta-llama model's accuracy by 2%, while increasing the top-K from 15 to 25 enhanced the performance by 1.5%. These results indicate that a broader selection of anchors can facilitate accurate attribution of the spans.

D Case Study

Context

The following is a list of the top National Football League (NFL) quarterbacks in wins. In the NFL, the quarterback is the only position that is credited with records of wins and losses.

Active quarterback Tom Brady holds the records for most wins with 220, most regular season wins with 195, and most postseason wins with 25, as of Week 16 of the 2017 NFL season.

Having played the entirety of his career with the New England Patriots, each of Brady 's win records also apply to wins with a single team.

Among retired players, the record for most wins is held by Peyton Manning with 200. In his final professional game, Manning set the then - record for wins, surpassing Brett Favre who retired with 199 wins. Other previous record - holders include John Elway (162), Fran Tarkenton (130), and Johnny Unitas (124). Otto Graham holds the record for the highest winning percentage with a minimum of 35 wins at. 788 (61 wins to 16 losses).

Query

Who had the most wins in the NFL?

Response

According to the given information, Tom Brady holds the record for the most wins in the NFL with 220 wins.

In this section, we present an example of attribution results produced by TokenShapley on the Natural Questions dataset. The context provides background information about the NFL player Tom Brady. The question asks, "Who had the most wins in the NFL?" and the response correctly states that Tom Brady had the most wins. We apply TokenShapley to identify the most influential sentences (evidence) in the context for generating the response. The top two sentences selected by TokenShapley are shown in blue: the first receives a Shapley score of 16.8, and the second receives a score of 2.83. The yellow highlighted text corresponds to the ground-truth evidence labels provided by the dataset. We observe that both the precision and recall are 0.5. Notably, the sentence "... Tom Brady holds the records for most wins with 220, ..." is correctly identified by TokenShapley , demonstrating its effectiveness. However, the sentence "... record for wins ..." also receives a non-negligible score. We hypothesize that this is due to the high surface-level similarity between "record for wins" and "most wins", which may lead to confusion. Future work may improve attribution precision by incorporating additional filtering mechanisms to reduce such spurious matches.