Efficient Token-Level Counterfactual Explanations for Retrieval-Augmented Generation

Yingli Zhou
Lyon 1 University, CNRS Liris
Lyon, France
yingli.zhou@univ-lyon1.fr

Angela Bonifati
Lyon 1 University, CNRS Liris & IUF
Lyon, France
angela.bonifati@univ-lyon1.fr

HuiZhong Wang
Lyon 1 University, CNRS Liris
Lyon, France
huizhong.wang@univ-lyon1.fr

ABSTRACT

Retrieval-augmented generation (RAG) grounds large language model (LLM) responses in external knowledge and is increasingly used in knowledge-intensive applications. Building reliable and interpretable RAG systems requires understanding which retrieved facts jointly determine a generated response. Existing RAG explanation methods, however, mainly rank individual tokens or sentences. They neither model how multiple tokens jointly influence the response nor identify the smallest intervention target set whose joint factual replacement changes the response. We call this operation a counterfactual token intervention and formulate the task as *Token-Level Counterfactual Explanations for RAG* (TCE). Solving TCE directly requires examining exponentially many token sets and rerunning the LLM for each one. We propose `ReFlow`, a two-stage framework that constructs a contribution graph to capture joint token influence and uses the residual maximum flow to evaluate a token set. Motivated by the observed connection between residual flow and answer changes, `ReFlow` employs a contract-priced residual-flow search to balance the number of intervention targets against remaining contribution. We prove that this formulation reduces the exponential search space to a linear number of candidates. Experiments against six baselines on eight RAG datasets and four mainstream open-source LLM families show that `ReFlow` achieves the best performance across all datasets and metrics. Besides RAG explanation, the identified tokens also support other downstream applications, such as efficient RAG attacks, effective RAG defenses, and RAG provenance.

PVLDB Reference Format:
Yingli Zhou, Angela Bonifati, and HuiZhong Wang. Efficient Token-Level Counterfactual Explanations for Retrieval-Augmented Generation. PVLDB, 20(1): XXX–XXX, 2027.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
Code, configurations, and auditable replacement records are available at https://github.com/JayLZhou/CausalityRAG.

1 INTRODUCTION

The rapid development of large language models (LLMs), such as GPT-5.6, Qwen3.7, and Claude Fable 5 [2, 46, 69], has transformed how users access and process information [24, 38]. Nevertheless, an LLM may produce incorrect or outdated responses when the required domain knowledge, recent information, or proprietary data is absent from its pre-training corpus [49].

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 20, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

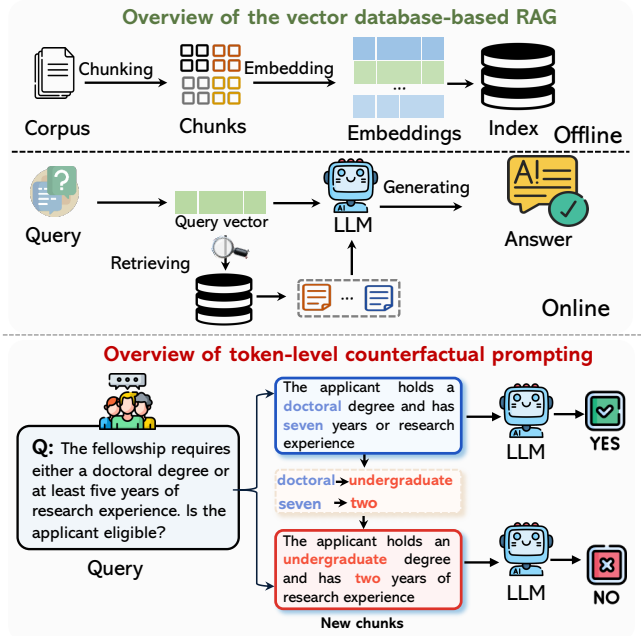


Figure 1: Vector-database-based RAG and a token-level counterfactual explanation.

Retrieval-augmented generation (RAG) [15, 18, 23, 25, 67, 73, 75] addresses this limitation by supplementing the LLM with external knowledge. Given a query q, a standard RAG pipeline [30] retrieves relevant chunks from an external corpus and feeds them, together with q, to the LLM to generate an answer, as shown in Figure 1. RAG is now widely used in knowledge-intensive domains, including healthcare [38, 63], finance [34, 45], and education [19, 64]. It also supports data management tasks such as NL2SQL [14, 31], data cleaning [32, 44], database configuration [29], DBMS diagnosis [55, 76, 77], and SQL rewriting [35, 56]. As RAG becomes an integral component of knowledge-intensive systems, building reliable and interpretable RAG systems is essential. To achieve this goal, we need to understand why a RAG system produces a particular response and which facts in the retrieved context the response depends on.

In explainable Artificial Intelligence, *counterfactual explanation* provides a natural way to understand the output of a black-box model. It asks how the input would need to change for the model to produce a different output [16, 52]. More recently, this idea has been extended to LLM-based systems through *counterfactual prompting*, which replaces selected tokens in a prompt with counterfactual alternatives, such as replacing *male* with *female*, and observes whether the generated response changes [71].

In this paper, we investigate counterfactual prompting in RAG systems and formulate the *Token-Level Counterfactual Explanation*

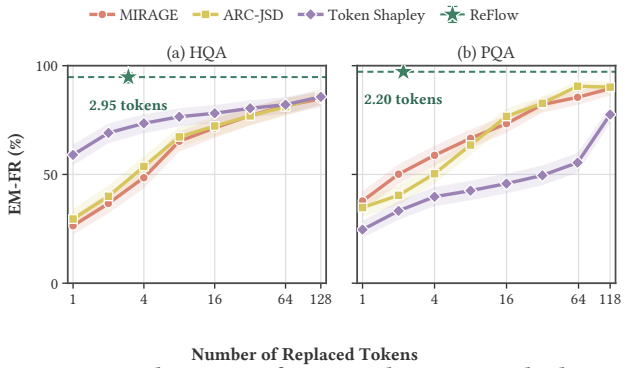


Figure 2: Token costs of RAG explanation methods.

for RAG (TCE) problem. Specifically, given a query q, its retrieved context, and the generated response, TCE seeks the smallest set of tokens in the retrieved chunks whose joint replacement with counterfactual alternatives changes the original response, while keeping the query and all other retrieved tokens unchanged. The resulting token set directly explains the original response by revealing the retrieved facts on which it depends. Figure 1 illustrates the TCE problem using a RAG system that assesses an applicant's eligibility for a fellowship. The retrieved context states that the applicant qualifies by either holding a doctoral degree or having seven years of research experience, leading the system to answer *Yes*. Jointly replacing *doctoral* with *undergraduate* and *seven* with *two* changes the answer to *No*. Thus, *doctoral*, *seven* forms the smallest counterfactual token set that changes the response, providing a concise explanation of the system's decision. Beyond explanation, TCE solutions also support token-efficient RAG attacks, effective RAG defenses, and compact RAG provenance.

• **Prior works.** To the best of our knowledge, the TCE problem has not been studied previously. Existing studies most closely related to TCE explain a generated RAG response by assigning importance scores to tokens or sentences in the retrieved chunks. `MIRAGE` and `ARC-JSD` are state-of-the-art (SOTA) token-level and sentence-level methods, respectively [33, 50], while Shapley-value-based methods estimate each token's average marginal contribution across different token coalitions [43, 54]. Despite their different scoring mechanisms, these methods ultimately evaluate individual tokens or entire sentences. However, a generated answer often depends on the joint effect of multiple tokens, which cannot be captured by simply aggregating their individual scores. As shown in Figure 2, these methods must replace substantially more top-ranked tokens with counterfactual alternatives to achieve comparable answer-flip rates. This indicates that their rankings fail to isolate the compact set of tokens jointly responsible for the response, resulting in less concise explanations of the RAG answer.

• **Challenges.** TCE is naturally a 0/1 black-box optimization problem whose candidate sets must be verified by rerunning the LLM [3, 10]. This raises three challenges.

(C1) *How to model token contributions to the generated response?* The LLM transforms and combines information across tokens, making both individual contributions and token interactions difficult to capture. (C2) *How to measure the effect of a token-set intervention?* After intervening on a token set, we need to quantify how much of its contribution to the original response remains. (C3) *How to efficiently find the minimum effective token set?* There are $2^n - 1$ nonempty sets, and testing each set requires another LLM execution, making exhaustive search infeasible.

• **Our contributions.** We present `ReFlow`, a two-stage framework for efficiently finding a small, verifiable counterfactual explanation. In the first stage, `ReFlow` constructs a contribution graph from one original LLM execution. It decomposes each attention output into messages exchanged between tokens, measures how each message contributes to the generated answer, and aggregates these contributions across Transformer layers. The resulting graph captures both individual token contributions and the paths through which multiple tokens jointly influence the answer (Section 4).

In the second stage, `ReFlow` evaluates a token set by the maximum contribution flow that remains in the graph after removing its selected tokens. Our calibration study shows that a smaller residual maximum flow is associated with a higher probability of changing the generated answer. Guided by this observation, `ReFlow` searches for a small token set that leaves little residual flow. Inspired by classical contract theory, whose foundations were recognized by the 2016 Nobel Prize in Economic Sciences [60], we assign each selected token a price λ and combine its modification cost with the residual flow. For a fixed λ, the exact optimum is obtained by one minimum-cut computation. We prove that, over all possible values of λ, at most $n + 1$ distinct token sets can become optimal. An exact divide-and-conquer algorithm finds all of them using at most $2n + 1$ maximum-flow computations, avoiding enumeration of the original $2^n - 1$ sets.

[zhou: Extensive experiments on eight RAG datasets spanning diverse domains and three mainstream open-source LLM families show that `ReFlow` achieves the best performance across all datasets. TCE also enables three downstream applications: RAG attacks, defenses, and provenance. Specifically, attacking only the tokens `ReFlow` identifies achieves a higher success rate with far fewer costs than SOTA attacks [36, 79]; preserving the sentences containing these tokens defends against poisoning better than SOTA defenses [58, 68]; and prompting the LLM with only these sentences serves as provenance, matching the accuracy of the full retrieved chunks with 10× fewer tokens.]

In summary, our main contributions are as follows:

- We define the novel TCE problem to address token-based counterfactual explanations for RAG systems, which have not been studied in the literature yet.
- We construct a contribution graph to model joint token influence and develop a contract-priced residual-flow search that efficiently identifies small candidate sets with theoretical guarantees, which we prove reduces the search space from exponential to linear.
- We conduct extensive experiments on eight datasets and three LLM families to demonstrate the effectiveness and effective of `ReFlow`. We also show that our problem supports other downstream applications, such as efficient RAG attacks, effective RAG defenses, and RAG provenance.

• **Organization.** Section 2 introduces the preliminaries and formally defines the problem. Section 3 presents the overall framework. Sections 4 and 5 describe the contribution graph and flow-based

search algorithm, respectively. Section 6 reports the experiments. Section 7 reviews related work, and Section 8 concludes the paper.

2 PRELIMINARIES AND PROBLEM FORMULATION

In this section, we review the key concepts of large language models (LLMs), the pipeline of retrieval-augmented generation (RAG), and counterfactual prompting. We then formalize the problem studied in this paper and position it against existing explanation methods.

2.1 Large Language Models

We introduce some fundamental concepts of LLMs, including LLM prompting and retrieval augmented generation (RAG).

▶ **LLM Prompting.** Instruction-tuned LLMs follow natural-language instructions and perform many tasks without task-specific parameter updates [47]. A prompt combines a user query with a system instruction that specifies the model's role, task, and output format. The LLM processes the rendered prompt and autoregressively generates a response.

▶ **Retrieval-Augmented Generation.** LLMs can produce unsupported or outdated statements despite encoding broad knowledge in their parameters [24]. RAG augments the model input with evidence retrieved from an external corpus before generation [30]. This pattern is supported by application frameworks such as LlamaIndex and LangChain [28, 39].

A typical RAG pipeline consists of an offline stage and an online stage [18]. During the offline stage, an external corpus is divided into short chunks. An embedding model, such as BGE-M3 [5], maps each chunk to a high-dimensional vector. After that, all embeddings of chunks are stored in a vector database for efficient online retrieving. During the online stage, given a user query q, the pipeline first encodes q and retrieves the top-k most relevant chunks from the index, denoted by $C = \{C_1, C_2, \ldots, C_k\}$. Finally, the retrieved chunks C, system prompt P, and query q are combined into a single prompt and fed into the LLM to generate answer $\mathcal{R}_q(C)$. Formally, we have $\mathcal{R}_q(C) := \texttt{LLM}(P, q, C)$.

2.2 Counterfactual Prompting

Counterfactual prompting examines how an LLM's response changes under interventions on selected input tokens [71]. Formally, a token-level counterfactual intervention replaces the selected tokens with counterfactual tokens while keeping the rest of the prompt unchanged. It asks whether the model would produce the same response if those tokens conveyed different information.

In RAG systems, retrieved chunks provide the external knowledge that an LLM uses to answer a query. The generated response often depends on specific information contained in these chunks. For example, an answer to a financial question may depend on a company name, date, monetary value, or transaction, while a medical answer may depend on a disease, symptom, treatment, or dosage. Thus, tokens in the retrieved chunks that express such information can play an important role in helping the LLM generate a correct response. To this end, if intervening on a set of retrieved tokens changes the response, those tokens provide a counterfactual explanation of the original response.

2.3 Problem Definition

In this subsection, we formally define the research problem studied in this paper. Given the retrieved chunks $C = \{C_1, \ldots, C_k\}$, let $\texttt{Token}(C) = \{t_1, \ldots, t_n\}$ denote the set of all tokens in C.

▶ **Counterfactual Intervention.** For each token $t_i \in \texttt{Token}(C)$, we use $\Psi(t_i)$ to denote its counterfactual token. The counterfactual token preserves the token's semantic type, syntactic role, and morphology while making its containing sentence express incompatible factual information. It is neither a deletion nor arbitrary noise. It also differs from a meaning-preserving paraphrase: replacing *Paris* with *the French capital* preserves the referenced entity, whereas replacing *Paris* with *London* changes the location. [zhou: Note that, in our paper, we construct a counterfactual token pool using an LLM and NLP tools, such that each selected token in C is associated with at least one valid counterfactual alternative. The details appear in Appendix A.1.]

Formally, for each token $t_i \in \texttt{Token}(C)$, the token-level *counterfactual intervention* operator is defined as $\mathrm{do}(t_i := \Psi(t_i))$. This means replacing t_i with $\Psi(t_i)$ while keeping all other tokens unchanged, analogous to using the *do*-operator to change a variable in causal analysis [48]. For a selected token set $T \subseteq \texttt{Token}(C)$, $\Psi(T)$ denotes applying counterfactual intervention to each token in T. We use $C[T \leftarrow \Psi(T)]$ to denote the chunk obtained by jointly applying these interventions. Based on this, we can formally define our research problem.

Problem 2.1: Token-Level Counterfactual Explanations for RAG

Given a system prompt P, a query q, retrieved chunks C, an LLM, and a counterfactual replacement operator Ψ, find

$$T^* \in \arg\min_{T \subseteq \texttt{Token}(C)} |T| \quad \text{s.t.} \quad \mathcal{R}_q(C[T \leftarrow \Psi(T)]) \not\equiv \mathcal{R}_q(C).$$

In our problem, a counterfactual intervention on T is successful if $\mathcal{R}_q(C[T \leftarrow \Psi(T)]) \not\equiv \mathcal{R}_q(C)$, where $\equiv$ denotes answer equivalence under the adopted evaluation criterion. We can view this condition as a *what-if* question: Would the same RAG system still produce the same response if the selected tokens conveyed different information? We seek a successful intervention with the fewest selected tokens; these tokens form a counterfactual explanation of the original response.

Given the ground-truth answer y_q, we determine whether an intervention is successful using a RAG evaluation metric f, such as F1-score, accuracy, or exact match. Specifically, the intervention is successful under f if

$$f\big(\mathcal{R}_q(C[T \leftarrow \Psi(T)]), y_q\big) \neq f\big(\mathcal{R}_q(C), y_q\big)\,. \tag{1}$$

Example 2.1. As shown in Figure 1, the answer *Yes* is supported by two independent facts: the applicant holds a doctoral degree and has seven years of research experience. Intervening on either highlighted token alone leaves one supporting fact unchanged, so neither intervention changes the answer. Jointly intervening on both tokens causes both requirements to fail and changes the answer to *No*. The pair is therefore the smallest successful intervention and forms the minimum counterfactual explanation.

2.4 0/1 Optimization Problem

In this subsection, we show that Problem 2.1 is an instance of 0/1 black-box optimization. For each token $t_i \in \texttt{Token}(C)$, let $z_i = 1$ select it as an intervention target and apply its admissible replacement, and let $z_i = 0$ leave it unchanged. Given $\mathbf{z} = (z_1, \ldots, z_n)$, let $T(\mathbf{z}) = \{t_i \in \texttt{Token}(C) \mid z_i = 1\}$ denote the intervention target set. The problem can then be written as

$$\begin{aligned} \min_{\mathbf{z}\in\{0,1\}^n} \quad & \sum_{i=1}^{n} z_i \\ \text{s.t.} \quad & \mathcal{R}_q(\mathcal{C}[T(\mathbf{z}) \leftarrow \Psi(T(\mathbf{z}))]) \not\equiv \mathcal{R}_q(C). \end{aligned} \tag{2}$$

This problem is challenging because finding the exact solution may therefore require examining all $2^n - 1$ nonempty token sets. Such an exhaustive search is infeasible in practice: even $n = 100$ yields approximately 1.27×10^{30} candidates.

2.5 Analysis of Existing Methods

Existing methods related to TCE mainly follow two directions: attribution-based RAG explanation and LLM-based counterfactual prompting. However, neither direction directly identifies the minimum set of retrieved tokens whose joint counterfactual intervention changes the generated response.

▶ **Attribution-based RAG explanation.** A straightforward approach is to assign an importance score to each token in the retrieved chunks and replace the highest-ranked tokens. Attention measures how strongly the generated response attends to each retrieved token [1], while gradient-based attribution measures how sensitive the response score is to that token.

More specialized methods have been developed for explaining RAG responses. `MIRAGE`, a SOTA token-level method, first compares the LLM's predictions with and without the retrieved chunks to identify response tokens that depend on the retrieved information. For each identified response token, it computes contrastive gradients with respect to the retrieved tokens and uses their gradient magnitudes as token-level importance scores [50]. `ARC-JSD`, a SOTA sentence-level method, temporarily removes each retrieved sentence and uses the LLM to rescore the previously generated response token by token. It measures sentence influence by summing the Jensen–Shannon divergence between the token-prediction distributions obtained before and after removing that sentence [33]. To apply `ARC-JSD` to TCE, we assign each sentence score to its constituent tokens and rank the resulting token scores. Besides, shapley-value-based methods additionally evaluate each token across different token coalitions and compute its average marginal contribution [43, 54]. Despite their different attribution mechanisms, these methods ultimately assign an individual importance score to each token or sentence. As a result, selecting the top-ranked tokens does not evaluate the selected set as a whole or optimize its joint effect on the generated response.

▶ **Counterfactual prompting.** Counterfactual prompting replaces selected tokens in a prompt with counterfactual alternatives and observes whether the generated response changes [71]. This provides a direct way to verify the effect of a given token replacement. However, existing counterfactual prompting methods do not determine which retrieved tokens in a RAG context must be replaced jointly. That is, applying counterfactual prompting alone to TCE would require testing different token subsets by repeatedly modifying the retrieved chunks and rerunning the LLM. For n retrieved tokens, this produces as many as $2^n - 1$ nonempty candidate sets and is therefore infeasible in practice.

▶ **Our goal.** In summary, what we desire is a method that not only effectively models individual token contributions and the joint contribution of multiple tokens to the generated response, but also measures the effect of a joint intervention and finds the minimum effective token set without requiring a large number of LLM executions or enumerating the $2^n - 1$ nonempty subsets.

3 OUR APPROACH

In this paper, we propose `ReFlow`, a novel **Re**sidual-**Flow**-based method for token-level counterfactual explanations in RAG. Given a query q, retrieved chunks C, and its response $\mathcal{R}_q(C)$. `ReFlow` contains two stages: *1) contribution graph construction* and *2) contract-priced residual-flow-based token identification.*

The first stage addresses the first challenge by constructing a contribution graph that models how information from retrieved tokens flows through the LLM and contributes to the generated response. Specifically, `ReFlow` records the information transmitted between token positions, measures how each message contributes to the response, and aggregates these contributions across Transformer layers. Each directed edge represents the direction of information flow, while its weight quantifies the contribution transmitted between two token positions. The resulting graph captures both individual token contributions and the joint influence of multiple tokens on the generated response.

In the second stage, we address the remaining challenges by first establishing the relationship between residual maximum flow and response changes: the lower the residual flow after removing a token set from the contribution graph, the more likely its counterfactual replacement is to change the generated response. We therefore use residual maximum flow to measure the effectiveness of a token set without repeatedly invoking the LLM, thereby addressing the second challenge. To address the third challenge, we develop a contract-priced residual-flow search that assigns a price to each selected token and balances the number of selected tokens against the residual flow. For any fixed price, the optimal token set can be obtained through a single maximum-flow computation. We prove that at most $n + 1$ distinct token sets can become optimal as the price varies. Finally, `ReFlow` verifies these candidates in increasing order of size by applying their counterfactual replacements and rerunning the LLM, returning the smallest set that changes the response (Section 5).

4 CONTRIBUTION GRAPH CONSTRUCTION

In this section, we introduce our contribution graph construction, which aims to model how retrieved tokens individually and jointly contribute to the generated response.

4.1 Forward Messages in a Transformer

Note that modern LLMs generate responses autoregressively through Transformer layers, where self-attention integrates information across different token positions [4, 62]. To this end, we use self-attention as the basis for our graph construction. During generation, information from a retrieved token may pass through multiple attention connections and layer-wise transformations before influencing an answer token. We therefore decompose each attention output

according to its source token positions and measure how much each source contributes to the generated answer. By aggregating these contributions across Transformer layers, we construct a graph that traces how retrieved information flows through the LLM during generation.

As discussed above, self-attention transfers information between tokens. To construct the contribution graph, we first briefly introduce this process. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N\times d}$ be the input representation matrix of N tokens, where $\mathbf{x}_i \in \mathbb{R}^d$ is the representation of the token at position i, and d is the model dimension. A self-attention head transforms $\mathbf{X}$ into the query, key, and value matrices: $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, $\mathbf{V} = \mathbf{X}\mathbf{W}_V$, where $\mathbf{W}_Q \in \mathbb{R}^{d\times d_q}$, $\mathbf{W}_K \in \mathbb{R}^{d\times d_k}$, and $\mathbf{W}_V \in \mathbb{R}^{d\times d_v}$ are learned projection matrices, with $d_q = d_k$. Consequently, $\mathbf{Q} \in \mathbb{R}^{N\times d_q}$, $\mathbf{K} \in \mathbb{R}^{N\times d_k}$, and $\mathbf{V} \in \mathbb{R}^{N\times d_v}$. We denote their i-th rows by $\mathbf{q}_i$, $\mathbf{k}_i$, and $\mathbf{v}_i$, respectively.

For two tokens at positions i and $j \le i$, the attention coefficient $\alpha_{i,j}$ measures the importance of token j to token i and is computed as

$$\alpha_{i,j} = \frac{\exp\left(\mathbf{q}_i \mathbf{k}_j^\top / \sqrt{d_k}\right)}{\sum_{u\le i} \exp\left(\mathbf{q}_i \mathbf{k}_u^\top / \sqrt{d_k}\right)}, \tag{3}$$

where $\mathbf{q}_i \mathbf{k}_j^\top$ measures the compatibility between tokens i and j, and $\sqrt{d_k}$ is the normalization factor. Modern LLMs use *causally masked self-attention* to prevent information leakage from tokens that have not yet been generated [62]. Specifically, when processing position i, all future positions $j > i$ are masked, so token i can attend only to tokens at positions $j \le i$. This masking induces a directed information flow from earlier tokens to later tokens. Let $\alpha_{i,j}$ denote the attention weight from position j to position i. We have $\alpha_{i,j} = 0$ for $j > i$.

In practice, each Transformer layer contains H heads that project the same input into different query, key, and value spaces. Their outputs are concatenated and mapped back to the model dimension d through an output projection. Let $\mathbf{W}_O^{(\ell,h)} \in \mathbb{R}^{d_v\times d}$ denote the corresponding output-projection block for head h at layer ℓ. We define the forward message transmitted from token j to token i as

$$\mathbf{m}_{j\to i}^{(\ell)} = \sum_{h=1}^{H} \alpha_{i,j}^{(\ell,h)} \mathbf{v}_j^{(\ell,h)} \mathbf{W}_O^{(\ell,h)}. \tag{4}$$

Thus, $\mathbf{m}_{j\to i}^{(\ell)} \in \mathbb{R}^d$ represents the information transmitted from token j to token i at layer ℓ after aggregating all attention heads. The complete attention output received by token i is

$$\mathbf{o}_i^{(\ell)} = \sum_{j\le i} \mathbf{m}_{j\to i}^{(\ell)}. \tag{5}$$

The forward message specifies what information is transmitted between two tokens. The next subsection measures how much this message contributes to the generated response.

4.2 Message Contributions

Section 4.1 defines $\mathbf{m}_{j\to i}^{(\ell)}$ as the information transmitted from token at position j to the projected attention output $\mathbf{o}_i^{(\ell)}$. The message describes what is transmitted, but it does not indicate whether that information supports the generated response. We next measure each message's contribution to the generated response.

▶ **Answer-support score.** Let the generated response contain m tokens: $\mathcal{R}_q(C) = (a_1, a_2, \ldots, a_m)$. For each answer token a_r, let $\zeta_r \in \mathbb{R}$ denote the logit that the LLM assigns to a_r at its generation step. We define the answer-support score as

$$S_q = \frac{1}{m} \sum_{r=1}^{m} \zeta_r. \tag{6}$$

A larger S_q means that the tokens in the generated response receive a larger average logit (taking the average prevents this value from increasing mechanically with response length).

▶ **Answer gradient.** For the projected attention output $\mathbf{o}_i^{(\ell)}$ in Equation (5), we compute

$$\mathbf{g}_i^{(\ell)} = \frac{\partial S_q}{\partial \mathbf{o}_i^{(\ell)}}. \tag{7}$$

The gradient $\mathbf{g}_i^{(\ell)}$ measures how a local change in the attention output at position i would change the score assigned to the generated response. It identifies which directions at the receiver matter for the answer, but it does not identify which source position supplies those directions.

▶ **Message contribution.** We combine the answer gradient with the forward message from Equation (4):

$$r_q^{(\ell)}(j,i) = \left\langle \mathbf{g}_i^{(\ell)}, \mathbf{m}_{j\to i}^{(\ell)} \right\rangle = \sum_{h=1}^{H} \alpha_{i,j}^{(\ell,h)} \left\langle \mathbf{g}_i^{(\ell)}, \mathbf{v}_j^{(\ell,h)} \mathbf{W}_O^{(\ell,h)} \right\rangle. \tag{8}$$

The value $r_q^{(\ell)}(j,i)$ is positive when the message transmitted from token j to token i increases S_q, and negative when it decreases S_q. Thus, the forward message captures the information transmitted between the two tokens, while the gradient measures whether this information supports the generated answer.

Why can $r_q^{(\ell)}(j,i)$ measure the contribution of a message? Consider removing the message $\mathbf{m}_{j\to i}^{(\ell)}$ from the attention output at token position i, while keeping all other incoming messages unchanged. Since $\mathbf{o}_i^{(\ell)}$ is the sum of these messages, the removal changes the attention output by $-\mathbf{m}_{j\to i}^{(\ell)}$. Let ΔS_q denote the resulting change in the generated-response score. Using a first-order approximation, we have

$$\Delta S_q \approx \left\langle \mathbf{g}_i^{(\ell)}, -\mathbf{m}_{j\to i}^{(\ell)} \right\rangle = -r_q^{(\ell)}(j,i). \tag{9}$$

Thus, when $r_q^{(\ell)}(j,i) > 0$, removing the message is expected to decrease the score of the generated response, indicating that the message makes a positive local contribution to that response. Since our contribution graph models how positive contributions propagate between token positions, we retain only messages satisfying $r_q^{(\ell)}(j,i) > 0$. The next subsection aggregates these message-level contributions into token-to-token edge capacities.

4.3 Constructing the Contribution Graph

Section 4.2 assigns a contribution score $r_q^{(\ell)}(j,i)$ to the message transmitted from token j to token i at layer ℓ. For each receiver token with at least one positive incoming contribution, we normalize the contribution of an incoming message $u \to v$ as

$$\omega_q(u,v) = \frac{[r_q(u,v)]_+}{\sum_{w\to v} [r_q(w,v)]_+}, \qquad [x]_+ = \max\{x, 0\}. \tag{10}$$

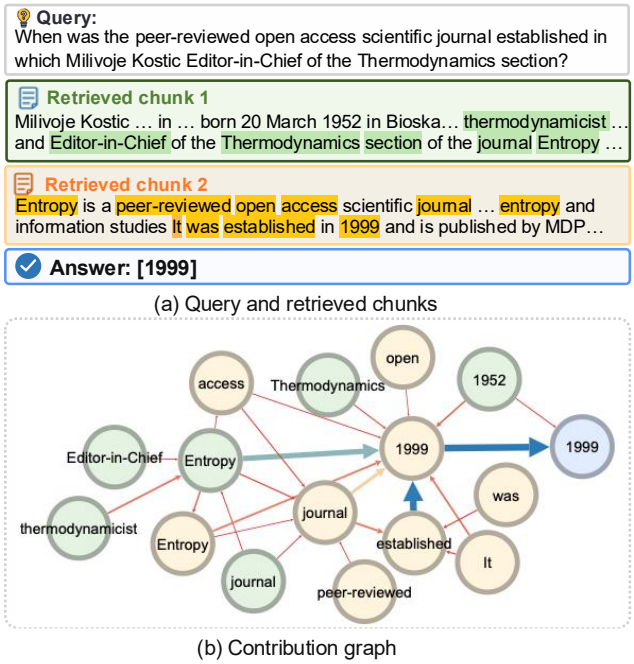


Figure 3: A real multi-hop contribution graph for a HotpotQA example. Edge width encodes its contribution.

Thus, messages that contribute more positively to token v receive a larger weight, while $\sum_{u \to v} \omega_q(u, v) = 1$. This step only normalizes the incoming message scores of the same receiver token.

Because the same token pair may exchange information in multiple Transformer layers, we then sum its normalized contributions across all L layers:

$$c_q(j, i) = \sum_{\ell=1}^{L} \omega_q^{(\ell)}(j, i). \tag{11}$$

Thus, $c_q(j, i)$ measures the total normalized contribution transmitted from token j to token i throughout the LLM execution.

▸**Contribution graph.** We retain every non-punctuation token in retrieved chunks as a graph node. The directed edge set contains all token pairs with positive aggregated contribution:

$$\mathcal{E}_q = \left\{(i, j) : c_q(i, j) > 0\right\}. \tag{12}$$

The resulting contribution graph with respect to query q is $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_q, \mathbf{c}_q(\cdot, \cdot))$, where the node set is partitioned as $\mathcal{V}_q = \texttt{Token}(q) \cup \texttt{Token}(C) \cup \texttt{Token}(\mathcal{R}_q(C))$. The three subsets contain tokens in the query, retrieved chunks and generated answer, respectively. The directed edge set $\mathcal{E}_q$ contains the positive token-to-token connections, and $\mathbf{c}_q(\cdot, \cdot)$ is their capacity function. Specifically, each edge $(i, j) \in \mathcal{E}_q$ has capacity $c_q(i, j)$ from Equation (11).

Note that if the positive contributions do not form a path from the retrieved chunks to the generated answer, we add one edge between the input- and answer-side tokens with the largest contribution weights, setting its capacity to the smaller of the two weights.

Example 4.1. Figure 3 illustrates the contribution graph for a HotpotQA query. The retrieved chunks identify the journal as *Entropy* and state that it was established in *1999*, leading the LLM to answer *1999*. For readability, the figure shows only the key evidence and strongest contribution paths. The route through *Entropy*, *established*, and *1999* illustrates how information from multiple retrieved chunks jointly supports the generated response.

4.4 Construction Algorithm and Cost

Algorithm 1: `ConstructContributionGraph`

Input: Query q, chunks C, and generated response $\mathcal{R}_q(C)$
Output: Contribution graph $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_q, c_q)$

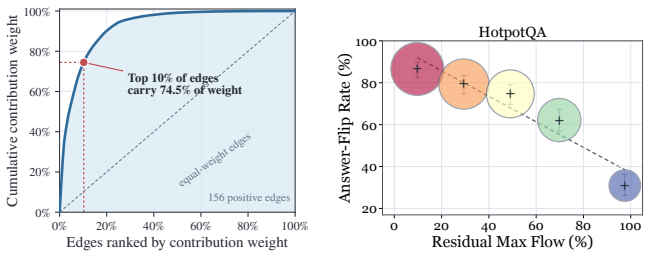
1 $\mathcal{V}_q \leftarrow \texttt{Token}(q) \cup \texttt{Token}(C) \cup \texttt{Token}(\mathcal{R}_q(C))$; $\mathcal{E}_q \leftarrow \varnothing$;
`// trace the response in one inference pass`
2 **for** $\ell \leftarrow 1$ **to** L **do**
3 **foreach** *token pair* (j, i) *with* $j \le i$ **do**
4 Record the forward message $\mathbf{m}_{j \to i}^{(\ell)}$;
5 Compute S_q from the response-token logits using Equation (6);
`// normalize each receiver's positive contributions`
6 **for** $\ell \leftarrow 1$ **to** L **do**
7 **foreach** *receiver token* i **do**
8 Compute $r_q^{(\ell)}(j, i)$ for every source token $j \le i$ using Equation (8);
9 Set $\omega_q^{(\ell)}(j, i) \leftarrow 0$ for every source token $j \le i$;
10 $z_i^{(\ell)} \leftarrow \sum_{j \le i} [r_q^{(\ell)}(j, i)]_+$;
11 **if** $z_i^{(\ell)} > 0$ **then**
12 **foreach** *source token* $j \le i$ *with* $r_q^{(\ell)}(j, i) > 0$ **do**
13 $\omega_q^{(\ell)}(j, i) \leftarrow [r_q^{(\ell)}(j, i)]_+ / z_i^{(\ell)}$;
`// aggregate the same token pair across layers`
14 **foreach** *token pair* (j, i) **do**
15 $c_q(j, i) \leftarrow \sum_{\ell=1}^{L} \omega_q^{(\ell)}(j, i)$;
16 **if** $c_q(j, i) > 0$ **then**
17 $\mathcal{E}_q \leftarrow \mathcal{E}_q \cup \{(j, i)\}$;
18 **return** $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_q, c_q)$;

We construct the contribution graph in three steps, as shown in Algorithm 1. First, we run the LLM on the original RAG input and record the information passed between token positions at each Transformer layer (lines 2-4). Second, we perform one backward pass to measure how much each message contributes to the generated response (lines 6-13). Finally, we normalize the positive contributions and aggregate them across layers to obtain the graph edges and their weights (lines 14-18).

Complexity analysis. Constructing the contribution graph requires one forward pass on the original RAG input and one backward pass for contribution tracing. Since contribution tracing has the same asymptotic complexity as the attention computation in the LLM, the overall construction cost is of the same asymptotic order as a single LLM inference.

5 CONTRACT-PRICED RESIDUAL-FLOW SEARCH

In this section, we present the second stage of `ReFlow`, which uses the contribution graph to evaluate the joint effect of a token set and efficiently search for a solution to the TCE problem.



(a) Contribution-weight concentration. **(b) Residual flow and answer changes.**

Figure 4: Motivation for residual-flow search.

5.1 Maximum Flow and Search Objective

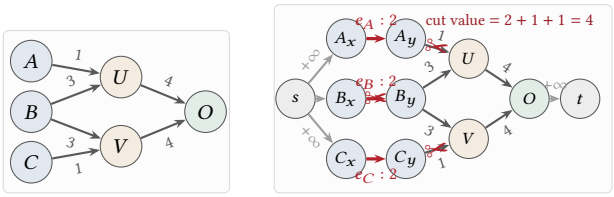
▶ **What does maximum contribution flow represent?** In the contribution graph, each directed edge represents positive contribution transmitted between two token positions, and its capacity records the amount of that contribution. A directed path from a retrieved token to a generated-answer token therefore represents a route through which retrieved information can contribute to the response. We treat the retrieved tokens as the source set and the generated-answer tokens as the target set. Within this graph abstraction, a feasible flow distributes contribution across multiple routes without exceeding their edge capacities. We denote the resulting maximum flow by $\Phi_q(\mathcal{G}_q)$, which measures the largest total contribution that can be carried from the retrieved context to the generated response.

The edge capacities are important because different contribution routes can have substantially different strengths. As shown in Figure 4a, the positive edge weights in a HotpotQA contribution graph span several orders of magnitude, with the largest 10% accounting for 74.5% of the total contribution. Thus, treating all edges or paths equally would lose important information about how strongly

▶ **Residual flow after a token intervention.** For a selected token set $T \subseteq \texttt{Token}(C)$, we need to estimate how strongly replacing these tokens disrupts the support for the original response. To this end, for a selected token set T, we represent its intervention on the contribution graph by removing the corresponding nodes and incident edges, thereby eliminating the contribution routed through these tokens. We write $\Phi_q(\mathcal{G}_q \setminus T)$ for the *residual maximum flow* that remains after the tokens in T are removed from $\mathcal{G}_q$; it quantifies how much contribution can still reach the original response along paths that avoid T. A smaller residual flow means that T intercepts more response-supporting contribution and is therefore more likely to change the response after counterfactual intervention. Unlike independent token scores, this measure evaluates T jointly: removing one token may leave alternative contribution paths intact, whereas removing several tokens together may block these paths simultaneously.

We note that residual maximum flow does not predict the new response produced by a counterfactual intervention. Instead, it estimates how much contribution to the original response remains after excluding the selected tokens from the contribution graph. This graph operation serves as a surrogate for replacing the selected tokens with their counterfactual tokens, allowing us to evaluate a candidate token set without rerunning the LLM.

We empirically examine the relationship between this surrogate and actual response changes on HotpotQA. For each query, we sample multiple sets of 20 tokens from the retrieved chunks, replace the selected tokens with their counterfactual tokens, and rerun the LLM. As shown in Figure 4b, token sets with lower residual flow are more likely to change the response. The bubble area represents the magnitude of the response change, with larger bubbles indicating greater differences from the original response. This observation motivates using residual maximum flow to guide the search for effective token sets.



(a) Original $\mathcal{G}_q$. **(b) $\mathcal{F}_q^2$ with $\lambda = 2$.**

Figure 5: A contribution graph and its priced flow network.

▶ **Search objective.** We seek a small token set whose removal substantially reduces the residual flow. To balance these two goals, we assign a price $\lambda \geq 0$ to each selected token and define

$$T_\lambda \in \arg\min_{T \subseteq \texttt{Token}(C)} \left\{ \lambda |T| + \Phi_q(\mathcal{G}_q \setminus T) \right\}. \tag{13}$$

The first term penalizes the number of selected tokens, while the second measures the contribution flow that remains after their removal. A larger λ places more emphasis on selecting fewer tokens, whereas a smaller λ places more emphasis on reducing the residual flow. This pricing formulation is inspired by linear contract theory [12, 60], where an action is chosen by balancing its benefit against its cost. Here, selecting more tokens can remove more contribution flow but also incurs a higher token cost.

5.2 Priced Flow-Network Transformation

Although Equation (13) captures the desired tradeoff, directly optimizing it appears to require searching all token sets. For a fixed price λ, we instead obtain its exact optimum through one minimum-cut computation. The key is to represent selecting a retrieved token by an edge of capacity λ, while preserving the contribution-edge capacities that measure residual flow.

Definition 5.1 (Priced contribution flow network)**.** For a token price $\lambda \geq 0$, the priced contribution flow network $\mathcal{F}_q^\lambda$ is constructed from $\mathcal{G}_q$ as follows.

(1) **Split selectable tokens.** For each $i \in \texttt{Token}(C)$, split i into i_x and i_y, and add a selection edge $e_i = (i_x, i_y)$ with capacity λ.
(2) **Preserve contribution edges.** Each original edge keeps its capacity; edges enter i_x and leave i_y for split tokens.
(3) **Add terminals.** Connect a super-source s to all i_x and all tokens in the generated answer to a sink t with capacity $+\infty$.

Example 5.1. In Figure 5(a), the two branches carry four units each, giving a maximum flow of eight. For $\lambda = 2$, the minimum cut in Figure 5(b) crosses e_B and two unit-capacity edges. It selects $T_2 = \{B\}$ with value $2 + 1 + 1 = 4$, where two is the selection cost and $1 + 1$ is the residual flow. Selecting no token has value eight, while selecting all three has value six.

▶ **Solving one token price.** For a given λ, one minimum s–t cut in $\mathcal{F}_q^\lambda$ simultaneously decides which selection edges to cut and how much contribution flow remains after those selections.

Lemma 5.2. *The minimum cut value in $\mathcal{F}_q^\lambda$ is exactly:*

$$\min_{T \subseteq \texttt{Token}(C)} \left\{ \lambda |T| + \Phi_q(\mathcal{G}_q \setminus T) \right\}.$$

The selection edges crossed by the cut identify a set T_λ satisfying Equation (13).

Proof Sketch. Consider a cut that selects a token set T. For every token in T, the cut crosses its edge (i_x, i_y) of capacity λ, giving a total token cost of $\lambda|T|$. The cut must further disconnect every s-t path that does not pass through a token in T. By the max-flow/min-cut theorem, the minimum cost of disconnecting these paths is $\Phi_q(\mathcal{G}_q \setminus T)$. Hence, the minimum cut associated with T has capacity: $\lambda|T| + \Phi_q(\mathcal{G}_q \setminus T)$. Minimizing this value over all cuts is exactly Equation (13); therefore, the token edges crossed by the minimum cut identify T_λ. □

5.3 Residual Flow-based Token Identification

Section 5.2 shows that, for any fixed price λ, one minimum-cut computation returns a token set $T_\lambda \subseteq \texttt{Token}(C)$. Since the appropriate price is unknown, we need to identify all distinct token sets that can be returned as λ varies.

For any token set $X \subseteq \texttt{Token}(C)$, its objective value at price λ is: $f_X(\lambda) = \lambda|X| + \Phi_q(\mathcal{G}_q \setminus X)$. For a fixed λ, the fixed-price problem returns the token set with the smallest objective value:

$$f_{T_\lambda}(\lambda) = \min_{X \subseteq \texttt{Token}(C)} f_X(\lambda). \tag{14}$$

Price range. The maximum relevant price is $\lambda_q^{\max} = \Phi_q(\mathcal{G}_q)$, because every $\lambda \geq \lambda_q^{\max}$ returns $T_\lambda = \varnothing$. It is therefore sufficient to consider $\mathcal{T}_q = \left\{ T_\lambda : 0 \leq \lambda \leq \lambda_q^{\max} \right\}$.

Monotonicity and the $n+1$ bound. Let $n = |\texttt{Token}(C)|$. We first establish how the number of returned tokens changes with the price. The following lemma shows that it cannot increase with λ and decreases whenever the returned set changes.

Lemma 5.3. *For any $0 \leq \lambda_a < \lambda_b \leq \lambda_q^{\max}$, $|T_{\lambda_a}| \geq |T_{\lambda_b}|$. If $T_{\lambda_a} \neq T_{\lambda_b}$, then, $|T_{\lambda_a}| > |T_{\lambda_b}|$. Moreover, if $T_{\lambda_a} = T_{\lambda_b}$, then the same set is returned for every price between them: $T_\lambda = T_{\lambda_a} = T_{\lambda_b}, \forall \lambda \in [\lambda_a, \lambda_b]$.*

The complete proof is provided in Appendix B. By Lemma 5.3, different returned sets have different cardinalities. Since the cardinality can take only the $n+1$ values $\{0, \ldots, n\}$ (i.e, $|\mathcal{T}_q| \leq n+1$.)

Equivalently, $[0, \lambda_q^{\max}]$ can be partitioned into at most $n+1$ consecutive price intervals, with all prices in each interval returning the same token set.

▶**Divide-and-conquer enumeration.** Lemma 5.3 bounds the number of candidates but does not locate them. Consider two prices $\lambda_a < \lambda_b$. If $T_{\lambda_a} = T_{\lambda_b}$, the same set is returned throughout $[\lambda_a, \lambda_b]$ by Lemma 5.3. Otherwise, $|T_{\lambda_a}| > |T_{\lambda_b}|$, and their objective values are equal at

$$\tau = \frac{\Phi_q(\mathcal{G}_q \setminus T_{\lambda_b}) - \Phi_q(\mathcal{G}_q \setminus T_{\lambda_a})}{|T_{\lambda_a}| - |T_{\lambda_b}|}. \tag{15}$$

Lemma 5.4. *Let $0 \leq \lambda_a < \lambda_b$, $T_{\lambda_a} \neq T_{\lambda_b}$, and let τ be given by Equation* (15). *Then $\lambda_a < \tau \leq \lambda_b$. Moreover, $T_\tau = T_{\lambda_b}$ if and only if $[\lambda_a, \lambda_b]$ contains no other returned token set; that is, $T_\lambda \in \{T_{\lambda_a}, T_{\lambda_b}\}$ for every $\lambda \in [\lambda_a, \lambda_b]$. Otherwise, $\lambda_a < \tau < \lambda_b$, T_τ is a new candidate, and*

$$|T_{\lambda_b}| < |T_\tau| < |T_{\lambda_a}|.$$

The complete proof is provided in Appendix B.

Starting from $[0, \lambda_q^{\max}]$, we recursively refine price ranges whose endpoint sets are different. By Lemma 5.4, each range either contains only its two endpoint sets or yields a new candidate T_τ. In the latter case, we split the range as

$$[\lambda_a, \lambda_b] : (T_{\lambda_a}, T_{\lambda_b}) \quad \longrightarrow \quad [\lambda_a, \tau] : (T_{\lambda_a}, T_\tau), \qquad [\tau, \lambda_b] : (T_\tau, T_{\lambda_b}).$$

Both new ranges have smaller cardinality gaps, so repeated refinement terminates without missing any candidate.

Example 5.2. Let $\lambda_q^{\max} = \Phi_q(\mathcal{G}_q) = 10$, so the initial price range is $[0, 10]$. Its endpoint solutions satisfy $\left(|T_0|, \Phi_q(\mathcal{G}_q \setminus T_0)\right) = (4, 0)$, and $\left(|T_{10}|, \Phi_q(\mathcal{G}_q \setminus T_{10})\right) = (0, 10)$. Equation (15) gives $\tau = 5/2$, where the minimum cut returns a new candidate with size two and residual flow three. The two resulting ranges yield $\tau = 3/2$ and $\tau = 7/2$, returning three- and one-token candidates, respectively. The remaining dividing prices $1, 2, 3, 4$ add no candidate. Thus, Figure 6 finds all five candidates without scanning the price range.

Algorithm 2 enumerates the candidates from the price range (lines 1–2) and verifies them by increasing cardinality (lines 3–5). For each range, `Divide` computes T_τ (lines 6–8). It returns if $T_\tau = T_{\lambda_b}$

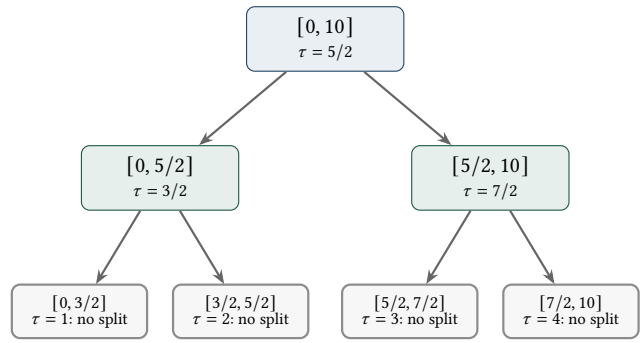


Figure 6: Example of recursive interval refinement.

Algorithm 2: `PricedFlowSearch`

Input: Contribution graph $\mathcal{G}_q$, and retriced chunks C
Output: Selected token set T^*

```
// initialize the endpoint candidates
1  T_q ← {};
2  Divide(0, Φ_q(G_q));
// verify candidates with the LLM
3  Order the nonempty sets in T_q by increasing cardinality;
4  foreach T ∈ T_q \ {∅} in this order do
5      if R_q(C[T ← Ψ(T)]) ≠ R_q(C) then return T;

// recursively refine one price interval
6  Function Divide(λ_a, λ_b):
7      τ ← compute τ via Equation (15);
8      Compute a minimum cut of F_q^τ, obtaining T_τ;
9      if T_τ = T_λb then return; ;
10     T_q ← T_q ∪ {T_τ};
11     Divide(λ_a, τ);
12     Divide(τ, λ_b);
```

Table 1: Dataset statistics.

Dataset	Abbr.	Domain	Corpus tokens	Queries
HotpotQA	HQA	Wikipedia	1,296,845	1,000
TimeQA	TQA	Temporal	1,711,829	1,000
FinQA	FQA	Finance	1,122,262	1,000
MuSiQue	MSQ	Multi-hop	1,349,003	1,000
QuaRTz	QTZ	Science	2,386	1,000
TriviaQA	TrQA	Wikipedia	2,603,344	1,000
2WikiMultiHopQA	2Wiki	Wikipedia	679,759	1,000
PopQA	PopQA	Long-tail facts	2,672,041	1,000

(line 9); otherwise, it records T_τ and recurses on the two subranges (lines 10–12).

Theorem 5.5. *Algorithm 2 can correctly enumerate every token set in $\mathcal{T}_q$.*

▶**Contract-theoretic interpretation.** Here, we interpret our search algorithm through the lens of contract theory. This theory studies how a principal induces an agent's action when that action cannot be prescribed directly. Its foundations were recognized by the 2016 Nobel Prize in Economic Sciences awarded to Hart and Holmström [60]. In a linear contract, the principal specifies a payment based on the action's output, and the agent chooses an action that maximizes the payment minus its cost [12].

Our priced search follows this decision structure. `ReFlow` sets the token price λ, and the fixed-price problem returns an intervention target set T, viewed as the induced action. Its output is the removed contribution flow $\Phi_q(\mathcal{G}_q) - \Phi_q(\mathcal{G}_q \setminus T)$, while its cost is $\lambda|T|$. The resulting utility is

$$u_\lambda(T) = \Phi_q(\mathcal{G}_q) - \Phi_q(\mathcal{G}_q \setminus T) - \lambda|T| = \Phi_q(\mathcal{G}_q) - f_T(\lambda). \quad (16)$$

Since $\Phi_q(\mathcal{G}_q)$ is independent of T, maximizing $u_\lambda(T)$ returns the same T_λ as minimizing $f_T(\lambda)$. Thus, the fixed-price problem can be viewed as a linear contract in which λ is the token price and T_λ is the induced action.

6 EXPERIMENTAL EVALUATION

We now present the experimental results. Section 6.1 discusses the setup. We discuss the results in Sections 6.2–6.4.

6.1 Setup

▶ **Benchmark datasets.** We employ eight real-world datasets to evaluate the performance of different RAG explanation methods. These datasets cover diverse domains, including multi-hop reasoning, scientific inquiry, and finance question answering, ensuring both breadth and challenge in evaluation. Their statistics, including the number of tokens and questions as well as their corresponding domains, are summarized in Table 1. For each dataset, we randomly sample 1,000 queries [78]. Detailed descriptions are provided in Appendix C.1.

▶ **Compared algorithms.** We compare `ReFlow` with the following six baselines:

- `Random` uniformly samples tokens from the retrieved chunks. We use five random seeds and report their mean.
- `Attention` ranks retrieved tokens by the attention they receive from generated-response tokens.
- `Gradient (G×I)` uses gradient times input to measure how strongly each retrieved token affects the original-response score and ranks tokens by the resulting attribution.
- `Token Shapley (TS)` applies shapley-value analysis to measure the contributions of retrieved tokens [54]. To reduce computation, we retain the 128 tokens with the highest attention scores and estimate their shapley values using paired leverage-score coalition sampling [43].
- `MIRAGE`, a state-of-the-art token-level RAG explanation method, ranks retrieved tokens by its attribution scores [50].
- `ARC-JSD`, a sentence-level RAG explanation method, assigns its sentence attribution to the token occurrences in that sentence and ranks the resulting token scores [33].

Each baseline selects its highest-ranked tokens using a budget equal to the number of intervention targets attempted by `ReFlow`.

▶ **Evaluation metrics.** We report two groups of flip-rate metrics. First, the *Answer Flip Rate* (Ans-FR) is the percentage of queries whose response changes after the counterfactual intervention, regardless of whether the new response is correct. To measure correctness loss, we further use exact match (EM) and accuracy (Acc) [21]. For each criterion, we retain the queries answered correctly before intervention and report the percentage that becomes incorrect afterward, yielding EM-FR and Acc-FR. Token-level F1 and normalized EM produce identical flip decisions for our short-answer outputs; we therefore report only EM in the remainder of the paper.

Second, a meaning-preserving intervention may also change the response because the LLM is sensitive to surface wording. Following recent counterfactual prompting work [71], we pair each counterfactual intervention with a control intervention on the same token set: lexical tokens are replaced by meaning-preserving synonyms or aliases, while dates and numbers remain unchanged. We subtract the control flip rate from the corresponding flip rate under counterfactual intervention and call the difference the *Counterfactual Flip Rate* (CFR). This gives Ans-CFR, EM-CFR, and Acc-CFR; a larger value indicates that the observed flips arise primarily from changed factual information rather than surface-form sensitivity.

▶ **Model settings.** In our experiments, we use Qwen2.5-7B-Instruct [70] as the default target LLM because of its strong performance among open-weight models of similar scale. In addition, we test Llama-3.1-8B-Instruct [11], Mistral-Nemo-Instruct-2407 [42], and Qwen3-4B-Instruct-2507 [69], covering three model families, and parameter scales ranging from 4B to 12B.

▶ **Counterfactual intervention protocol.** We instantiate the counterfactual intervention defined in Section 2.3 by constructing a shared counterfactual token pool before running any method. For each candidate token, an LLM editor generates one-token alternatives that preserve its semantic type, syntactic role, and morphology while making the sentence express incompatible factual information. Each editable token is assigned at least one validated counterfactual token $\Psi(t_i)$, and the pool is constructed once and shared across methods. Given a selected set T, we can construct $C[T \leftarrow \Psi(T)]$ based on this pool. For the meaning-preserving control, we combine LLM-generated candidates with WordNet [41] and retain only synonyms or aliases that preserve the original referent, fact, and grammatical role; dates and numbers remain unchanged. The control intervention is applied to the same eligible positions in T. Appendix A.1 provides the complete prompts.

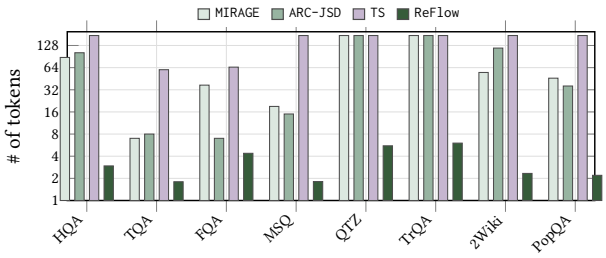


Figure 7: Intervention sizes needed to match ReFlow's Acc-FR.

▶ **Running details.** For each query, all methods use the same top-5 retrieved chunks. We use Qwen3-Embedding-0.6B [74] to encode queries and corpus chunks for vector search. When a dataset provides expert-annotated chunk boundaries, we retain them to preserve its human-curated semantic structure; otherwise, we segment the corpus into chunks of 384 tokens. All experiments are conducted on a server equipped with two NVIDIA A100 40GB GPUs.

6.2 Comparison with Existing Methods

We compare existing TCE methods in terms of both explanation effectiveness and token efficiency.

▶ **Exp. 1. Overall effectiveness and token efficiency.** We evaluate the methods from two complementary perspectives.

- First, each method uses the same number of intervention targets attempted by ReFlow. This setting compares how effectively different methods select tokens. Table 2 reports the resulting flip-rate metrics.
- Second, we gradually increase the number of top-ranked tokens replaced by TS, MIRAGE, and ARC-JSD, and record the minimum number required to match the Acc-FR achieved by ReFlow. This setting compares the number of token replacements required to achieve the same effectiveness. Figure 7 reports these results.

We make three observations. *1) Under the same token budget,* ReFlow *consistently achieves the best performance across all reported metrics on all eight datasets.* Compared with the strongest baseline, ReFlow improves Ans-FR by up to 45.8 percentage points on TrQA and EM-FR by up to 47.3 points on PopQA. Moreover, this performance is achieved using only 3.17 token replacements per query on average, with dataset-level averages ranging from 1.57 to 5.70 and a median of at most two tokens on seven datasets. Since all baselines use the same per-query budget in this comparison, the gains come from identifying more effective token sets rather than replacing more tokens. The absolute flip rates are lower for all methods on QTZ, likely because its binary answer space makes the same answer more likely to remain unchanged by chance after an intervention.

2) Matching the effectiveness of ReFlow *requires substantially more baseline tokens.* On TQA and FQA, TS first matches the Acc-FR of ReFlow using 60 and 65 tokens, respectively, whereas ReFlow uses only 1.80 and 4.38 tokens on average. On HQA, MIRAGE and ARC-JSD require 89 and 102 tokens, respectively, compared with only 2.95 for ReFlow. Increasing the replacement budget does not always close the gap: even with up to 128 tokens, TS fails to match ReFlow on six of the eight datasets, while MIRAGE and ARC-JSD also fail on QTZ and TrQA. These results show that ReFlow identifies compact token sets whose joint replacement has a much stronger effect on the generated response.

3) No baseline performs consistently best across datasets. TS is the strongest baseline on HQA and TQA, ARC-JSD on FQA, MSQ, and QTZ, and MIRAGE on TrQA, 2Wiki, and PopQA, while Random and Attention generally perform worst. This variation indicates that no single attribution score consistently captures the joint token dependencies required by TCE.

6.3 Detailed Analysis of ReFlow

In this subsection, we provide an in-depth analysis of why and how ReFlow achieves superior performance, from four perspectives.

▶ **Exp.1. Residual flow and answer changes.** We first test whether a lower residual maximum flow is associated with a higher probability of changing the response. Specifically, for each dataset, we sample 100 queries and, for each query and replacement budget $\{1, 3, 5\}$, uniformly sample 10 token sets. For each set, we compute its residual maximum flow, apply the stored counterfactual replacements, and rerun the LLM, yielding 3,000 evaluations per dataset and 24,000 in total. Figure **??** ranks the 10 sets within each query-budget group from the lowest to the highest residual flow. Across all datasets, the lowest-residual-flow group achieves a higher Ans-FR than the highest-residual-flow group. These results show that a larger reduction in contribution flow is consistently associated with a higher probability of changing the response.

▶ **Exp.2. Effectiveness of contract-priced search.** We evaluate the contract-priced search using two variants: One-shot, which uses a single price λ=1, and Grid, which searches 33 logarithmically spaced prices from 2^{-8} to 2^{8}. Figure **??** shows that ReFlow consistently achieves higher Ans-FR with fewer or comparable token replacements. For example, on HQA, ReFlow reaches 94.0% Ans-FR with 2.66 tokens, compared with 90.2%/24.05 for One-shot and 93.9%/3.10 for Grid. Overall, exact enumeration improves Ans-FR over Grid by 0.1–16.6 points, showing that a single price may over-select while a fixed grid may miss effective candidates.

▶ **Exp.3. Efficiency of candidate enumeration.** We compare the number of candidate sets actually needed in practice with the theoretical $n + 1$ bound, which ranges from 36.0 to 577.0 across datasets. In practice, only 8.2–27.3 candidate sets are needed, yielding a 4.5–48.5× reduction.

▶ **Exp.4. Running-time analysis.** ReFlow consists of two main computational stages: contribution-graph construction and exact candidate enumeration. The first stage performs forward and backward computation through the Transformer, whose cost grows quadratically with the number of prompt tokens and also depends on the model dimension and number of layers. The second stage performs maximum-flow computations on the constructed contribution graph, whose practical cost mainly depends on the number of graph nodes and edges. Since the contribution graph typically remains very sparse, candidate enumeration is much cheaper. As shown in Figure **??**, graph construction takes 2.07 to 11.60 seconds per query, whereas exact enumeration requires only 0.008 to 0.201 seconds. On HQA, for example, graph construction takes 4.40 seconds, while all 46.4 maximum-flow computations together take only 0.137 seconds. Thus, graph construction dominates the running time, while exact candidate enumeration adds little overhead.

6.4 Application Studies

Finally, we demonstrate how ReFlow can support three downstream RAG applications: RAG attack, RAG defense, and RAG provenance.

Table 2: Detailed performance comparison of different token-selection methods across eight datasets, where darker shading indicates higher flip rates and bold marks the best results..

Dataset	Metric	Random	Attention	G×I	TS	MIRAGE	ARC-JSD	ReFlow
		Baselines						*Ours*
HQA	Ans-FR	25.9	35.8	52.1	76.3	46.1	48.2	**95.1**
	EM-FR	10.4	23.6	37.2	70.9	33.2	36.0	**94.8**
	Acc-FR	8.3	18.9	31.8	63.3	30.3	31.4	**85.2**
	Ans-CFR	3.6	17.9	18.6	48.6	21.6	24.2	**56.5**
	EM-CFR	2.4	15.3	21.2	55.5	20.8	24.9	**69.7**
	Acc-CFR	1.9	11.3	17.9	51.9	20.1	21.5	**62.4**
TQA	Ans-FR	7.8	12.7	45.3	75.3	70.6	76.2	**93.4**
	EM-FR	5.3	7.5	37.5	79.9	73.0	75.1	**99.0**
	Acc-FR	4.7	6.8	35.9	68.8	67.3	68.8	**86.4**
	Ans-CFR	3.3	8.2	23.1	52.5	45.4	49.5	**62.1**
	EM-CFR	3.6	4.8	23.9	63.1	54.9	56.0	**77.1**
	Acc-CFR	3.1	3.9	24.3	55.2	53.1	53.7	**70.4**
FQA	Ans-FR	20.8	28.0	34.6	60.7	43.2	60.2	**84.9**
	EM-FR	11.1	23.5	18.5	60.5	33.3	63.0	**82.7**
	Acc-FR	10.6	18.3	15.5	45.1	28.2	57.0	**71.1**
	Ans-CFR	7.6	17.8	14.7	54.8	34.5	58.4	**79.2**
	EM-CFR	1.5	17.3	11.1	55.6	30.9	61.7	**76.5**
	Acc-CFR	1.8	12.0	6.3	42.3	25.4	56.3	**69.0**
MSQ	Ans-FR	9.4	17.0	42.5	61.2	58.8	71.0	**95.3**
	EM-FR	4.1	11.3	33.3	63.3	62.7	79.3	**99.3**
	Acc-FR	4.1	14.0	29.4	58.3	56.2	71.5	**89.4**
	Ans-CFR	2.3	8.1	17.0	42.1	34.3	46.8	**68.3**
	EM-CFR	1.3	8.0	14.7	51.3	48.0	67.3	**84.0**
	Acc-CFR	1.2	9.8	14.0	49.8	43.0	57.9	**74.5**
QTZ	Ans-FR	11.8	18.4	20.5	26.1	20.8	23.9	**40.2**
	EM-FR	8.7	15.4	16.9	19.8	17.2	19.8	**35.7**
	Ans-CFR	5.5	11.2	12.1	13.6	12.8	17.2	**32.3**
	EM-CFR	5.6	12.3	12.2	13.7	12.5	17.0	**31.4**
TrQA	Ans-FR	2.7	7.9	22.2	16.0	31.0	23.1	**76.8**
	EM-FR	0.8	5.6	14.1	7.8	23.0	15.7	**70.2**
	Acc-FR	0.4	4.4	11.4	6.2	19.4	13.3	**57.0**
	Ans-CFR	0.1	2.4	9.4	9.3	15.2	10.4	**50.6**
	EM-CFR	0.1	2.8	9.0	6.1	14.7	10.1	**55.9**
	Acc-CFR	0.0	2.8	8.4	5.1	14.6	9.8	**48.4**
2Wiki	Ans-FR	8.3	13.8	52.1	55.4	63.8	59.0	**90.1**
	EM-FR	3.6	10.0	58.2	52.1	67.1	54.3	**94.8**
	Acc-FR	3.1	8.9	53.1	47.2	63.7	49.7	**88.2**
	Ans-CFR	1.0	5.7	25.2	34.5	33.8	35.0	**59.6**
	EM-CFR	1.1	5.2	33.9	37.7	44.2	38.8	**71.8**
	Acc-CFR	1.1	5.6	33.0	34.8	44.0	37.9	**68.6**
PQA	Ans-FR	4.7	15.0	43.5	45.5	60.1	55.9	**98.1**
	EM-FR	1.9	15.1	31.7	30.9	49.9	40.0	**97.2**
	Acc-FR	1.5	11.2	29.8	26.4	45.3	36.2	**81.3**
	Ans-CFR	1.5	7.7	20.9	27.7	34.5	33.6	**65.2**
	EM-CFR	1.3	14.0	23.5	24.4	40.8	32.4	**84.7**
	Acc-CFR	0.9	10.0	23.4	21.2	37.2	30.7	**69.6**

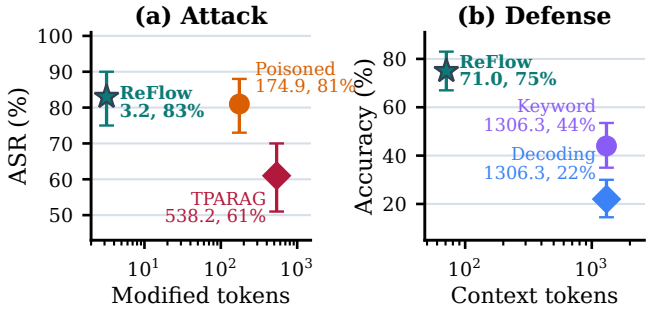


Figure 8: RAG attack and defense on the HotpotQA dataset.

▶ **Exp.1. Token-efficient RAG attack.** In this experiment, we show that the tokens identified by ReFlow can be used for RAG poisoning. Specifically, for each query, we use ReFlow to select tokens and replace them with their counterfactual alternatives. We consider an attack successful when this replacement changes the generated answer. We compare ReFlow with two SOTA RAG poisoning methods: PoisonedRAG [79], which injects five malicious passages, and TPARAG [36], which iteratively generates malicious passages at the token level. Figure 8(a) reports the attack success rate (ASR) and the number of modified tokens for the three methods on 100 HotpotQA queries. We observe that ReFlow achieves an attack success rate of 83.0% while modifying only 3.19 tokens on average, outperforming PoisonedRAG (81.0% with 174.9 injected tokens) and TPARAG (61.0% with 538.2 tokens). These results show that directly poisoning the tokens identified by ReFlow is both more effective and substantially more token-efficient than injecting complete malicious passages, further confirming that ReFlow identifies tokens critical to the RAG response.

▶ **Exp.2. Effective RAG defense.** In this experiment, we first use PoisonedRAG or TPARAG to inject one malicious chunk into the top-10 retrieved chunks for each query. We then compare three defense strategies. Keyword-Guard, from RobustRAG [68], lets the LLM answer the question using each chunk independently, extracts keywords from the ten answers, and generates the final answer

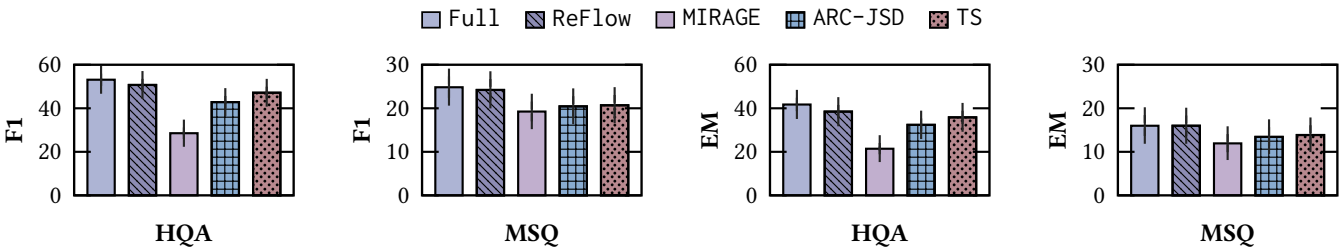


Figure 9: Illustration of RAG provenance.

from keywords supported by multiple chunks. `Decoding-Guard` also processes each chunk independently, but sums their next-token probabilities at every decoding step to generate the final answer. For our `ReFlow-Guard`, we use `ReFlow` to identify important tokens before poisoning and store the sentences containing them. After poisoning, the LLM answers the question using only these protected sentences. Figure 8(b) reports the accuracy under attack and the number of context tokens used by the three methods on 100 HotpotQA queries. We observe that `ReFlow-Guard` achieves 75.0% accuracy using only 71.0 context tokens on average, substantially outperforming `Keyword-Guard` and `Decoding-Guard`, which use 1,306.3 tokens but achieve only 44.0% and 22.0% accuracy, respectively. Thus, `ReFlow-Guard` uses 18.4× fewer tokens while achieving much higher accuracy, showing that `ReFlow` effectively identifies the sentences that should be protected.

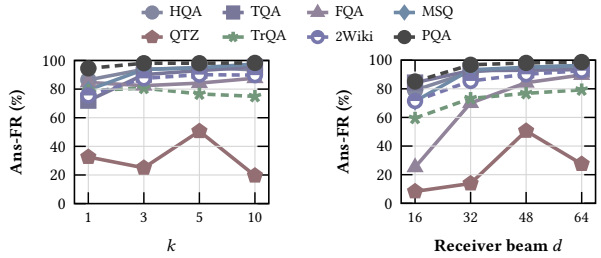
▶ **Exp.3. RAG provenance.** We evaluate whether the sentences containing the tokens identified by `ReFlow` can serve as provenance for the generated answer. Specifically, we prompt the LLM using only these sentences and compare the resulting answer quality with `Full`, which uses all top-5 retrieved chunks. We also compare with `TS`, `MIRAGE` and `ARC-JSD` by prompting the LLM with the sentences containing their selected tokens under the same sentence budget. Figure 9 reports the results on HQA and MSQ. On HQA, `ReFlow` achieves 38.48 EM and 50.72 F1 using only 51.7 tokens, compared with 41.73 EM and 53.13 F1 using 524.2 tokens for `Full`. On MSQ, `ReFlow` achieves 15.99 EM and 24.21 F1 using 48.8 tokens, nearly matching `Full`, which achieves 15.99 EM and 24.83 F1 using 550.0 tokens. Moreover, `ReFlow` achieves higher F1 than `MIRAGE` and `ARC-JSD`: 50.72 versus 28.55 and 42.84 on HQA, and 24.21 versus 19.25 and 20.44 on MSQ. These results show that the sentences identified by `ReFlow` retain the key retrieved information supporting the generated answer and can therefore serve as concise RAG provenance.

6.5 Sensitivity to Key System Parameters

We study the sensitivity of `ReFlow`.

▶ **Exp. 1. Effect of the number of retrieved chunks k.** We vary $k \in \{1, 3, 5, 10\}$ while keeping all other settings fixed. As shown in Figure 10a, we can see `ReFlow` consistently achieves strong performance across different values of k, with similar trends observed across all evaluation metrics. These results demonstrate that `ReFlow` is robust to the number of retrieved chunks.

▶ **Exp. 2. Effect of receiver beam d.** We vary $d \in \{16, 32, 48, 64\}$, where d denotes the maximum number of positive incoming contributions retained for each receiver token. As shown in Figure 10b, we can see that the performance of `ReFlow` generally improves



(a) Retrieval depth k. **(b) Receiver beam d.**

Figure 10: Sensitivity of `ReFlow` to key system parameters.

as d increases and reaches its best overall result at $d = 48$, with similar trends across the other evaluation metrics. Increasing d to 64 provides no consistent gain while producing a substantially larger contribution graph. We therefore use $d = 48$ as the default setting. Additional results, including contribution-graph sizes and performance under the remaining evaluation metrics, are reported in our technical report [].

7 RELATED WORK

In this section, we first review existing RAG methods and their applications. We then discuss methods for interpreting LLM-generated responses and RAG.

• **RAG methods and applications.** RAG improves LLM generation on knowledge-intensive tasks by retrieving relevant information from an external corpus and adding it to the model input [15, 18, 30]. Recent methods go beyond the standard retrieve-then-generate pipeline by organizing external knowledge into richer hierarchical or graph structures that capture relationships within and across documents [13, 20, 21, 53]. RAG has been applied to a wide range of knowledge-intensive tasks, from open-domain and multi-hop question answering [26, 30, 59] to document understanding, including multi-document question answering and reasoning over mixed textual and tabular content [65**?**]. It is also increasingly used in high-stakes domains such as medicine and law, where responses must be grounded in specialized evidence [66**?**]. In data management, RAG has been used for NL2SQL [14, 31], data cleaning [32, 44], database configuration [29], DBMS diagnosis [55, 76, 77], and SQL rewriting [35, 56]. Across these applications, prior work primarily improves how evidence is organized, retrieved, and used to produce accurate responses. In contrast, we hold the RAG pipeline fixed and identify which retrieved tokens jointly determine a particular generated response.

• **LLM and RAG explanations.** Existing interpretability methods explain model behavior from either the input or internal computation. Input-attribution methods assign importance scores to input features using attention [1], gradients or Integrated Gradients [**?**], perturbations [54], or Shapley values [43]. Mechanistic interpretability instead intervenes on internal activations to identify the components involved in processing factual information [**? ? ?**]. These methods mainly explain which input features or internal components are important to a given prediction, rather than construct an alternative input that changes the model output.

For RAG, explanation methods broadly follow two paradigms: citation generation and context attribution. The former links generated claims to passages presented as supporting evidence [17**? ?**]. The latter estimates the influence of retrieved content using either model-internal saliency signals or context perturbations [9, 33, 37, 50]. Regardless of their signals and granularities, these methods return citations or importance scores for individual tokens, sentences, or context segments. They do not explicitly search for a small set of retrieved tokens whose joint factual replacement changes the response.

Counterfactual explanations address the latter by constructing an input change that alters the output. `MiCE` and Shapley-guided latent search produce minimally edited counterfactuals for text classifiers [16, 52], where the target is a discrete label flip. Counterfactual prompting varies prompt factors under controlled baselines that separate their effect from general sensitivity to paraphrasing [71], and has been applied in RAG to assess prediction risk under varying retrieval quality and context utilization [6]. Closest to our setting, `RAGE` searches combinations and permutations of retrieved sources for minimal source-level perturbations that change the answer [51]. TCE differs in both intervention unit and objective: it replaces retrieved tokens with alternative facts rather than removing or reordering whole sources, and minimizes the joint cardinality of those replacements for an open-ended response. Replacement leaves the context well-formed and attributes the answer to what the evidence states rather than to its presence alone, while token-level granularity localizes that dependence to the specific facts the response rests on.

8 CONCLUSION

In this paper, we investigated the Token-Level Counterfactual Explanations for RAG (TCE) problem, aiming to identify the smallest set of retrieved tokens whose joint counterfactual intervention changes the generated response. To tackle this problem, we proposed `ReFlow`. It first constructs a contribution graph that captures how retrieved tokens jointly contribute to the response. Next, `ReFlow` uses a contract-priced search based on residual maximum flow to balance intervention size against remaining contribution. We prove that only a linear number of candidate sets require LLM verification. Extensive experiments on eight datasets validated the effectiveness and efficiency of `ReFlow`.

REFERENCES

[1] Samira Abnar and Willem Zuidema. 2020. Quantifying Attention Flow in Transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 4190–4197. https://doi.org/10.18653/v1/2020.acl-main.385

[2] Anthropic. 2026. Claude Fable 5 and Claude Mythos 5 System Card. https://www.anthropic.com/system-cards

[3] Ricardo Baptista and Matthias Poloczek. 2018. Bayesian Optimization of Combinatorial Structures. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. 462–471. https://proceedings.mlr.press/v80/baptista18a.html

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

[5] Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 2318–2335. https://doi.org/10.18653/v1/2024.findings-acl.137

[6] Lu Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2024. Controlling Risk of Retrieval-Augmented Generation: A Counterfactual Prompting Framework. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, 2380–2393. https://doi.org/10.18653/v1/2024.findings-emnlp.133

[7] Wenhu Chen, Xinyi Wang, and William Yang Wang. 2021. A Dataset for Answering Time-Sensitive Questions. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*. https://arxiv.org/abs/2108.06314

[8] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 3697–3711. https://doi.org/10.18653/v1/2021.emnlp-main.300

[9] Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. 2024. ContextCite: Attributing Model Generation to Context. *arXiv preprint arXiv:2409.00729* (2024). https://arxiv.org/abs/2409.00729

[10] Carola Doerr. 2020. Complexity Theory for Discrete Black-Box Optimization Heuristics. In *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, Benjamin Doerr and Frank Neumann (Eds.). Springer, 133–212. https://doi.org/10.1007/978-3-030-29414-4_3

[11] 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* (2024).

[12] Paul Dütting, Michal Feldman, and Yoav Gal Tzur. 2024. Combinatorial Contracts Beyond Gross Substitutes. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 92–108. https://doi.org/10.1137/1.9781611977912.5

[13] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *arXiv preprint arXiv:2404.16130* (2024).

[14] Ju Fan, Zihui Gu, Songyue Zhang, Yuxin Zhang, Zui Chen, Lei Cao, Guoliang Li, Samuel Madden, Xiaoyong Du, and Nan Tang. 2024. Combining small language models and large language models for zero-shot nl2sql. *Proceedings of the VLDB Endowment* 17, 11 (2024), 2750–2763.

[15] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. 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*. 6491–6501.

[16] Xiaoli Fern and Quintin Pope. 2021. Text Counterfactuals via Latent Optimization and Shapley-Guided Search. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 5578–5593. https://doi.org/10.18653/v1/2021.emnlp-main.452

[17] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. *arXiv preprint arXiv:2305.14627* (2023).

[18] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997* (2023).

[19] Aashish Ghimire, James Prather, and John Edwards. 2024. Generative AI in Education: A Study of Educators' Awareness, Sentiments, and Influencing Factors. *arXiv preprint arXiv:2403.15586* (2024).

[20] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. *arXiv e-prints* (2024), arXiv–2410.

[21] Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=hkujvAPVsg

[22] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics*. 6609–6625. https://doi.org/10.18653/v1/2020.coling-main.580

[23] Yucheng Hu and Yuxing Lu. 2024. Rag and rau: A survey on retrieval-augmented language model in natural language processing. *arXiv preprint arXiv:2404.19543* (2024).

[24] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *arXiv preprint arXiv:2311.05232* (2023).

[25] Yizheng Huang and Jimmy Huang. 2024. A Survey on Retrieval-Augmented Text Generation for Large Language Models. *arXiv preprint arXiv:2404.10981* (2024).

[26] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403* (2024).

[27] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1601–1611. https://doi.org/10.18653/v1/P17-1147

[28] Langchain. 2023. Langchain. https://python.langchain.com/docs/additional_resources/arxiv_references/.

[29] Jiale Lao, Yibo Wang, Yufei Li, Jianping Wang, Yunjia Zhang, Zhiyuan Cheng, Wanghu Chen, Mingjie Tang, and Jianguo Wang. 2024. Gptuner: A manual-reading database tuning system via gpt-guided bayesian optimization. *Proceedings of the VLDB Endowment* 17, 8 (2024), 1939–1952.

[30] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 9459–9474.

[31] Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. 2024. The Dawn of Natural Language to SQL: Are We Fully Ready? *arXiv preprint arXiv:2406.01265* (2024).

[32] Lan Li, Liri Fang, Bertram Ludäscher, and Vetle I Torvik. 2025. AutoDCWorkflow: LLM-based Data Cleaning Workflow Auto-Generation and Benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (Eds.). Association for Computational Linguistics, Suzhou, China, 7766–7780. https://doi.org/10.18653/v1/2025.findings-emnlp.410

[33] Ruizhe Li, Chen Chen, Yuchen Hu, Yanjun Gao, Xi Wang, and Emine Yilmaz. 2025. Attributing Response to Context: A Jensen–Shannon Divergence Driven Mechanistic Study of Context Attribution in Retrieval-Augmented Generation. *arXiv preprint arXiv:2505.16415* (2025). https://arxiv.org/abs/2505.16415

[34] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023. Large Language Models in Finance: A Survey. In *Proceedings of the fourth ACM international conference on AI in finance*. 374–382.

[35] Zhaodonghui Li, Haitao Yuan, Huiming Wang, Gao Cong, and Lidong Bing. 2024. LLM-R2: A Large Language Model Enhanced Rule-Based Rewrite System for Boosting Query Efficiency. *Proceedings of the VLDB Endowment* 18, 1 (2024), 53–65. https://doi.org/10.14778/3696435.3696440

[36] Zizhong Li, Haopeng Zhang, and Jiawei Zhang. 2026. Token-Level Precise Attack on RAG: Searching for the Best Alternatives to Mislead Generation. In *Findings of the Association for Computational Linguistics: EACL 2026*. Association for Computational Linguistics, 3193–3206. https://doi.org/10.18653/v1/2026.findings-eacl.167

[37] Fengyuan Liu, Nikhil Kandpal, and Colin Raffel. 2025. AttriBoT: A Bag of Tricks for Efficiently Approximating Leave-One-Out Context Attribution. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=9kJperA2a4

[38] Lei Liu, Xiaoyan Yang, Junchi Lei, Xiaoyang Liu, Yue Shen, Zhiqiang Zhang, Peng Wei, Jinjie Gu, Zhixuan Chu, Zhan Qin, et al. 2024. A Survey on Medical Large Language Models: Technology, Application, Trustworthiness, and Future Directions. *arXiv preprint arXiv:2406.03712* (2024).

[39] llamaindex. 2023. llamaindex. https://www.llamaindex.ai/.

[40] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 9802–9822. https://doi.org/10.18653/v1/2023.acl-long.546

[41] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41. https://doi.org/10.1145/219717.219748

[42] Mistral AI. 2024. Mistral NeMo. https://mistral.ai/news/mistral-nemo/.

[43] Christopher Musco and R. Teal Witter. 2025. Provably Accurate Shapley Value Estimation via Leverage Score Sampling. In *Proceedings of the Thirteenth International Conference on Learning Representations*.

[44] Zan Ahmad Naeem, Mohammad Shahmeer Ahmad, Mohamed Eltabakh, Mourad Ouzzani, and Nan Tang. 2024. RetClean: Retrieval-Based Data Cleaning Using LLMs and Data Lakes. *Proceedings of the VLDB Endowment* 17, 12 (2024), 4421–4424.

[45] Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M Mulvey, H Vincent Poor, Qingsong Wen, and Stefan Zohren. 2024. A Survey of Large Language Models for Financial Applications: Progress, Prospects and Challenges. *arXiv preprint arXiv:2406.11903* (2024).

[46] OpenAI. 2026. GPT-5.6 System Card. https://deploymentsafety.openai.com/gpt-5-6

[47] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training Language Models to Follow Instructions with Human Feedback. In *Advances in Neural Information Processing Systems*, Vol. 35. 27730–27744.

[48] Judea Pearl. 2009. *Causality: Models, Reasoning, and Inference* (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511803161

[49] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph Retrieval-Augmented Generation: A Survey. *arXiv preprint arXiv:2408.08921* (2024).

[50] Jirui Qi, Gabriele Sarti, Raquel Fernández, and Arianna Bisazza. 2024. Model Internals-based Answer Attribution for Trustworthy Retrieval-Augmented Generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 6037–6053. https://doi.org/10.18653/v1/2024.emnlp-main.347

[51] Joel Rorseth, Parke Godfrey, Lukasz Golab, Divesh Srivastava, and Jaroslaw Szlichta. 2024. RAGE Against the Machine: Retrieval-Augmented LLM Explanations. *arXiv preprint arXiv:2405.13000* (2024). https://arxiv.org/abs/2405.13000

[52] Alexis Ross, Ana Marasović, and Matthew E. Peters. 2021. Explaining NLP Models via Minimal Contrastive Editing (MiCE). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, 3840–3852. https://doi.org/10.18653/v1/2021.findings-acl.336

[53] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. *arXiv preprint arXiv:2401.18059* (2024).

[54] Michelle Si and Jian Pei. 2024. Counterfactual Explanation of Shapley Value in Data Coalitions. *Proceedings of the VLDB Endowment* 17, 11 (2024), 3332–3345. https://doi.org/10.14778/3681954.3682004

[55] Vikramank Singh, Kapil Eknath Vaidya, Vinayshekhar Bannihatti Kumar, Sopan Khosla, Murali Narayanaswamy, Rashmi Gangadharaiah, and Tim Kraska. 2024. Panda: Performance Debugging for Databases Using LLM Agents. (2024).

[56] Zhaoyan Sun, Xuanhe Zhou, and Guoliang Li. 2024. R-Bot: An LLM-based Query Rewrite System. *arXiv preprint arXiv:2412.01661* (2024).

[57] Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019. QuaRTz: An Open-Domain Dataset of Qualitative Relationship Questions. In *Proceedings of EMNLP-IJCNLP*. 5941–5946.

[58] Xue Tan, Yi Zheng, Chang Huo, Yunruo Zhang, Yu Liu, Hao Luan, Zhuyang Yu, Jun Dai, Xiaoyan Sun, and Ping Chen. 2026. PRA-RAG: Provably Robust Aggregation in Retrieval-Augmented Generation against Retrieval Corruption. In *Findings of the Association for Computational Linguistics: ACL 2026*. Association for Computational Linguistics, 36002–36017. https://doi.org/10.18653/v1/2026.findings-acl.1794

[59] Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391* (2024).

[60] The Royal Swedish Academy of Sciences. 2016. The Prize in Economic Sciences 2016: Contract Theory. https://www.nobelprize.org/prizes/economic-sciences/2016/popular-information/ Oliver Hart and Bengt Holmström were awarded the 2016 Prize in Economic Sciences for their contributions to contract theory.

[61] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30.

[63] Jinqiang Wang, Huansheng Ning, Yi Peng, Qikai Wei, Daniel Tesfai, Wenwei Mao, Tao Zhu, and Runhe Huang. 2024. A Survey on Large Language Models from General Purpose to Medical Applications: Datasets, Methodologies, and Evaluations. *arXiv preprint arXiv:2406.10303* (2024).

[64] Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S Yu, and Qingsong Wen. 2024. Large Language Models for Education: A Survey and Outlook. *arXiv preprint arXiv:2403.18105* (2024).

[65] Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 19206–19214.

[66] Junde Wu, Jiayuan Zhu, Yunli Qi, Jingkun Chen, Min Xu, Filippo Menolascina, and Vicente Grau. 2024. Medical Graph RAG: Towards Safe Medical Large Language Model via Graph Retrieval-Augmented Generation. *arXiv preprint arXiv:2408.04187* (2024).

[67] Shangyu Wu, Ying Xiong, Yufei Cui, Haolun Wu, Can Chen, Ye Yuan, Lianming Huang, Xue Liu, Tei-Wei Kuo, Nan Guan, et al. 2024. Retrieval-augmented generation for natural language processing: A survey. *arXiv preprint arXiv:2407.13193* (2024).

[68] Chong Xiang, Tong Wu, Zexuan Zhong, David Wagner, Danqi Chen, and Prateek Mittal. 2026. Certifiably Robust RAG against Retrieval Corruption. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. arXiv:2405.15556 https://arxiv.org/abs/2405.15556

[69] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388* (2025).

[70] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 Technical Report. *arXiv preprint arXiv:2412.15115* (2024).

[71] Zihao Yang, Mosh Levy, Yoav Goldberg, and Byron C. Wallace. 2026. Compared to What? Baselines and Metrics for Counterfactual Prompting. https://doi.org/10.48550/arXiv.2605.01048 arXiv:2605.01048 [cs.CL]

[72] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600* (2018).

[73] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of Retrieval-Augmented Generation: A Survey. *arXiv preprint arXiv:2405.07437* (2024).

[74] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176* (2025).

[75] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473* (2024).

[76] Yihang Zheng, Bo Li, Zhenghao Lin, Yi Luo, Xuanhe Zhou, Chen Lin, Jinsong Su, Guoliang Li, and Shifu Li. 2024. Revolutionizing Database Q&A with Large Language Models: Comprehensive Benchmark and Evaluation. *arXiv preprint arXiv:2409.04475* (2024).

[77] Xuanhe Zhou, Guoliang Li, Zhaoyan Sun, Zhiyuan Liu, Weize Chen, Jianming Wu, Jiesi Liu, Ruohang Feng, and Guoyang Zeng. 2024. D-bot: Database diagnosis system using large language models. *Proceedings of the VLDB Endowment* 17, 10 (2024), 2514–2527.

[78] Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Taotao Wang, Runyuan He, Yongwei Zhang, Sicong Liang, Xilin Liu, Yuchi Ma, and Yixiang Fang. 2026. In-Depth Analysis of Graph-Based RAG in a Unified Framework. *Proceedings of the VLDB Endowment* 18, 13 (Jan. 2026), 5623–5637. https://doi.org/10.14778/3773731.3773738

[79] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2025. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. In *34th USENIX Security Symposium (USENIX Security 25)*. 3827–3844. https://www.usenix.org/conference/usenixsecurity25/presentation/zou

A ADDITIONAL DISCUSSIONS

A.1 Counterfactual intervention protocol

In this subsection, we introduce the complete pipeline for constructing and validating the counterfactual-intervention and meaning-preserving replacement pools used in our evaluation.

Pipeline. Our pipeline consists of six steps. First, we tokenize the retrieved chunks and identify candidate tokens that can be replaced. For each token, we record its containing sentence and use spaCy to identify its entity type. Second, we prompt an LLM to generate several single-token counterfactual alternatives for each candidate. Each alternative must preserve the token's entity type and role in the sentence while keeping the sentence grammatical and coherent. Next, we remove alternatives that contain multiple tokens, break the sentence grammar, or do not match the original entity type. We classify the remaining alternatives as `COUNTERFACTUAL`, `PARAPHRASE`, or `INVALID`, and retain only those labeled `COUNTERFACTUAL`. The resulting replacement pool is fixed and shared by all methods. We then generate meaning-preserving alternatives, such as synonyms or aliases, for each selected token that is not a number or date. These alternatives are generated using WordNet [41] and an LLM, and only those that preserve the original meaning are retained. The generation prompts and validation rules for the two pools are given below.

A.1.1 Counterfactual Generation and Validation. We construct the counterfactual pool for all candidate tokens before running any TCE method. For each target token, we mark its position with `[[double brackets]]` in the complete sentence and provide the LLM with its entity type. Prompt P1 asks the editor to generate several single-token alternatives that retain this contextual role but express a different factual value.

P1 Counterfactual Intervention: Generation

SYSTEM You generate meaningful factual counterfactual tokens. Preserve type and grammar, but change information. Never return synonyms. Output strict JSON only.

USER For every target, generate factual counterfactual replacements for the token enclosed in `[[double brackets]]`. A replacement must be one token, retain the target's contextual semantic category and grammatical role, and make the sentence express an incompatible factual alternative. It must not be a synonym, alias, paraphrase, spelling variant, inflection, generic placeholder, hypernym, hyponym, deletion, or random noise. For a token inside a multi-token named entity, preserve its name slot. Type hints may be wrong; infer the actual type from the marked sentence and complete entity. Return a compact strict JSON object mapping each short ID to its candidate list.

INPUT `{"id": "...", "token": "...", "semantic_type_hint": "...", "part_of_speech": "...", "fine_grained_tag": "...", "complete_entity": "...", "entity_token_position": "...", "marked_sentence": "...", "forbidden": [...] }`

Each generated candidate passes two validation stages. First, deterministic checks reject empty or multi-token outputs, unchanged forms, malformed text, and candidates that violate the token's grammatical or entity slot. Second, Prompt P2 determines whether the remaining edit changes the factual value while preserving the original semantic role.

P2 Counterfactual Intervention: Validation

SYSTEM You perform conservative semantic-relation classification for token edits. Output strict JSON only.

USER Classify each one-token edit using only the marked sentence. Return COUNTERFACTUAL only if the candidate keeps the same fine-grained semantic slot and grammar while asserting a different, incompatible factual value. Return PARAPHRASE for synonyms, aliases, inflections, hypernyms, hyponyms, and broader or narrower restatements. Return INVALID for a different semantic role, malformed text, random noise, or an ungrammatical sentence. Return one compact strict JSON object mapping each ID to exactly one label.

INPUT `{"id": "...", "original": "...", "candidate": "...", "semantic_type_hint": "...", "part_of_speech": "...", "complete_entity": "...", "entity_token_position": "...", "marked_sentence": "..." }`

We retain only candidates labeled COUNTERFACTUAL. The accepted alternatives are ordered deterministically and stored with their token positions in a shared pool. Consequently, whenever two methods select the same token, they apply the same counterfactual replacement.

A.1.2 Meaning-Preserving Replacement. Specifically, for each selected token, we construct a meaning-preserving replacement. We first retrieve its candidate synonyms from WordNet [41] and then provide the token, its sentence, and the corresponding metadata to the LLM. As shown in Prompt P3, the LLM generates synonyms or aliases that change the surface form while preserving the original factual meaning. Since numbers and dates generally have no valid synonymous replacements, we leave them unchanged.

P3 Meaning-Preserving Replacement: Generation

SYSTEM You generate contextual synonyms and aliases without changing factual meaning. Output strict JSON only.

USER For each token enclosed in `[[double brackets]]`, generate meaning-preserving replacements that fit the original sentence. Each output must be a synonym, alias, abbreviation expansion, or equivalent rendering of the marked token. It must preserve the original referent, fact, relation, polarity, and grammatical role. Do not change any entity, value, date, or number. WordNet suggestions may be used when they fit the sentence. If the marked token belongs to a multi-token name, replace only the marked position. Return an empty list when no valid replacement exists; otherwise, return a strict JSON object mapping each ID to its candidate list.

INPUT `{"id": "...", "token": "...", "part_of_speech": "...", "semantic_type": "...", "marked_sentence": "...", "wordnet_suggestions": [...], "forbidden": [...] }`

Next, we merge the candidates obtained from WordNet and Prompt P3 and remove duplicated, unchanged, or malformed outputs. For each remaining candidate, we replace the marked token and construct the corresponding revised sentence. Prompt P4 then compares the original and revised sentences to determine whether the replacement preserves both factual meaning and grammatical correctness.

P4 Meaning-Preserving Replacement: Validation

SYSTEM You are a strict factual-equivalence auditor for contextual paraphrases. Output strict JSON only.

USER Compare the original and revised sentences for each record. Return true only when the candidate changes the surface wording while preserving the original referent, fact, relation, polarity, and grammatical role. Return false if the candidate changes the meaning, is broader or narrower than the original token, produces an ungrammatical sentence, or duplicates a neighboring word. Return a strict JSON object mapping each ID to true or false.

INPUT `{"id": "...", "original": "...", "candidate": "...", "part_of_speech": "...", "semantic_type": "...", "marked_sentence": "...", "revised_sentence": "..." }`

Finally, we retain only the candidates accepted by Prompt P4 and order them deterministically. We assign one candidate to each token; if none is accepted, we use a deterministic surface variation, such as changing the capitalization of a proper name.

Now, we are ready to present the contribution-graph construction algorithm, named ConstructContributionGraph. Its details are shown in Algorithm 1. It first constructs the node set and initializes the edge set (lines 1–2). It then runs one LLM forward pass on the original RAG input and response. During this pass, it records the forward message between every visible token pair at each Transformer layer (lines 2–5). Next, it computes the answer-support score from the response-token logits and performs one backward pass to obtain the corresponding answer gradients (line 5). For each receiver token at each layer, the algorithm computes the contribution of every incoming message and normalizes its positive contributions (lines 6–13). Finally, it sums the normalized contribution of each token pair across all layers, adds every pair with positive capacity to the edge set, and returns the resulting contribution graph (lines 14–18).

A.2 Limitations of Automatic Replacement Validation

Neither replacement pipeline provides an oracle guarantee. The factual editor may accept an alternative that subtly changes grammatical function or does not realize the intended semantic type, while the meaning-preserving editor may accept a synonym or alias whose interpretation shifts in context. Conversely, the validators may reject valid replacements. The answer-blind prompts, lexical checks, semantic audits, and shared replacement pools improve consistency and ensure that all methods are compared under the same replacement contract; they do not prove that every replacement is semantically perfect. An oracle-level guarantee would require human review of every token together with its containing sentence. We therefore interpret the reported results under the adopted automatic replacement contract and expose the complete prompts and replacement-pool hashes for auditing.

A.3 Detailed Evaluation-Metric Definitions

In this subsection, we provide the formal definitions of the two groups of flip-rate metrics introduced in Section 6.1. Let Q denote the evaluation queries, y_q the gold answer of query q, and T_q the token set returned by an evaluated method. We denote the responses before and after the counterfactual intervention by

$$\mathcal{R}_q^{\text{orig}} := \mathcal{R}_q(C), \quad \mathcal{R}_q^{\text{CF}} := \mathcal{R}_q\big(C[T_q \leftarrow \Psi(T_q)]\big) \tag{17}$$

respectively.

Flip-rate metrics. The Answer Flip Rate (Ans-FR) measures the percentage of queries whose responses change after the counterfactual intervention:

$$\text{Ans-FR} = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{I}\Big[\mathcal{R}_q^{\text{orig}} \not\equiv \mathcal{R}_q^{\text{CF}}\Big]. \tag{18}$$

Ans-FR compares only the two generated responses and does not use y_q.

To measure whether the intervention changes a correct response into an incorrect one, let $s_c(\mathcal{R}, y_q) \in \{0, 1\}$ indicate whether response $\mathcal{R}$ is correct under criterion $c \in \{\text{EM}, \text{Acc}\}$. We first construct the corresponding originally correct query set

$$Q_c = \Big\{q \in Q : s_c(\mathcal{R}_q^{\text{orig}}, y_q) = 1\Big\} \tag{19}$$

and then compute

$$\mathrm{FR}_c^{\mathrm{CF}} = \frac{1}{|Q_c|} \sum_{q \in Q_c} \mathbb{I}\big[s_c(\mathcal{R}_q^{\mathrm{CF}}, y_q) = 0\big]. \tag{20}$$

We report $\mathrm{FR}_{\mathrm{EM}}^{\mathrm{CF}}$ and $\mathrm{FR}_{\mathrm{Acc}}^{\mathrm{CF}}$ as EM-FR and Acc-FR, respectively. Thus, their denominators contain only queries answered correctly before the intervention.

Counterfactual flip-rate metrics. A response may also change after a meaning-preserving intervention. To measure this surface-form sensitivity, let $T_q^{\mathrm{MP}} \subseteq T_q$ contain the selected tokens that are neither numbers nor dates. We replace these tokens with their meaning-preserving alternatives, leave the remaining selected tokens unchanged, and obtain

$$\mathcal{R}_q^{\mathrm{MP}} := \mathcal{R}_q\Big(C[T_q^{\mathrm{MP}} \leftarrow \Psi^{\mathrm{Syn}}(T_q^{\mathrm{MP}})]\Big) \tag{21}$$

as the meaning-preserving response. Its answer and correctness-conditioned flip rates are

$$\text{Ans-FR}^{\mathrm{MP}} = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{I}\Big[\mathcal{R}_q^{\mathrm{orig}} \not\equiv \mathcal{R}_q^{\mathrm{MP}}\Big] \tag{22}$$

and

$$\mathrm{FR}_c^{\mathrm{MP}} = \frac{1}{|Q_c|} \sum_{q \in Q_c} \mathbb{I}\big[s_c(\mathcal{R}_q^{\mathrm{MP}}, y_q) = 0\big], \qquad c \in \{\mathrm{EM}, \mathrm{Acc}\}. \tag{23}$$

Following Section 6.1, we subtract each meaning-preserving flip rate from its counterfactual counterpart:

$$\begin{aligned} \text{Ans-CFR} &= \text{Ans-FR} - \text{Ans-FR}^{\mathrm{MP}}, \\ \text{EM-CFR} &= \mathrm{FR}_{\mathrm{EM}}^{\mathrm{CF}} - \mathrm{FR}_{\mathrm{EM}}^{\mathrm{MP}}, \\ \text{Acc-CFR} &= \mathrm{FR}_{\mathrm{Acc}}^{\mathrm{CF}} - \mathrm{FR}_{\mathrm{Acc}}^{\mathrm{MP}}. \end{aligned} \tag{24}$$

All reported values are multiplied by 100 and expressed as percentages. A larger CFR indicates that the observed response change is more attributable to the changed factual information than to surface-form sensitivity.

Finally, token-level F1 and normalized EM produce identical binary flip decisions for our short-answer outputs, so we report only EM. We omit Acc-FR and Acc-CFR for PQA because its three-way categorical labels do not use the answer-containment Acc criterion adopted for the other datasets. For the binary-choice QTZ dataset, canonical choice normalization makes EM and Acc identical; therefore, we omit the redundant Acc metrics.

B ADDITIONAL PROOFS

This section provides the complete proofs of the results stated in Section 5.

B.1 Proof of Lemma 5.2

Proof. Fix any $T \subseteq \mathtt{Token}(C)$. Cutting the selection edges $\{e_i : i \in T\}$ removes all paths through the tokens in T and costs $\lambda|T|$. Contracting every remaining selection edge recovers $\mathcal{G}_q \setminus T$. By max-flow/min-cut duality, its contribution edges can be separated at cost $\Phi_q(\mathcal{G}_q \setminus T)$. The union of these two edge sets is therefore an s–t cut in $\mathcal{F}_q^{\lambda}$, and

$$\mathtt{mincut}(\mathcal{F}_q^{\lambda}) \;\le\; \lambda|T| + \Phi_q(\mathcal{G}_q \setminus T).$$

As T was arbitrary, the minimum cut value is at most the fixed-price optimum stated in Lemma 5.2.

For the reverse direction, let $K^\star$ be a minimum s–t cut in $\mathcal{F}_q^{\lambda}$. Because every terminal edge has infinite capacity, $K^\star$ contains only selection and contribution edges. Let

$$T_\lambda := \{i \in \mathtt{Token}(C) : e_i \in K^\star\}, \qquad K_q^\star := K^\star \cap \mathcal{E}_q.$$

The contribution edges in $K_q^\star$ must separate every context-to-answer path in $\mathcal{G}_q \setminus T_\lambda$. Otherwise, such a path could be extended through the uncut selection and terminal edges to form an s–t path avoiding $K^\star$, contradicting that $K^\star$ is a cut. Hence,

$$\sum_{e \in K_q^\star} c_q(e) \;\ge\; \Phi_q(\mathcal{G}_q \setminus T_\lambda),$$

and thus

$$\begin{aligned} \mathtt{mincut}(\mathcal{F}_q^{\lambda}) &\ge \lambda|T_\lambda| + \Phi_q(\mathcal{G}_q \setminus T_\lambda) \\ &\ge \min_{T \subseteq \mathtt{Token}(C)} \big\{\lambda|T| + \Phi_q(\mathcal{G}_q \setminus T)\big\}. \end{aligned}$$

Combining the two directions proves the equality stated in Lemma 5.2. It also shows that T_λ minimizes the fixed-price objective and therefore satisfies Equation (13). □

B.2 Proof of Lemma 5.3

Proof. The empty set has objective value $\Phi_q(\mathcal{G}_q)$. For every nonempty set T and every $\lambda \ge \lambda_q^{\max}$,

$$f_T(\lambda) = \lambda|T| + \Phi_q(\mathcal{G}_q \setminus T) \ge \lambda_q^{\max}.$$

The minimum-cardinality tie rule therefore selects the empty set at $\lambda_q^{\max}$ and at every larger price.

There are only finitely many token sets, and their objective lines have only finitely many intersection points. Between two consecutive intersection points, the ordering of all objective values is unchanged. Together with the deterministic tie rule, this implies that the same set T_λ is returned throughout each resulting interval.

It remains to order these intervals. Let $T_a = T_{\lambda_a}$ and $T_b = T_{\lambda_b}$ for $\lambda_a < \lambda_b$. Write $k_a = |T_a|$, $k_b = |T_b|$, $\phi_a = \Phi_q(\mathcal{G}_q \setminus T_a)$, and $\phi_b = \Phi_q(\mathcal{G}_q \setminus T_b)$. Optimality at the two prices gives

$$\begin{aligned} \lambda_a k_a + \phi_a &\le \lambda_a k_b + \phi_b, \\ \lambda_b k_b + \phi_b &\le \lambda_b k_a + \phi_a. \end{aligned}$$

After rearranging and combining them, we obtain

$$(\lambda_b - \lambda_a)(k_b - k_a) \le 0,$$

and hence $k_a \ge k_b$. If two different returned sets had the same cardinality, their objective lines would be parallel. The set with the smaller residual flow would dominate at every price; if both residual flows were equal, the deterministic tie rule would return the same set at every price. Thus, two distinct candidates must satisfy $k_a > k_b$.

Substituting $k_a > k_b$ into either optimality inequality gives $\phi_a \le \phi_b$. Equality is impossible: at a positive price, T_b would strictly dominate T_a, while at price zero the minimum-cardinality tie rule would again select T_b. Therefore, $\phi_a < \phi_b$, proving the strict inequalities stated in Lemma 5.3. Finally, each candidate has an integer cardinality in $\{0, 1, \ldots, n\}$, and this cardinality decreases strictly from one interval to the next. There can consequently be at most $n + 1$ intervals and $|\mathcal{T}_q| \le n + 1$. □

B.3 Proof of Theorem 5.5

Proof. We prove two properties: (1) every enumerated set is an exact fixed-price solution, and (2) the recursion terminates without missing any such solution.

1) Every enumerated set is exact. The left endpoint is obtained by a minimum-cut computation at $\lambda = 0$; by Lemma 5.2, it is exactly T_0. The right endpoint is the empty set, which is exactly $T_{\lambda_q^{\max}}$ by Lemma 5.3. Every other recorded set is returned by a minimum-cut computation at a price τ and is therefore the exact optimizer T_τ. Hence, the algorithm introduces no spurious candidate.

2) No exact solution is missed. All distinct solutions occur in $[0, \lambda_q^{\max}]$ by Lemma 5.3. Consider a call to `Divide` with endpoint records (a, T_A, ϕ_A) and (b, T_B, ϕ_B), where $T_A = T_a \neq T_B = T_b$. Let $k_A = |T_A|$ and $k_B = |T_B|$. The same lemma gives $k_A > k_B$. Optimality of the two endpoint solutions implies

$$a(k_A - k_B) \leq \phi_B - \phi_A \leq b(k_A - k_B).$$

Thus, the dividing price $\tau = \tau(T_A, T_B)$ lies in $[a, b]$. It cannot equal a: at that price T_A and T_B have equal objective values, and the smaller-cardinality rule would select T_B instead of T_A. Therefore, $a < \tau \leq b$.

Let $T_M = T_\tau$ and $k_M = |T_M|$. If $\tau = b$, then $T_M = T_b = T_B$. Suppose now that $\tau < b$. If $k_M > k_A$, then $f_{T_M}(\tau) \leq f_{T_A}(\tau)$ implies $f_{T_M}(a) < f_{T_A}(a)$, contradicting the optimality of T_A at a. Similarly, if $k_M < k_B$, then $f_{T_M}(\tau) \leq f_{T_B}(\tau)$ implies $f_{T_M}(b) < f_{T_B}(b)$, contradicting the optimality of T_B at b. Hence, $k_B \leq k_M \leq k_A$.

If $k_M = k_A$, the objective lines of T_M and T_A have the same slope. Their optimality at τ and a, respectively, forces these two lines to coincide. At τ, this common value also equals that of T_B, whose cardinality is smaller; hence, the tie rule gives $T_M = T_B$. If $k_M = k_B$, the endpoint optimality conditions similarly force the objective lines of T_M and T_B to coincide, and the deterministic tie rule again gives $T_M = T_B$. Consequently,

$$T_M = T_B \qquad \text{or} \qquad k_B < k_M < k_A.$$

We next show that the first outcome is a valid stopping condition. Assume $T_M = T_B$. For any token set S, endpoint optimality gives $f_S(a) \geq f_{T_A}(a)$ and $f_S(\tau) \geq f_{T_A}(\tau)$. Because $f_S - f_{T_A}$ is linear in λ, it is nonnegative throughout $[a, \tau]$; hence, T_A is optimal on this entire interval. Applying the same argument to T_B at τ and b shows that T_B is optimal throughout $[\tau, b]$. Any objective line tied at an interior price must coincide with the corresponding endpoint line, in which case the deterministic rule selects the same set. Thus, no distinct T_λ is hidden inside (a, b), and the recursion can safely stop.

Otherwise, T_M is a new solution with $k_B < k_M < k_A$. The two recursive calls have exact endpoint solutions, and every remaining price belongs to one of their two subintervals. Moreover, their cardinality gaps, $k_A - k_M$ and $k_M - k_B$, are both strictly smaller than $k_A - k_B$. Induction on this nonnegative integer gap shows that every recursive call terminates and enumerates every solution in its interval. Applied to the initial interval $[0, \lambda_q^{\max}]$, this proves that the enumeration phase terminates and computes exactly all distinct T_λ. □

C ADDITIONAL EXPERIMENTS

This section provides additional experimental details and complete results for the experiments in Section 6. Unless stated otherwise, all experiments use the same query sets, retrieval outputs, counterfactual intervention pool, target LLM, and implementation settings described in Section 6.1.

C.1 Datasets

We convert every benchmark to the same query–corpus interface while preserving its official questions and gold answers. For each dataset, we sample 1,000 queries and retrieve from the associated corpus summarized in Table 1. Specifically, the datasets include:

- **HotpotQA** [72] is a Wikipedia-based multi-hop QA benchmark. Its questions require combining evidence from multiple documents, and its supporting-fact annotations make it suitable for testing whether a token explanation captures distributed answer evidence.
- **TimeQA** [7] evaluates time-sensitive question answering. Questions contain explicit or implicit temporal constraints, so the correct answer depends on locating evidence associated with the requested date or period rather than retrieving a timeless fact.
- **FinQA** [8] requires numerical reasoning over financial reports. Evidence may occur in report text or tables; we expose the corresponding report content to retrieval while retaining the benchmark answers for evaluation.
- **MuSiQue** [61] constructs multi-hop questions by composing connected single-hop questions while reducing shortcut cues. It therefore tests whether `ReFlow` can identify evidence that participates in a longer compositional reasoning chain.
- **QuaRTz** [57] is a binary-choice benchmark for qualitative scientific relationships. Answering requires applying textual rules about how one quantity increases or decreases with another; we use canonical choice normalization when computing its flip metrics.
- **TriviaQA** [27] is a large-scale open-domain trivia benchmark paired with distantly supervised evidence documents. Its long retrieval contexts and lexical variation stress both evidence localization and compact intervention.
- **2WikiMultiHopQA** [22] is a Wikipedia-based multi-hop benchmark constructed to expose intermediate reasoning steps. Questions connect evidence across articles through bridge, inference, and comparison relations.
- **PopQA** [40] contains entity-centric, open-domain factual questions spanning popular and long-tail entities. It tests whether retrieved non-parametric evidence can identify answer support beyond facts already likely to be stored in model parameters.

C.2 Additional Experimental Results

In this subsection, we report the complete results that complement the experiments in Sections 6.3–6.5.

Exp.1. Effectiveness of contract-priced search. In this experiment, we extend the comparison in Figure **??** to FQA, QTZ, TrQA, and PQA. As in the main experiment, `One-shot` solves the fixed-price problem only at $\lambda = 1$, `Grid` tests 33 logarithmically spaced prices, and `ReFlow` enumerates all candidate sets induced by the complete price range. The three methods use the same 1,000 queries, contribution graphs, counterfactual intervention pool, and target LLM within each dataset.

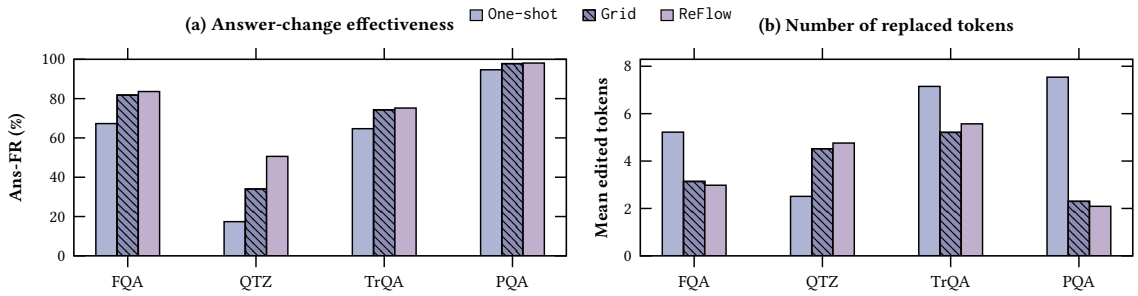


Figure 11: Contract-priced-search variants on the remaining four datasets. All variants use the same retrieval results, counterfactual intervention pool, and target LLM within each dataset.

Table 3: Aggregate retrieval-depth results across all eight datasets.

k	Ans-FR	EM-FR	Acc-FR	Tok.	Cov.	Nodes	Edges	E2E (s)
1	75.2	70.5	66.3	2.19	96.9	99.5	309.3	2.66
3	81.2	71.6	65.7	2.93	98.8	303.5	509.0	7.01
5	85.2	78.7	72.1	3.08	97.0	455.5	506.3	11.94
10	82.0	69.0	63.0	3.57	99.6	1,042.4	614.4	24.50

Figure 11 supports two observations. *1) Exact enumeration consistently finds more effective interventions than a numerical grid.* It improves Ans-FR over `Grid` by 1.8, 16.6, 0.9, and 0.3 percentage points on FQA, QTZ, TrQA, and PQA, respectively. *2) The improvement does not require substantially larger interventions.* `ReFlow` uses fewer tokens than `Grid` on FQA and PQA; on QTZ and TrQA, the gains require only 0.24 and 0.36 additional tokens. Together with the main-text results, this shows that a single price may over-select tokens, while a fixed grid may miss useful candidate sets.

Exp.2. Effect of retrieval depth k. We provide the complete results for the retrieval-depth experiment in Section 6.5. We vary $k \in \{1, 3, 5, 10\}$ while keeping the target LLM, contribution tracing, and all other settings unchanged. Figure 12 reports Ans-FR, EM-FR, Acc-FR, and their three control-adjusted counterparts on all eight datasets. For every setting, we rerun the LLM after applying meaning-preserving replacements to exactly the same selected token positions, which provides the paired controls required by the three CFR metrics. Table 3 additionally reports the aggregate factual metrics and operational costs.

We make two observations. *1) The default $k = 5$ gives the strongest overall effectiveness.* Its aggregate Ans-FR, EM-FR, and Acc-FR are 85.2%, 78.7%, and 72.1%, while the corresponding CFRs are 61.1%, 64.8%, and 60.0%. Increasing the depth to $k = 10$ does not improve any aggregate metric; in particular, EM-CFR and Acc-CFR decrease by 8.4 and 7.5 percentage points. *2) Increasing k raises the computational and intervention costs.* From $k = 1$ to $k = 10$, the mean intervention size increases from 2.19 to 3.57 tokens, the mean graph size from 99.5 to 1,042.4 nodes, and the end-to-end time from 2.66 to 24.50 seconds per query. These results support $k = 5$ as the default effectiveness–cost tradeoff.

Exp.3. Effect of receiver beam d. We next provide additional diagnostics for the receiver-beam experiment in Section 6.5. We vary $d \in \{16, 32, 48, 64\}$, where d is the maximum number of positive incoming contributions retained for each receiver token. Figure 13 provides one panel for each of the six metrics. Each CFR value uses a paired meaning-preserving intervention over the same token positions selected under the corresponding receiver beam. Table 4 reports the corresponding graph sizes.

We make two observations. *1) Increasing d improves both FR and CFR up to $d = 48$.* From $d = 16$ to $d = 48$, aggregate Ans-FR, EM-FR, and Acc-FR increase by 24.8, 26.6, and 25.0 percentage points, while Ans-CFR, EM-CFR, and Acc-CFR increase by 23.3, 24.0, and 22.6 points. Increasing d further to 64 does not improve any aggregate metric and reduces the three CFRs by 1.1–5.3 points. *2) Larger beams substantially increase graph size.* From $d = 16$ to $d = 64$, the average number of active nodes and flow edges grows by 3.9× and 9.9×, respectively. We therefore use $d = 48$ by default.

Exp.4. Effect of LLM backbone. We evaluate whether `ReFlow` depends on one model family by repeating the complete protocol with Qwen2.5-7B-Instruct, Llama-3.1-8B-Instruct, Mistral-Nemo-Instruct-2407, and Qwen3-4B-Instruct-2507. All models use the same queries, retrieved chunks, and counterfactual intervention protocol.

Table 5 reports all six metrics for every backbone–dataset pair. `ReFlow` remains effective across the four model families: on the seven open-ended datasets, the lowest Ans-FR and Ans-CFR are 48.9% and 47.8%, respectively. QTZ is consistently harder because its binary choice protocol makes response changes less frequent. No backbone is uniformly best across the 48 dataset–metric combinations, showing that the result is not specific to the default Qwen2.5 model.

Exp.5. Additional RAG provenance results. We extend the provenance experiment in Section 6.4 to the remaining six datasets. `Full` prompts the LLM with all top-5 retrieved chunks. `ReFlow`, `MIRAGE`, and `ARC-JSD` instead use only the sentences containing their selected tokens, with the two baselines receiving the same per-query number of sentences as `ReFlow`.

Figure 14 shows that `ReFlow` achieves higher F1 than both attribution baselines on all six datasets, with gains of 0.8–9.8 percentage points over the stronger baseline. On TrQA, for example, `ReFlow` obtains 71.5 F1, close to the 74.5 of `Full`, while reducing the mean context from 1,566.6 to 127.9 tokens. Under the same sentence budget, `MIRAGE` and `ARC-JSD` obtain only 61.7 and 44.6 F1. These results further show that the sentences identified by `ReFlow` retain the information needed to reproduce the full-context response.

Exp.6. Contribution-graph size. Finally, we report the average contribution-graph size for each of the eight 1,000-query workloads.

Table 6: Mean contribution-graph size per query.

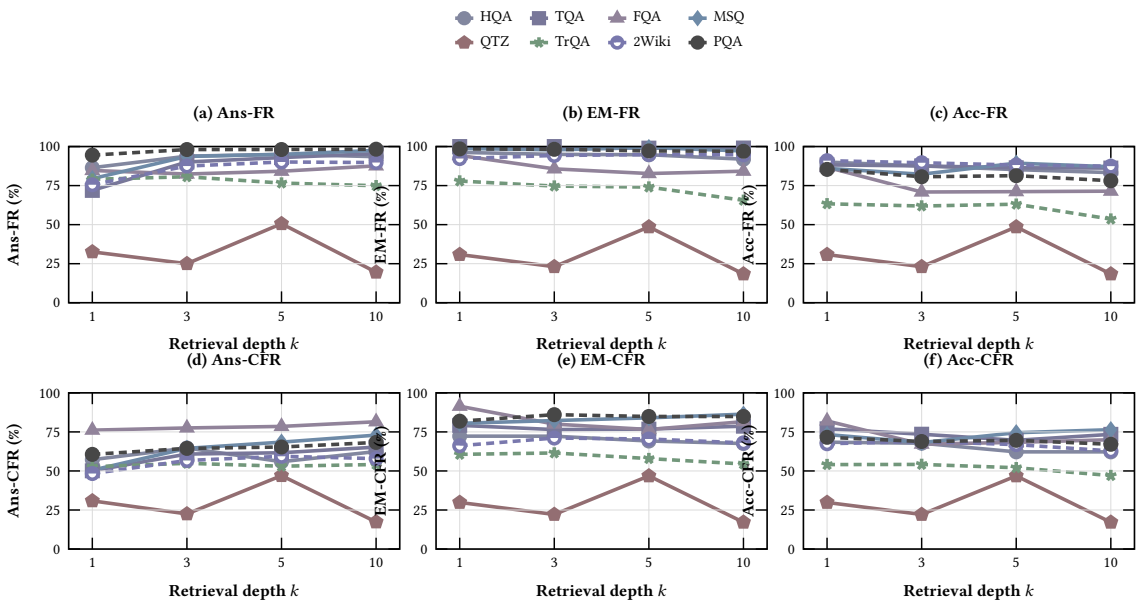


Figure 12: Six-metric sensitivity to retrieval depth.

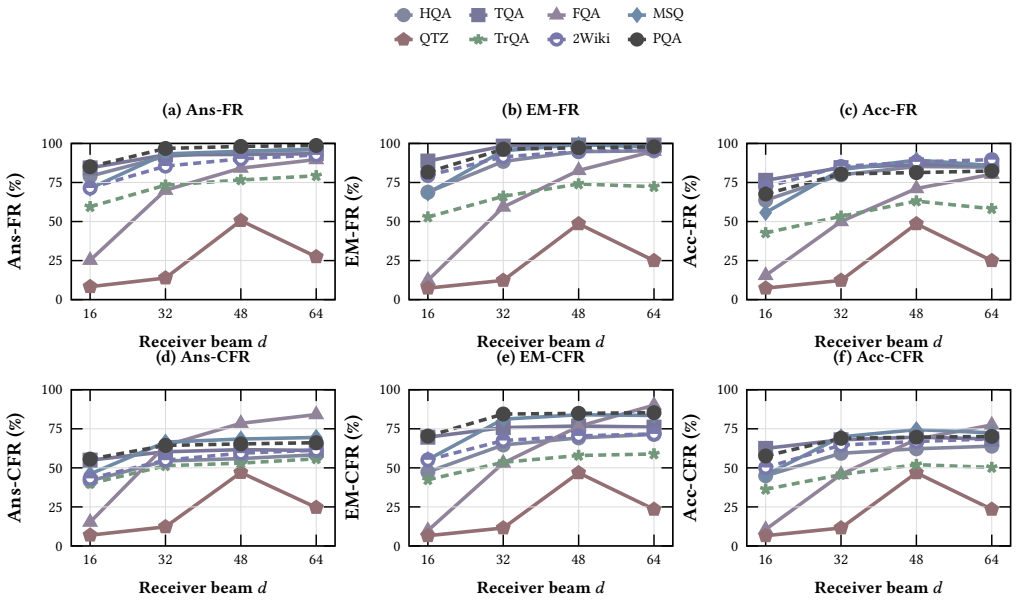


Figure 13: Six-metric sensitivity to receiver beam.

Dataset	n	m	d_{avg}
HotpotQA	427	660	1.55
TimeQA	305	964	3.16
FinQA	834	188	0.23
MuSiQue	457	718	1.57
QuARTz	73	203	2.78
TriviaQA	1297	819	0.63
2Wiki	395	467	1.18
PopQA	316	568	1.80

Table 6 shows that the contribution graphs remain sparse: they contain 73–1,297 nodes and 188–964 edges, with average out-degree between 0.23 and 3.16. TrQA has the most nodes, whereas TQA has the largest average out-degree. This sparsity is consistent with the running-time results in Figure ??, where candidate enumeration adds little overhead beyond contribution-graph construction.

Table 4: Graph size under each receiver beam. Columns n and m report the mean number of active nodes and positive flow edges, respectively.

Dataset	$d = 16$		$d = 32$		$d = 48$		$d = 64$	
	n	m	n	m	n	m	n	m
HQA	1,236.5	3,160.8	2,445.4	9,524.9	3,639.2	18,889.0	4,819.2	31,197.6
TQA	1,242.4	3,166.5	2,465.2	9,616.2	3,671.4	19,078.0	4,861.5	31,494.2
FQA	1,217.5	3,119.2	2,375.2	9,312.1	3,574.6	18,615.0	4,781.9	31,051.9
MSQ	1,241.7	3,177.5	2,454.2	9,574.8	3,652.7	18,977.1	4,833.1	31,346.2
QTZ	1,241.2	3,166.4	2,485.6	9,664.0	3,652.1	18,858.6	4,885.2	31,533.1
TrQA	1,239.0	3,164.1	2,450.6	9,583.1	3,644.1	18,994.1	4,830.8	31,410.4
2Wiki	1,240.5	3,184.8	2,442.8	9,543.6	3,624.2	18,868.4	4,796.9	31,165.6
PQA	1,235.5	3,141.0	2,445.7	9,483.2	3,622.9	18,725.4	4,769.1	30,865.6

Table 5: Detailed cross-model results. Darker shading indicates higher flip rates, and bold marks the best backbone.

Dataset	Metric	Qwen2.5 7B	Llama3.1 8B	Mistral 12B	Qwen3 4B
HotpotQA	Ans-FR	**95.1**	91.9	88.4	75.7
	EM-FR	**94.8**	89.1	85.2	70.4
	Acc-FR	**85.2**	79.4	78.8	62.5
	Ans-CFR	56.5	76.1	**76.2**	64.8
	EM-CFR	69.7	**79.3**	74.9	62.9
	Acc-CFR	62.4	69.5	**69.8**	54.8
TimeQA	Ans-FR	93.4	**98.0**	97.0	87.8
	EM-FR	99.0	**99.1**	98.0	93.9
	Acc-FR	86.4	85.2	**88.4**	85.8
	Ans-CFR	62.1	77.9	**82.2**	75.6
	EM-CFR	77.1	83.0	**86.3**	84.2
	Acc-CFR	70.4	74.5	78.7	**79.8**
FinQA	Ans-FR	84.9	**92.4**	90.4	48.9
	EM-FR	82.7	**83.3**	82.7	45.0
	Acc-FR	**71.1**	69.2	61.6	34.5
	Ans-CFR	79.2	**91.2**	89.6	47.8
	EM-CFR	76.5	**83.3**	82.7	45.0
	Acc-CFR	69.0	**69.2**	61.6	34.5
MuSiQue	Ans-FR	95.3	**96.4**	96.2	72.6
	EM-FR	99.3	**100.0**	99.4	80.0
	Acc-FR	89.4	88.3	**90.0**	66.0
	Ans-CFR	68.3	81.6	**84.1**	63.6
	EM-CFR	84.0	83.1	**87.3**	69.6
	Acc-CFR	74.5	72.3	**80.0**	59.0

Dataset	Metric	Qwen2.5 7B	Llama3.1 8B	Mistral 12B	Qwen3 4B
QuaRTz	Ans-FR	25.1	**33.0**	20.8	11.1
	EM-FR	22.8	**31.3**	21.1	10.3
	Acc-FR	22.8	**31.3**	21.1	10.3
	Ans-CFR	22.1	**28.7**	20.1	10.3
	EM-CFR	21.0	**28.5**	20.7	9.8
	Acc-CFR	21.0	**28.5**	20.7	9.8
TriviaQA	Ans-FR	**76.8**	70.8	60.9	57.7
	EM-FR	**70.2**	60.6	53.6	50.9
	Acc-FR	**57.0**	48.3	45.2	41.7
	Ans-CFR	50.6	**60.8**	55.2	50.3
	EM-CFR	**55.9**	54.8	50.7	46.4
	Acc-CFR	**48.4**	44.5	43.2	38.3
2Wiki	Ans-FR	**90.1**	87.4	81.8	64.8
	EM-FR	**94.8**	87.7	73.5	70.8
	Acc-FR	**88.2**	78.3	71.5	67.2
	Ans-CFR	59.6	**76.5**	69.1	57.4
	EM-CFR	71.8	**80.2**	65.8	65.4
	Acc-CFR	68.6	**71.3**	63.4	61.2
PopQA	Ans-FR	**98.1**	95.0	93.9	73.7
	EM-FR	**97.2**	90.2	89.3	58.5
	Acc-FR	**81.3**	70.2	74.0	48.8
	Ans-CFR	65.2	74.8	**76.5**	59.0
	EM-CFR	**84.7**	84.3	83.2	52.6
	Acc-CFR	**69.6**	64.0	68.2	42.6

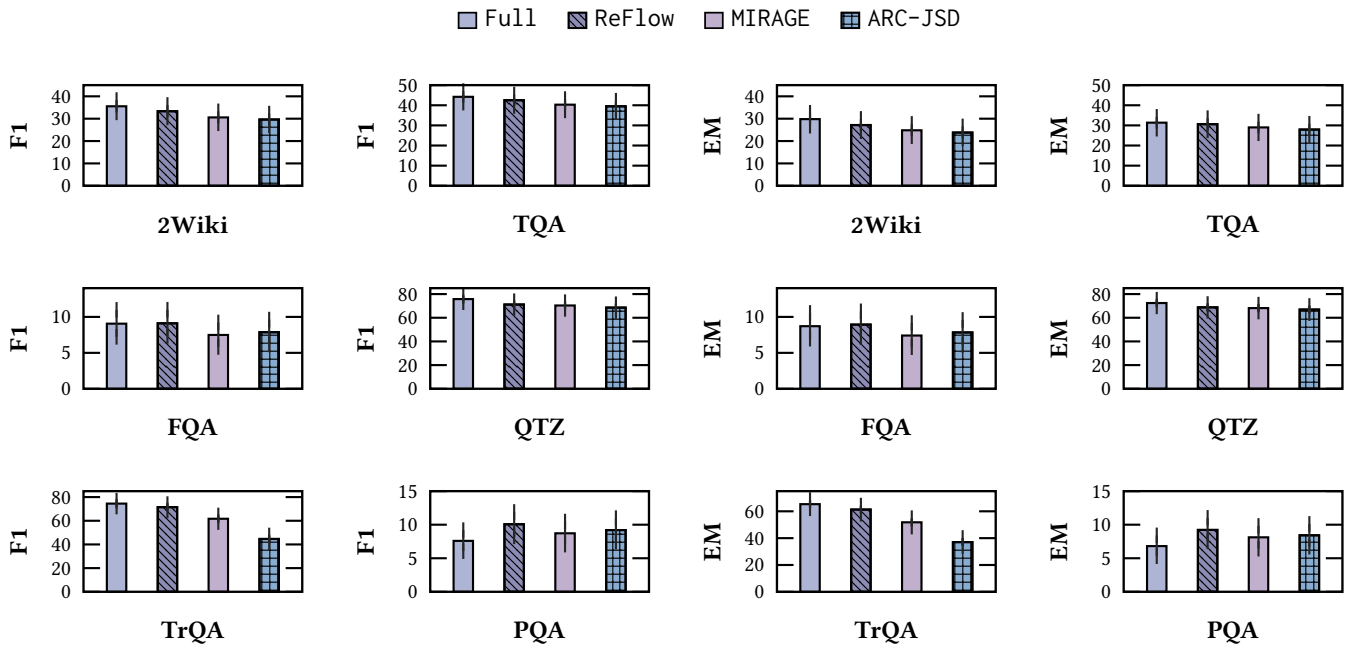


Figure 14: Illustration of RAG provenance.