E-Verify: A Paradigm Shift to Scalable Embedding-based Factuality Verification

Zeyang Liu, Jingfeng Xue, Xiuqi Yang, Wenbiao Du,
Jiarun Fu, Junbao Chen, Wenjie Guo, Yong Wang
Beijing Institute of Technology
liuzeyang@bit.edu.cn

Abstract

Large language models (LLMs) exhibit remarkable text-generation capabilities, yet struggle with factual consistency, motivating growing interest in factuality verification. Existing factuality verification methods typically follow a *Decompose-Then-Verify* paradigm, which improves granularity but suffers from poor scalability and efficiency. We propose a novel ***Decompose-Embed-Interact*** paradigm that shifts factuality verification from costly text-level reasoning to efficient alignment in embedding space, effectively mitigating the scalability bottlenecks and computational inefficiencies inherent to prior approaches. While the proposed paradigm promises scalable verification, its implementation faces three practical challenges: efficient decomposition, factually faithful embedding, and accurate verification in embedding space. To address these challenges, we introduce **E-Verify**, a lightweight framework that resolves them through three specially designed modules, each aligned with a specific stage of the paradigm and designed to preserve scalability and efficiency. Experiments demonstrate that E-Verify significantly improves both decomposition and verification efficiency while maintaining competitive accuracy. These results confirm that the proposed paradigm enables scalable and fine-grained factuality verification with minimal performance trade-offs. [1]

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in text generation tasks (Mann et al., 2020; Li et al., 2024; Iqbal et al., 2024; Fu et al., 2025). Nonetheless, LLMs often generate content with hallucinations, including incorrect dates, numerical errors, and fabricated relationships, which can mislead decision-making and exacerbate misinformation spread (Ji et al., 2023; Bang et al., 2023; Sadasivan et al., 2023). This raises an urgent need for factuality verification systems that can evaluate the factual consistency of LLM-generated content, especially in knowledge-intensive scenarios (Panchendrarajan and Zubiaga, 2024; Si et al., 2024; Atanasova, 2024).

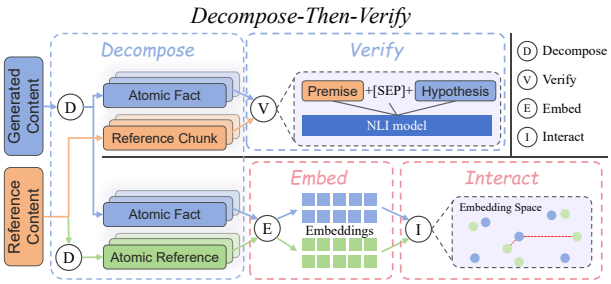

Figure 1: The top half shows the traditional *Decompose-Then-Verify* approach with costly pairwise NLI inference. The bottom half presents our proposed *Decompose-Embed-Interact* paradigm, which performs efficient verification via alignment in embedding space.

A dominant line of research in factuality verification [2] adopts the *Decompose-Then-Verify* paradigm, shown in Figure 1 (top half), which decomposes generated text into atomic facts and verifies them against reference sources using LLMs or natural language inference (NLI) models (Zhang and Bansal, 2021; Chern et al., 2023; Zhao et al., 2023; Tang et al., 2024). While this paradigm enhances granularity and interpretability, the inherent pairwise verification—where each fact must be individually compared to all reference segments—leads to quadratic computational overhead, which quickly becomes prohibitively expensive for long generations, posing a critical obstacle to scalability.

We begin with the observation that atomic facts

[1] Data and code are available at https://github.com/lzy248/e-verify

[2] Factuality verification is generally understood as a two-stage process comprising *evidence retrieval* and *evidence verification*. In this work, we primarily focus on the verification stage under the assumption that relevant evidence has already been provided.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 5759–5776
November 4-9, 2025

are typically short and structurally simple, making them well-suited for semantic embedding. This insight motivates a shift in verification strategy: instead of performing pairwise reasoning at the text level, we shift verification to alignment in embedding space. To this end, we propose the ***Decompose-Embed-Interact*** paradigm, shown in Figure 1 (bottom half), which reframes factuality verification as a modular process of atomic decomposition, independent embedding, and lightweight interaction. By encoding facts into dense vectors and verifying them efficiently in embedding space, this paradigm eliminates the need for costly LLM or NLI-based cross-encoding, enabling scalable and fine-grained consistency assessment.

While the proposed paradigm theoretically enables scalable factuality verification, its practical implementation poses several concrete challenges: how to decompose long-form text efficiently, how to preserve factual precision in embeddings, and how to conduct accurate verification in embedding space. To address these issues, we introduce **E-Verify**—an **E**fficient and **E**mbedding-based Factuality **Verifi**cation framework for LLMs. E-Verify operationalizes the proposed paradigm through three carefully designed modules: (1) A sentence-level atomic decomposer based on a fine-tuned small language model (SLM) improves decomposition efficiency for long-form text; (2) A Bi-Encoder embedder augmented with Pooling-based Multi-Head Attention enhances the factual fidelity of atomic fact embeddings beyond simple pooling; (3) A lightweight Multi-Feature Interaction Module verifies consistency through efficient embedding-level alignment, capturing both surface-level matching and directional factual discrepancy.

Experimental results confirm the effectiveness of our framework, demonstrating substantial gains in decomposition and verification efficiency while maintaining competitive accuracy. Importantly, our study reveals a key insight: embedding models, when paired with structured atomic decomposition and lightweight interaction modules, can deliver fine-grained factual verification performance previously thought to require deep cross-encoding—highlighting the potential of E-Verify as a scalable alternative to traditional NLI-based pipelines.

Our contributions can be summarized as:

- We introduce a novel ***Decompose-Embed-Interact*** paradigm that reframes factuality verification as an embedding-native task, transforming costly pairwise verification into efficient embedding-space alignment.
- We instantiate this paradigm in **E-Verify**, a lightweight and scalable framework that operationalizes embedding-native verification and overcomes key practical challenges, enabling efficient process.
- Experiments demonstrate that E-Verify substantially improves verification efficiency while maintaining strong accuracy, validating the paradigm's practical value.

2 Related Works

2.1 Hallucinations in LLMs

Hallucinations in LLMs, where models generate non-factual content such as temporal inconsistencies, numerical errors, or fabricated relationships, pose significant challenges to their reliability, particularly in knowledge-intensive tasks (Huang et al., 2023). Current strategies to mitigate hallucinations include training-phase interventions (e.g., knowledge distillation) (Gekhman et al., 2024; Abbas et al., 2023; McDonald et al., 2024; Huang et al., 2022), retrieval-augmented generation (RAG) approaches that integrate external knowledge during inference (Ram et al., 2023; Gao et al., 2022; Lewis et al., 2020), and post-hoc verification methods to assess factual consistency after text generation (Manakul et al., 2023; Dhuliawala et al., 2023; Maynez et al., 2020). While these methods aim to reduce hallucinations from various perspectives, another direction centers on factuality verification through explicit consistency checking against trusted reference sources.

2.2 Factuality Verification

Factuality verification, also referred to as fact-checking, typically involves comparing generated content with a trusted reference source. FactScore (Min et al., 2023) proposed a two-stage method that was later abstracted into the widely adopted *Decompose-Then-Verify* paradigm: first decomposing the generated text into atomic facts and then verifying each fact against references. An atomic fact refers to a minimal, self-contained unit that expresses a single verifiable proposition. Recent methods have extended this paradigm in various directions. FGLR (Stacey et al., 2024) enhances NLI-based reasoning by generating auxiliary premise facts, while FineSumFact (Oh et al., 2025) uses

fine-grained LLM feedback to supervise factuality in summarization. While this paradigm improves granularity, it suffers from poor scalability due to reliance on costly LLM APIs and quadratic complexity in pairwise verification between facts and references. MiniCheck (Tang et al., 2024) explores a more efficient solution by training a small NLI verifier on synthetic data, significantly reducing inference cost. While it eliminates dependency on LLM APIs, it still performs pairwise verification between each generated fact and all reference segments via transformer-based NLI models, resulting in non-trivial computation cost—especially as the number of facts and reference chunks grows.

3 Decompose-Embed-Interact Paradigm

We begin with a central observation: atomic facts are structurally simple and semantically compact, typically taking the form of short declarative sentences expressing a single verifiable proposition (see Figure 2, Stage 1). This localized, context-independent structure aligns well with modern sentence embedding models, which are designed to encode bounded propositions into fixed-length vectors. Such simplicity allows atomic facts to be faithfully compressed into embeddings with minimal semantic loss, making factuality verification possible through lightweight embedding-level interactions. Crucially, this enables scalable verification by avoiding the quadratic cost of cross-encoding each reference-fact pair.

Motivated by this observation, we propose the ***Decompose-Embed-Interact paradigm***, which reframes factuality verification as a modular, embedding-native process. Given generated content G and reference material R, the process unfolds in three stages:

Decompose: Decompose G and R into atomic fact sets,

$$F_G = \{f_1^G, \ldots, f_{K_G}^G\} = \text{Decompose}(G),$$
$$F_R = \{f_1^R, \ldots, f_{K_R}^R\} = \text{Decompose}(R),$$

where f_i^G and f_j^R denote the i-th and j-th atomic fact extracted from G and R, respectively, and K_G, K_R are the total number of facts from each source.

Embed: Independently encode each atomic fact into a dense semantic embedding,

$$\mathbf{h}_i^G = \text{Embed}(f_i^G), \quad \mathbf{h}_j^R = \text{Embed}(f_j^R),$$

where $\mathbf{h}_i^G \in \mathbb{R}^d$ and $\mathbf{h}_j^R \in \mathbb{R}^d$ represent the d-dimensional embeddings of the i-th generated fact and the j-th reference fact, respectively.

Interact: Predict a binary label indicating factual consistency through operations in the embedding space,

$$\text{y}_{j,i} = \text{Interact}(\mathbf{h}_j^R, \mathbf{h}_i^G),$$

where $\text{y}_{j,i} \in \{0, 1\}$ denotes whether the generated fact f_i^G is supported or not supported by the reference fact f_j^R.

4 E-Verify

While the proposed paradigm provides a conceptual blueprint for efficient verification in embedding space, its practical implementation poses three key challenges: (1) achieving efficient decomposition of long-form text, (2) preserving factual fidelity during embedding, and (3) verifying factual consistency via accurate embedding-level interactions. We present **E-Verify**, addressing these challenges through three carefully designed modules, as illustrated in Figure 2. We provide detailed descriptions of each component below, with implementation settings provided in Appendix A.

4.1 Decomposer: Sentence-Level Atomic Fact Extraction

The use of SLMs to replace LLMs has become a common practice across many NLP tasks to improve efficiency. However, we find that applying SLMs directly to factual decomposition, especially on long-form text, often leads to incomplete extraction or hallucinated facts. To mitigate this, we adopt a sentence-level decomposition strategy that reduces contextual hallucination and improves atomic fact fidelity.

We segment the input text (generated content and reference material) into sentences using Stanza (Qi et al., 2020), denoted as $S = \{s_1, s_2, ..., s_n\}$, where n is the total number of sentences. Each sentence $s_j \in S$ is individually processed by the SLM to extract atomic facts F_j, and these are aggregated into a unified fact set $F = \bigcup_{j=1}^{n} F_j = \{f_1, f_2, \ldots, f_K\}$, where K is the total number of atomic facts.

4.2 Embedder: Context Encoding with Token-Level Attention Pooling

The Embedder encodes atomic facts into dense vector representations to enable efficient factuality verification. Traditional BERT-based sentence embedding methods, such as using the `[CLS]` token

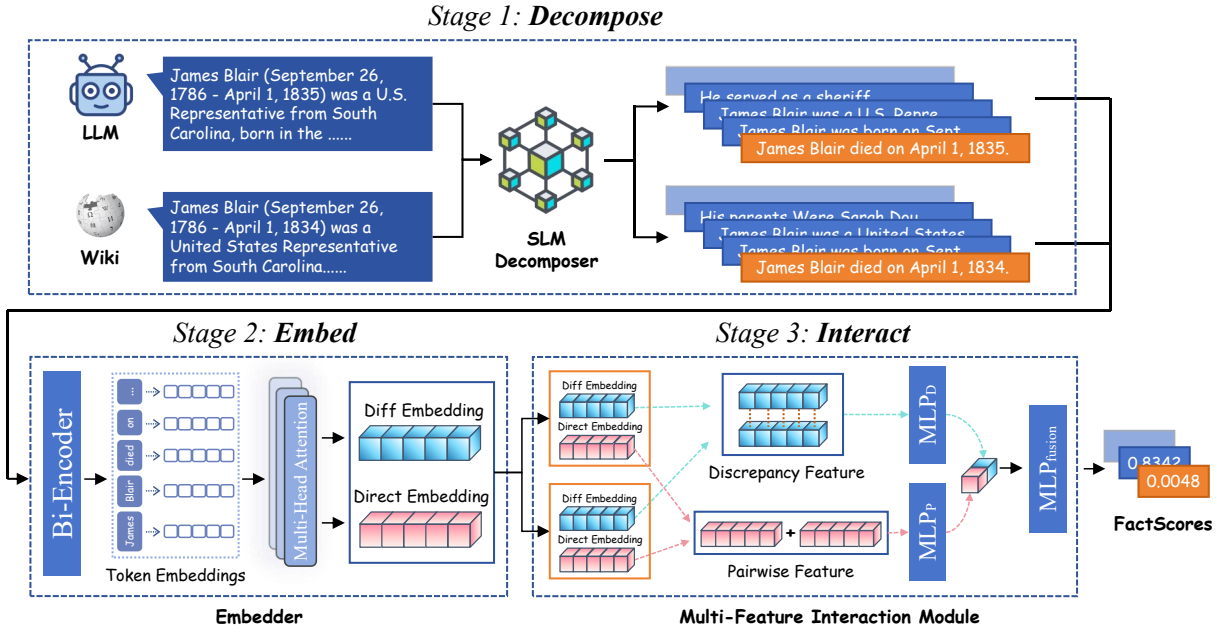

Figure 2: Overview of the **E-Verify** framework for factuality verification. The system follows a three-stage process: **Decompose**, **Embed**, and **Interact**. In the ***Decompose*** *stage*, the LLM-generated text and the corresponding reference text from Wikipedia are processed using a SLM decomposer. In the ***Embed*** *stage*, these atomic facts are encoded using a Bi-Encoder, with the use of PMA to capture different embedding features. In the ***Interact*** *stage*, the embeddings undergo multi-feature interactions through feature-based processing, producing fact scores to assess the factuality of the content.

or mean pooling (Reimers, 2019), often fail to capture fine-grained semantic nuances that are crucial for distinguishing subtle factual differences. To address this, we adopt a Pooling-based Multi-Head Attention (PMA) mechanism (Liao et al., 2024; Lee et al., 2019) built on top of the BERT encoder to enhance factuality-oriented embeddings.

Given an atomic fact set $F = \{f_1, f_2, \ldots, f_K\}$, K is the total number of atomic facts, each fact $f_i \in F$ is tokenized and encoded by BERT into token embeddings $T_i = \{t_1, t_2, \ldots, t_l\}$, where l is the number of tokens in f_i. Each token $t_k \in T_i$ is a d-dimensional vector. The PMA module then aggregates T_i to produce a multi-view sentence embedding:

$$h = \text{LN}(\text{MHA}(q, T_i, T_i) + q),$$
$$H_i = \text{LN}(h + \text{FFN}(h)),$$

where LN denotes Layer Normalization, MHA is Multi-Head Attention, and q is a learnable query vector dynamically aggregating token-level information. We use two learnable queries within PMA to produce multi-view embeddings, denoted as $H_i[0]$ and $H_i[1]$, that preserve richer contextual information. These embeddings are later assigned distinct roles during factuality verification, enabling fine-grained modeling of factual alignment and discrepancy signals.

4.3 MFIM: Embedding-Space Interaction for Factuality Verification

Traditional sentence similarity models often rely on cosine similarity between embeddings. However, cosine similarity is symmetric and fails to capture the directional nature of factual entailment, which is essential for distinguishing support and non-support in factuality verification. To address this, we design the Multi-Feature Interaction Module (MFIM) as a lightweight verifier that produces a scalar fact score directly from embedding representations. This design aligns with our paradigm-level goal of replacing expensive pairwise verification with scalable vector operations.

We observe two major error types in factual consistency: (1) surface-level mismatches (e.g., entity names, numbers, dates), and (2) subtle factual additions or omissions requiring directional reasoning to determine whether generated content is sufficiently supported by the reference. While simple pairwise alignment in embedding space (e.g., concatenation) effectively addresses type (1), it fails to capture the strong directional factual entailment behavior observed in NLI tasks. To close this gap, we draw inspiration from *difference-based signal pro-*

cessing, where subtractive operations emphasize residual discrepancies by eliminating shared components. Accordingly, we explicitly introduce a discrepancy feature to model directional differences between reference and generated embeddings.

Thus, we define two features: the Pairwise Feature P and the Discrepancy Feature D:

$$P = \text{MLP}_P(\text{Concat}(H_r[0], H_g[0])),$$
$$D = \text{MLP}_D(H_r[1] - H_g[1]),$$

where H_r and H_g are multi-view embeddings of the reference and generated atomic fact.

The final fact score is computed by first concatenating these features and then passing the fused vector through a lightweight linear layer with Sigmoid activation:

$$\text{FactScore} = \text{Sigmoid}(\text{Linear}([P; D])) \in (0, 1).$$

Our ablation studies (Section 5.5) further confirm that both features offer complementary signals and are critical to optimal verification performance.

4.4 Computational Complexity Analysis

In this section, we theoretically analyze the computational efficiency of the E-Verify framework. We divide the analysis into two main components: the Decomposer, which is responsible for atomic fact extraction, and the Checker, which handles embedding and interaction.

4.4.1 Decomposer Complexity Analysis

E-Verify utilizes a supervised fine-tuned SLM to perform atomic fact decomposition at the sentence level. The primary computational cost lies in applying the decomposer to each sentence, as sentence segmentation itself is negligible.

Given that the input sequence of T tokens is partitioned into N sentences, language models employing self-attention mechanisms (Vaswani, 2017) incur quadratic computational complexity $O(T^2)$. E-Verify addresses this challenge through sentence-level decomposition. By constraining attention computations to individual sentences with average length $\bar{t} = \frac{T}{N} \ll T$, the overall complexity reduces to $O(N\bar{t}^2)$. This design drastically reduces global attention costs by restricting attention computations to shorter text segments, making E-Verify substantially more efficient than conventional passage-level LLM processing.

4.4.2 Checker Complexity Analysis

The Checker module consists of the Embedder and the MFIM, and its computational complexity is determined by two main components: embedding computation and factuality verification computation. We denote K_g and K_r as the numbers of atomic facts extracted from the generated content and the reference material, respectively.

Embedding Computation. Embedder employs a Bi-Encoder structure, enabling independent encoding of atomic facts before interaction. Assuming the BERT encoder has a computational complexity of $O(B)$ per atomic fact, the total embedding complexity is $O((K_g + K_r)B)$.

Factuality Verification Computation. The MFIM performs lightweight pairwise interactions between atomic fact embeddings in the embedding space. Each generated atomic fact is compared against all reference atomic facts, with verification complexity of $O(K_g K_r M)$, where M denotes the computational complexity of the MLP.

Thus, the overall computational complexity of the Checker module is $O((K_g+K_r)B+K_g K_r M)$. For a standard NLI-based model, each generated atomic fact is compared against reference material using cross-encoding. Assuming the computational complexity per cross-encoding is $O(B)$, the total complexity becomes $O(K_g K_r B)$.

Key Insight. While NLI models incur quadratic complexity at the transformer computation level, E-Verify shifts the costly inferences to lightweight MLP operations. Since MLPs are substantially more efficient than transformer encoders, E-Verify significantly reduces computational overhead.

5 Experiments

To evaluate the effectiveness of the E-Verify framework, we conduct experiments across four key dimensions: **Decomposition Quality**: Compare various models to identify the most effective atomic fact decomposer. **Factuality Verification**: Assess the Checker module and the end-to-end E-Verify framework against strong baselines. **Efficiency**: Analyze runtime and memory efficiency across all stages of the E-Verify pipeline. **Ablation Study**: Examine the contributions of core components such as PMA and MFIM. The detailed experiment settings are provided in Appendix B.

5.1 Datasets

wiki-en-sentences: A sentence-level factuality detection dataset containing pairs of independent Wikipedia sentences.

wiki-bio-hallu (Manakul et al., 2023): A hallucination detection dataset for biography generation, consisting of a generated biography and its corresponding Wikipedia source. The dataset includes a simple subset, which contains controlled factual errors in numbers, time, entities, or events; and a hard subset, in which errors naturally occur in LLM-generated biographies.

CNN (Tang et al., 2024): A fact verification dataset based on CNN news articles. Each instance includes a reference article and a generated summary that may contain factual errors.

Reveal (Tang et al., 2024): A dataset adapted from REVEAL (Jacovi et al., 2024), originally designed for evaluating reasoning chains in open-domain QA, which we use in our setting as (passage, fact) pairs with binary factuality labels.

5.2 Decomposition Capability Evaluation

We evaluate the decomposition performance of GPT-4o with several open-source models, Qwen2-7B (Bai et al., 2023), Qwen2.5-0.5B (Yang et al., 2024) and Flan-T5 (Chung et al., 2022), on the wiki-bio-hallu dataset. The evaluation metrics include **Precision**, **Recall**, and **F1 Score**. **Precision** measures the factual correctness of the extracted facts. Specifically, for each fact output by the decomposer, we check whether it is semantically supported by the original content. **Recall** measures the completeness of the decomposition. For each ground-truth fact, we check whether it is semantically entailed by any of the extracted facts. This reflects how much of the original factual content is successfully recovered. **F1 Score** is computed as the harmonic mean of Precision and Recall.

Model	Granularity	F1	Precision	Recall
GPT-4o*	Passage	0.9910	0.9830	0.9991
Qwen2-7B	Sentence	0.9797	0.9799	0.9795
Qwen2-7B	Passage	0.9703	0.9875	0.9536
Qwen2.5-0.5B	Sentence	0.9676	0.9628	0.9725
Flan-T5	Sentence	0.9486	0.9512	0.9460
Qwen2.5-0.5B	Passage	0.8837	0.8920	0.8754

Table 1: Performance comparison of different decomposers under different decomposition granularities. *GPT-4o was evaluated using few-shot prompting, while other open-source models were supervised fine-tuned using synthetic data generated by GPT-4o.

As shown in Table 1, GPT-4o achieves near-perfect results under few-shot prompting, serving as an upper bound for accuracy. Among fine-tuned open-source models, sentence-level decomposition consistently yields higher recall than passage-level variants, highlighting its advantage in recovering comprehensive factual content. Notably, Qwen2-7B exhibits strong performance but suffers a recall drop on longer inputs, indicating potential limitations in long-context handling. Qwen2.5-0.5B achieves a favorable balance between quality and efficiency at the sentence level, making it the most suitable choice for E-Verify's decomposition module in large-scale scenarios.

5.3 Factuality Verification Performance Assessment

In this section, we validate the factuality verification ability of E-Verify through two experiments. The first experiment focuses on assessing the effectiveness of the Checker. The second experiment evaluates the full E-Verify framework, incorporating both the Decomposer and Checker. A detailed case study is provided in Appendix D.

5.3.1 Experiment on Checker

To evaluate the performance of the Checker module, we compare E-Verify against several non-LLM baselines, including NLI-based and Bi-Encoder models. We conduct experiments on three in-domain datasets (**wiki-en-sentences**, **wiki-bio-hallu (simple)**, and **wiki-bio-hallu (hard)**) and two out-of-domain (OOD) datasets (**CNN** and **Reveal**). E-Verify is trained on a Wikipedia-style dataset, making the former the primary benchmark for in-domain evaluation, while the latter assesses generalization under OOD cases.

As shown in Table 2, E-Verify achieves the strongest performance among all non-LLM baselines on the in-domain datasets, attaining the highest accuracy and Macro-F1, particularly on the simpler factuality sets. On more challenging datasets, such as wiki-bio-hallu (hard) and the OOD cases, E-Verify remains competitive—slightly trailing MiniCheck in overall accuracy but outperforming traditional NLI and Bi-Encoder models in Macro-F1, indicating stronger handling of class imbalance and fine-grained distinctions. Notably, Bi-Encoder models exhibit acceptable accuracy but consistently lower Macro-F1, suggesting difficulty in capturing subtle factual discrepancies. While LLMs such as GPT-4o maintain consistently high performance

Types	Models	Para.	wiki-en-sentences		wiki-bio-hallu (simple)		wiki-bio-hallu (hard)		CNN*		Reveal*	
			Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
	Random	-	0.4956	0.4929	0.5101	0.4997	0.5011	0.5011	0.5107	0.4769	0.5483	0.5197
L	GPT-4o	-	0.9772	0.9768	0.9974	0.9972	0.9480	0.9480	0.9240	0.9010	0.8889	0.8705
	Qwen2-7B	7B	0.9866	0.9862	0.9257	0.9143	0.7974	0.7920	0.8551	0.7871	0.8527	0.8425
	Llama3.2-3B	3.2B	0.9056	0.9039	0.8590	0.8399	0.7006	0.6848	0.7648	0.6765	0.7754	0.7640
	MiniCheck-Flan-T5	750M	0.9518	0.9511	0.8814	0.8618	0.8291	0.8283	0.8599	0.8098	0.8647	0.8519
X	MiniCheck-DeBERTa	434M	0.9160	0.9155	0.7697	0.7678	**0.7289**	**0.7151**	**0.7743**	**0.7435**	**0.8937**	**0.8809**
	roberta-large-snli	355M	0.9208	0.9169	0.6798	0.4821	0.5157	0.3917	0.7221	0.4354	0.5314	0.5305
	DeBERTa-mnli-fever-anli	184M	0.9028	0.8962	0.7787	0.6985	0.5959	0.5375	0.7340	0.4841	0.6280	0.6271
	nli-deberta-v3-base	184M	0.9324	0.9289	0.8199	0.7846	0.6939	0.6877	0.7197	0.4699	0.7705	0.7604
	nli-roberta-base	125M	0.9422	0.9400	0.7967	0.7871	0.6436	0.6436	0.6746	0.5327	0.7657	0.7482
B	BERTScore	355M	0.5776	0.3661	0.6519	0.3946	0.4904	0.3291	0.7173	0.4177	0.3068	0.2348
	BGE-en-base-v1.5	109M	0.6562	0.5422	0.6519	0.3946	0.4934	0.3354	0.7173	0.4177	0.3092	0.2387
	Ours	114M	**0.9706**	**0.9697**	**0.8655**	**0.8480**	0.6631	0.6581	0.7197	0.6945	0.8188	0.8007

Table 2: Performance comparison of various models across different datasets. The table presents **Accuracy**, **Macro-F1** for different models, including random, LLM-based models, Cross-Encoders, Bi-Encoders, and our proposed method. The best results are marked in **bold**, and the next best results are underlined. L stands for LLM, X stands for Cross-Encoder, and B stands for Bi-Encoder. Para. denotes parameters. Datasets marked with * are considered out-of-distribution with respect to our method. Details of baseline models are provided in Appendix B.2.

across all datasets, they incur substantial computational overhead (e.g., GPT-4o consumed 5.03M tokens, costing $18.86 USD), making them less suitable for scalable or cost-sensitive verification scenarios.

5.3.2 Experiment on E-Verify

Decomposer	Checker	Pearson ↑	MAE ↓
GPT-4o	GPT-4o	0.9650	0.0783
	Qwen2-7B	0.9524	0.1040
	MiniCheck-Flan-T5-L	0.9005	0.1548
	MiniCheck-DeBERTa	**0.8100**	0.2132
	DeBERTa-mnli-fever-anli	0.6528	0.3498
	nli-deberta-v3-base	0.7394	0.1692
	BGE-en-base-v1.5	0.1739	0.6220
	Ours	0.7452	0.1792
Ours	Qwen2-7B	0.9171	0.1319
	Ours	0.7386	**0.1646**

Table 3: Performance of different decomposers and checkers on the wiki-bio-hallu (hard) dataset. **Pearson Correlation** and **Mean Absolute Error (MAE)** serve as evaluation metrics. **Bold** indicates the best results, and underlined indicates the next best results.

In this section, we assess the end-to-end reliability of E-Verify in factuality scoring, using the **wiki-bio-hallu (hard)** dataset comprising LLM-generated biographies with human-annotated factuality scores. We evaluate various combinations of decomposers and checkers, and compute alignment with ground-truth using **Pearson Correlation** and **Mean Absolute Error (MAE)**.

As shown in Table 3, LLM-based pipelines (e.g., GPT-4o and Qwen2-7B) unsurprisingly achieve the strongest overall performance, but at substantial computational cost—serving primarily as upper bounds in efficiency-constrained scenarios.

Among non-LLM models, MiniCheck-DeBERTa obtains the highest Pearson score, while our E-Verify checker achieves the lowest MAE across all non-LLM settings, demonstrating higher precision in capturing factual consistency. Importantly, E-Verify maintains stable performance regardless of whether it is paired with a high-resource decomposer (GPT-4o) or its own lightweight decomposer, demonstrating both robustness and modular adaptability. Compared to traditional NLI models and embedding-based baselines (e.g., BGE), E-Verify consistently achieves better correlation and lower error, confirming its stronger sensitivity to subtle factual discrepancies and more reliable factuality assessment—serving as a scalable alternative to NLI-based pipelines.

5.4 Computational Efficiency Analysis

In this section, we analyze the computational efficiency of E-Verify on the wiki-bio-hallu (hard) dataset. To ensure fair comparison, we evaluate decomposition and verification latency separately. **Decomposition time** (Figure 3, left) is measured as the total time each decomposer takes to process the same set of documents. **Verification time** (Figure 3, right) is measured using the same set of

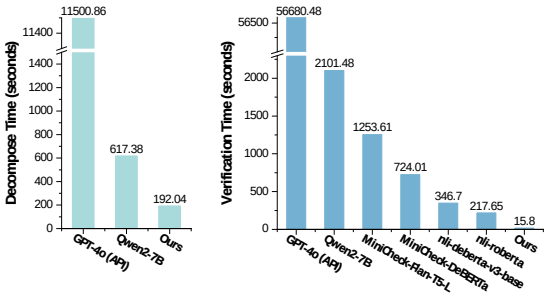

Figure 3: Computational efficiency comparison. The left plot shows the total decomposition time, while the right plot presents the total factuality verification time. All times reflect GPU wall-clock inference time, except GPT-4o which reflects external API latency. Our method achieves the lowest computation time in both stages.

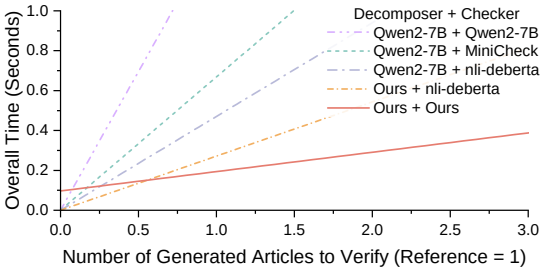

Figure 4: Overall time as the number of generated documents to verify increases, assuming a fixed reference document. E-Verify yields the lowest overall cost after amortizing the initial reference encoding cost, even prior to completing the first verification.

decomposed atomic facts (produced by GPT-4o) across all baseline checkers. For all models, we exclude decomposition time from this stage to isolate checker efficiency.

Efficient Atomic Fact Extraction in Sentence-Level. As shown in Figure 3 (left), E-Verify achieves a significant 60× speedup over GPT-4o API calls and a 3.21× speedup over Qwen2-7B in total decomposition time. This gain is not solely attributable to model downsizing, but of the finer-grained decomposition strategy. By performing sentence-level atomic decomposition with a fine-tuned SLM, E-Verify avoids the need for global attention over long-form text and enables parallel, lightweight processing of individual sentences.

Lightweight Verification in Embedding Space. As shown in Figure 3 (right), E-Verify completes factuality verification in just 15.8 seconds—a 22× speedup over nli-deberta and 46× over MiniCheck-DeBERTa. Unlike previous methods that rely on cross-encoding every reference-fact pair, our framework performs lightweight, embedding-space inference with fixed-size vector inputs and fully reusable reference representations. Notably, 98.4% of the verification time stems from embedding computation, which is amenable to precomputation and caching. The actual interaction takes only 0.25 seconds, illustrating how our decoupled design transforms factuality verification from a high-cost inference into an efficient embedding interaction.

Scalability Advantage under Real-World Verification Scenarios. We further evaluate E-Verify under a realistic verification workload, where a static reference document is used to verify a growing number of generated documents. This setup reflects real-world scenarios where the reference source is typically static and trusted, while LLM-generated content varies dynamically. As shown in Figure 4, our method exhibits the lowest growth rate in total computation time, growing only 20% as fast as `Qwen2-7B+nli-deberta`. Although our method incurs a small initial cost from reference processing, this cost is quickly amortized; E-Verify becomes the most efficient system even before completing the first document and maintains this advantage as the number of verifications grows. These results demonstrate that E-Verify is practically efficient and deployable in time-sensitive applications. A detailed cost breakdown is provided in Appendix C.

	wiki-en-sentences		wiki-bio-hallu (simple)	
	Acc	Macro-F1	Acc	Macro-F1
E-Verify	**0.9706**	**0.9697**	**0.8655**	**0.8480**
-PMA+Pool	0.9058	0.9004	0.7783	0.7149
-MFIM+Cosine	0.8190	0.8119	0.7482	0.6503
-PMA-MFIM	0.6562	0.5422	0.6519	0.3946
MFIM(only P)	0.9520	0.9504	0.8492	0.8309
MFIM(only D)	0.9546	0.9530	0.8642	0.8463

Table 4: Ablation study results comparing different configurations for factuality verification across two datasets.

5.5 Ablation Studies

We conduct ablation studies on the Checker module to evaluate the effect of the Pooling-based Multi-Head Attention (PMA) and the Multi-Feature Interaction Module (MFIM).

As shown in Table 4, replacing PMA with global pooling methods results in a significant drop in accuracy and Macro-F1, indicating the critical role of attention-based token aggregation in preserving fine-grained semantic information. Replacing

the MFIM with cosine similarity causes a notable decline in performance, particularly in Macro-F1, which reflects degraded ability to handle nuanced factual inconsistencies. This suggests that simple similarity metrics are insufficient for modeling entailment-style relations. Eliminating both PMA and MFIM yields the weakest overall performance, confirming that their combination is essential for robust factuality verification. We further evaluate the impact of the MFIM's internal features: the pairwise feature P and the discrepancy feature D. While D alone performs closest to the full model, the best results are achieved when both P and D are used together, highlighting their complementary roles in factuality verification. This underscores the importance of explicit discrepancy modeling in capturing subtle fact-level mismatches that may be missed by direct embedding alignment alone.

6 Conclusion

We propose E-Verify, a lightweight framework that redefines factuality verification through a novel *Decompose-Embed-Interact* paradigm. By decoupling decomposition, embedding, and interaction, E-Verify replaces costly cross-encoding with efficient embedding-space alignment. Experiments show that E-Verify significantly improves computational efficiency while maintaining competitive accuracy. These results validate the paradigm's practical value and highlight the potential of embedding-native verification as a scalable solution for real-world factuality verification tasks.

Limitations

Despite the strong empirical performance of the E-Verify framework on factuality verification tasks, several limitations remain:

Inference Limitation: E-Verify employs a Bi-Encoder-based design that prioritizes efficiency by independently encoding the generated content and reference materials. While this architecture greatly accelerates verification, it inevitably introduces semantic compression, where subtle factual nuances may be lost during fixed-length embedding. E-Verify may struggle with complex reasoning tasks such as causal inference, temporal reasoning, or conditional relationships, where capturing rich token-level interactions is critical. Such deep reasoning capabilities are better modeled by Cross-Encoder architectures, which allow joint representation learning.

Granularity Limitation: E-Verify verifies factual consistency at the atomic fact level by decomposing text into discrete factual units. While atomic-level verification ensures interpretability, it inherently abstracts away broader discourse dependencies. These include implicit relationships among multiple facts, or factual consistency that depends on paragraph-level context. Handling such inter-fact dependencies or hierarchical factual structures remains an open challenge for future work.

Acknowledgments

This work was supported by the National Science and Technology Major Project under Grant 2022ZD0120204.

References

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv:2303.09540*.

Pepa Atanasova. 2024. Generating fact checking explanations. *Accountable and Explainable Methods for Complex Reasoning over Text*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv:2309.16609*.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv:2302.04023*.

I Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. 2023. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv:2307.13528*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz,

and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv:abs/2309.11495*.

Jiarun Fu, Lizhong Ding, Hao Li, Pengqi Li, Qiuning Wei, and Xu Chen. 2025. Unveiling and causalizing cot: A causal pespective. *arXiv preprint arXiv:2502.18239*.

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

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. 2024. Does fine-tuning llms on new knowledge encourage hallucinations? *arXiv:2405.05904*.

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:2311.05232*.

Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. 2022. Knowledge distillation from a stronger teacher. *Neurips*, 35:33716–33727.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv:2410.21276*.

Hasan Iqbal, Yuxia Wang, Minghan Wang, Georgi Georgiev, Jiahui Geng, Iryna Gurevych, and Preslav Nakov. 2024. Openfactcheck: A unified framework for factuality evaluation of llms. *arXiv:2408.11832*.

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva Pipek. 2024. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. In *ACL*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Neurips*, 33.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. Pre-trained language models for text generation: A survey. *ACM Computing Surveys*, 56.

Zihan Liao, Hang Yu, Jianguo Li, Jun Wang, and Wei Zhang. 2024. D2llm: Decomposed and distilled large language models for semantic search. *arXiv:2406.17262*.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv:2303.08896*.

Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. 2020. Language models are few-shot learners. *arXiv:2005.14165*, 1.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv:2005.00661*.

Daniel McDonald, Rachael Papadopoulos, and Leslie Benningfield. 2024. Reducing llm hallucination using knowledge distillation: A case study with mistral large and mmlu benchmark. *Authorea Preprints*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv:2305.14251*.

Jihwan Oh, Jeonghwan Choi, Nicole Hee-Yoen Kim, Taewon Yun, and Hwanjun Song. 2025. Learning to verify summary facts with fine-grained llm feedback. In *Proceedings of the 31st International Conference on Computational Linguistics*.

Rrubaa Panchendrarajan and Arkaitz Zubiaga. 2024. Claim detection for automated fact-checking: A survey on monolingual, multilingual and cross-lingual research. *Natural Language Processing Journal*, 7.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv:2003.07082*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *ACL*, 11.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv:1908.10084*.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv:2303.11156*.

Jiasheng Si, Yibo Zhao, Yingjie Zhu, Haiyang Zhu, Wenpeng Lu, and Deyu Zhou. 2024. Checkwhy: Causal fact verification via argument structure. *arXiv:2408.10918*.

Joe Stacey, Pasquale Minervini, Haim Dubossarsky, Oana-Maria Camburu, and Marek Rei. 2024. Atomic inference for nli with generated facts as atoms. In *EMNLP*.

Liyan Tang, Philippe Laban, and Greg Durrett. 2024. Minicheck: Efficient fact-checking of llms on grounding documents. In *EMNLP*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*.

A Vaswani. 2017. Attention is all you need. *Neurips*.

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:2412.15115*.

Shiyue Zhang and Mohit Bansal. 2021. Finding a balanced degree of automation for summary evaluation. *arXiv:2109.11503*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv:1904.09675*.

Yiran Zhao, Jinghan Zhang, I Chern, Siyang Gao, Pengfei Liu, Junxian He, et al. 2023. Felm: Benchmarking factuality evaluation of large language models. *Neurips*.

Appendix

A Framework Implementation Details

A.1 Decomposer Training Settings

The decomposer training data is sourced from michaelauli/wiki_bio[3], from which 5,000 samples are randomly selected as the foundational dataset. The Wiki paragraphs are first split into sentences using Stanza, and each sentence is then decomposed by GPT-4o in a few-shot prompting setting (see Figure 6) to generate the training set. To ensure quality, all outputs are further filtered by a local model for factual consistency.

Each training instance follows an instruction format. Below is an example:

> **Input:** Decompose the following sentence into atomic facts: *Carl Demonte Crawford, born on August 5, 1981, and nicknamed "The Perfect Storm," is an American professional baseball left fielder who plays for the Los Angeles Dodgers in Major League Baseball (MLB).*
>
> **Output:** *Carl Demonte Crawford was born on August 5, 1981. Carl Demonte Crawford is nicknamed "The Perfect Storm." Carl Demonte Crawford is an American professional baseball left fielder. Carl Demonte Crawford plays for the Los Angeles Dodgers in Major League Baseball (MLB).*

Our decomposer is based on Qwen/Qwen2.5-0.5B-Instruct[4]. The base model undergoes supervised fine-tuning on all model parameters using the llama-factory framework[5], with a learning rate of 2.0e-5, batch size of 4, and trained for 3 epochs using the AdamW optimizer. We use standard token-level cross-entropy loss.

A.2 Checker Training Settings

The checker training dataset is wiki-en-sentences (see Appendix B.1). We first collect natural sentences from Wikipedia and use GPT-4o to synthesize positive and negative factual variants via prompts(see Figure 7, 8). Specifically, for each anchor sentence, GPT-4o generates: (1) a semantically equivalent sentence (positive) and (2) a minimally edited variant containing factual errors (negative), including entity, date, or numerical hallucinations. To ensure quality, all outputs are further filtered by a local model.

Each training instance follows a triplet format. Below is an example:

> **Anchor:** *This species, Anasimyia contracta, is a European inhabitant within the hoverfly family.*
>
> **Positive:** *Anasimyia contracta is a European species of hoverfly.*
>
> **Negative:** *Anasimyia contracta is an African species of hoverfly.*

We conduct end-to-end joint training of the Embedder and Multi-Feature Interaction Module (MFIM). The Embedder is responsible for generating high-quality sentence embeddings using the BERT model bge-base-en-v1.5[6]. The MFIM then computes fact scores based on these embeddings.

We employ two loss functions: Triplet Loss and Binary Cross-Entropy Loss. The objective of Triplet Loss is to optimize fact scores through supervised learning of triplets, ensuring that the factual score of the anchor sentence is higher when paired with a highly factual positive sentence while being lower when paired with a negative sample.

$$\mathcal{L}_{triplet} = \max(0, \alpha + \text{FactScore}(H_{anc}, H_{neg}) - \text{FactScore}(H_{anc}, H_{pos}))$$

where α denotes the margin, set to 0.5. H_{anc}, H_{pos}, and H_{neg} represent the embeddings of the anchor, positive, and negative samples, respectively, with fact scores computed via the MFIM.

Simultaneously, BCE Loss is employed for supervised training. The FactScore output by the MFIM is a value in the range $(0, 1)$, indicating the degree of alignment between the generated content G and the reference content R. The objective is to minimize the difference between the predicted score and the ground-truth label $y \in \{0, 1\}$:

$$\mathcal{L}_{\text{bce}} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(\text{FactScore}_i) + (1 - y_i) \cdot \log(1 - \text{FactScore}_i)]$$

[3] https://huggingface.co/datasets/michaelauli/wiki_bio

[4] https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct

[5] https://github.com/hiyouga/LLaMA-Factory

[6] https://huggingface.co/BAAI/bge-base-en-v1.5

The overall joint training objective function is formulated as the sum of Triplet Loss and BCE Loss:

$$\mathcal{L} = \mathcal{L}_{\text{triplet}} + \mathcal{L}_{\text{bce}}$$

The training process is conducted in two phases. In the first phase, the parameters of the BERT model are frozen, and only the PMA module within the Embedder and the MFIM are updated. During this phase, the learning rate is set to 5e-5, the batch size is 32, and the model is trained for 8 epochs. In the second phase, we unfreeze the BERT model and apply LoRA to train the final two layers of BERT jointly with the PMA and MFIM modules. The learning rate remains at 5e-5, and the batch size stays at 32. This phase further fine-tunes the model to improve performance.

B Experiment Settings

Our experiments are conducted on a system running Ubuntu 22.04, equipped with an NVIDIA RTX 4090 GPU, an AMD Ryzen 9 9950X CPU, 128GB RAM, and software dependencies, including CUDA 12.4, pytorch 2.4.1, transformers 4.49.0 and vllm 0.6.6.post1.

B.1 Datasets

wiki-en-sentences: A large-scale factuality detection dataset constructed from 500,000 Wikipedia sentences selected from wikipedia-en-sentences[7]. We employ Qwen2-7B to generate both positive and negative samples via prompting. The final training set consists of 2,749,030 triplets, with 50,000 sentence pairs used for validation and 5,000 for testing. We train our E-Verify model on this dataset.

wiki-bio-hallu (Manakul et al., 2023): A dataset for evaluating hallucinations in LLM-generated biographies, containing 238 Wikipedia biography articles. We expanded this dataset with both simple and hard subset to enhance its applicability in factuality verification. The simple subset consists of controlled factual hallucinations generated by GPT-4o (Hurst et al., 2024), with errors primarily focused on four categories: numbers, time, entities, or events. These controlled errors allow for targeted testing of factuality verification models. In contrast, the hard subset is sourced from real-world LLM-generated biographies, which are more diverse and naturally prone to factual inconsistencies. This dataset includes biographies produced by a mix of closed-source models, such as GPT-3.5-Turbo, GPT-4o, Claude-3.5-Haiku, and Claude-3.5-Sonnet, as well as open-source models, including Llama-2-7b, Llama-2-13b (Touvron et al., 2023), Qwen2-7B (Bai et al., 2023), and Qwen2.5-0.5B (Yang et al., 2024). The inclusion of diverse sources in the hard subset makes it more challenging and reflective of real-world factual discrepancies, compared to the simple subset. Each biography is also decomposed into atomic facts using GPT-4o, with each fact being manually labeled for factual accuracy based on the corresponding Wikipedia biography of the individual.

CNN (Tang et al., 2024): A fact verification dataset based on CNN news articles. It consists of 116 CNN news articles, each paired with a corresponding summary that may contain factual errors. We prompt GPT-4o to perform atomic fact decomposition on them, breaking down each fact into smaller, verifiable facts. Each atomic fact is then manually annotated to determine its factual accuracy.

Reveal(Tang et al., 2024): A dataset adapted from REVEAL (Jacovi et al., 2024), originally designed for evaluating reasoning chains in open-domain QA, and used in our setting as (passage, fact) pairs with binary factuality labels. The dataset consists of 300 pairs of passages and corresponding facts. We decompose each fact and passage into atomic facts using GPT-4o, and each atomic fact is manually labeled for factual accuracy.

B.2 Baseline Models

GPT-4o: A proprietary instruction-tuned large language model developed by OpenAI, designed for general-purpose reasoning, generation, and factuality-sensitive tasks. It is accessed via the OpenAI API.

Qwen2-7B[8]: A 7B-parameter open-source LLM developed by Alibaba's Qwen team. It is instruction-tuned for general-purpose alignment.

Llama3.2-3B[9]: A 3B-parameter open-source LLM released by Meta. It is instruction-tuned for general-purpose alignment.

MiniCheck Series (Tang et al., 2024): A family of factuality checkers fine-tuned on hallucination-annotated datasets for long-form generation. It

[7] https://huggingface.co/datasets/sentence-transformers/wikipedia-en-sentences

[8] https://huggingface.co/Qwen/Qwen2-7B-Instruct

[9] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

supports both DeBERTa and Flan-T5 backbones, where the former provides a parameter-efficient cross-encoder baseline, and the latter leverages instruction-tuned LLMs for stronger performance. We include both variants in our experiments to compare task-specific cross-encoders and LLM-based factuality verifiers.

DeBERTa-mnli-fever-anli[10]: A cross-encoder model fine-tuned from Microsoft's DeBERTa-v3-base on multiple datasets including MNLI, FEVER, and ANLI. It is optimized for NLI and fact verification tasks, serving as a strong baseline for sentence-level factuality checking.

nli-deberta-v3-base[11]: A cross-encoder model fine-tuned for NLI using datasets such as MNLI, SNLI, and ANLI. It is used as a strong NLI-based factuality checker baseline in our experiments.

nli-roberta-base[12]: A RoBERTa-base cross-encoder model fine-tuned on NLI datasets including SNLI and MNLI. It provides a compact yet strong baseline for textual entailment tasks and is widely used for sentence-pair reasoning evaluations.

roberta-large-snli[13]: A RoBERTa-large model jointly fine-tuned on multiple NLI datasets including SNLI, MNLI, FEVER, and ANLI, incorporating multi-round R1-R3 training. It serves as a robust high-capacity cross-encoder for general-purpose NLI-based factuality assessments.

BGE-en-base-v1.5[14]: A Bi-Encoder embedding model from BAAI, pre-trained for text retrieval and sentence similarity tasks. We use it as a semantic similarity baseline to compute vector-based scores between facts and references.

BERTScore(Zhang et al., 2019): A reference-based evaluation metric that computes token-level semantic similarity between candidate and reference texts using contextual embeddings from BERT. It is widely used in generation evaluation. In our setting, it serves as a lightweight, embedding-based factuality checker baseline.

[10] https://huggingface.co/MoritzLaurer/DeBERTa-v3-base-mnli-fever-anli

[11] https://huggingface.co/cross-encoder/nli-deberta-v3-base

[12] https://huggingface.co/cross-encoder/nli-roberta-base

[13] https://huggingface.co/ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli

[14] https://huggingface.co/BAAI/bge-base-en-v1.5

B.3 Decomposition Capability Evaluation Settings

For ground-truth decomposition, we use GPT-4o to produce atomic facts, which are then manually verified for factual accuracy. We further use Qwen2-7B to assist in factuality judgment via prompted entailment classification. GPT-4o's decomposition is done via few-shot prompting through API calls, while other open-source models are fine-tuned using supervised fine-tuning (SFT).

Both sentence-level and passage-level training datasets are generated by prompting GPT-4o. The sentence-level training set contains 10,986 instances, while the passage-level training set contains 986 instances.

Due to context length limitations (512 tokens), Flan-T5 is only trained for sentence-level decomposition tasks. Qwen2.5-0.5B and Flan-T5 are fine-tuned with full parameters, while Qwen2-7B is fine-tuned using LoRA. The training uses a learning rate of 2.0e-5 and runs for 3 epochs by llama-factory.

B.4 Factuality Verification Performance Assessment Settings

For long-context factuality verification, LLM-based approaches are provided with the entire reference content as the premise input, using a unified prompt to assess factual support (see Figure 9). Cross-Encoder models process the reference in overlapping chunks of 500 characters with a 100-character stride to handle length constraints. In contrast, Bi-Encoder-based methods, including ours, use atomic facts extracted from the reference as premise inputs. In all setups, the hypothesis input consists of atomic facts extracted from the generated content, ensuring a consistent and fact-level comparison across all model types.

B.5 Computational Efficiency Analysis Settings

For the decomposition efficiency experiments, GPT-4o is accessed via API calls, while both Qwen2-7B and our decomposer model are executed using vLLM (Kwon et al., 2023) for inference acceleration, configured with a GPU memory utilization ratio of 0.9.

In the checker efficiency experiments, GPT-4o is also evaluated through API access. Qwen2-7B is accelerated using vLLM, while the other baselines, MiniCheck-DeBERTa, nli-deberta, and our checker, are run using the transformers and pytorch for inference.

C Efficiency Analysis

We begin by analyzing the decomposition and verification time per document or document pair based on experimental measurements from the wiki-bio-hallu (hard) dataset. The decomposition time per document is 0.2882 seconds for Qwen2-7B and 0.0897 seconds for Qwen2.5-0.5B (Ours). For our method, embedding takes 0.0073 seconds per document, and interaction requires only 0.00013 seconds per document pair. In contrast, the verification time per document pair is 1.1037 seconds for Qwen2-7B, 0.3803 seconds for MiniCheck-DeBERTa, and 0.1821 seconds for nli-deberta.

Assuming a fixed reference document and x newly generated documents, the overall verification time for different `Decomposer+Checker` combinations is provided in Table 5. For our framework that requires decomposing and embedding the reference content, the total time includes a one-time cost associated with the single reference document, reflected as an additional $+1$ term in the formulas.

Compared to `Qwen2-7B+NLI`, our method `Ours+Ours` achieves a $4.84\times$ reduction in per-document verification cost. Even when paired with a standard NLI verifier (`Ours+NLI`), our decomposition and embedding pipeline still provides a $2.80\times$ cost reduction.

It is important to note that our method introduces a small initialization cost due to the need to decompose and embed the reference material. However, in most real-world factuality verification scenarios, the reference corpus is typically static, consisting of a fixed collection of trusted sources such as Wikipedia documents, curated news reports, scientific papers, or legal documents. These reference materials are stable and do not change with each generation request. As a result, both the decomposition outputs and embeddings for the reference content can be precomputed and cached offline, significantly reducing the online computational cost to only processing the newly generated content.

Furthermore, under the wiki-bio-hallu dataset setup, the factuality verification task involves fixed one-to-one document comparisons. In NLI-based verification methods, the generated content is decomposed into atomic facts, while the reference content is segmented into overlapping chunks (approximately 500 characters each with a 100-character overlap). Each atomic fact is then individually matched against all reference chunks to assess factual consistency. On average, each generated biography contains 26 atomic facts, and each reference biography consists of around 4 chunks, resulting in approximately 108 fact-chunk pairs per document pair. This controlled setting maintains a moderate and fixed number of reference-fact pairs, and thus does not fully expose the quadratic complexity growth typically associated with NLI-based verification under large-scale or dense-generation scenarios.

Nevertheless, even in this relatively mild verification setting, our lightweight decoupled architecture demonstrates substantial computational advantages, achieving significant efficiency gains over traditional NLI-based baselines. This highlights the scalability and robustness of E-Verify, suggesting even greater benefits when applied to larger, more complex fact-checking tasks where traditional methods would suffer from severe pairwise verification explosion.

D Case Study: Fact-to-Fact Alignment and Interpretability

To further illustrate the interpretability of E-Verify, we present a case study using an example from the wiki-bio-hallu dataset. Figure 5 shows the atomic facts extracted from the generated content (left) and the reference content (right). Each line represents the highest-scoring fact-to-fact alignment between a generated fact and a reference fact, with the predicted FactScore shown alongside. We use color-coding to visualize the verification outcomes:

- **Green lines** indicate correctly verified facts with high FactScore values (e.g., *Bill Quinn was born on May 6, 1912.* with a score of 0.9889).
- **Red lines** connect hallucinated or factually incorrect statements to unrelated reference facts with FactScore values close to 0 (e.g., *Bill Quinn concluded his career on 'All in the Family' in 1990* matched against *Bill Quinn's last acting role was in 1989 in 'All in the Family'*, score = 4.30×10^{-7}).
- **Yellow lines** highlight an incorrect high-confidence prediction (false positive). In this example, *Bill Quinn appeared in 'Star Trek.'* was mistakenly linked to *Bill Quinn was an American actor* with a relatively high score of 0.5278, despite lacking supporting evidence.

The yellow case (*Bill Quinn appeared in 'Star Trek.'* matched to *Bill Quinn was an American ac-*

Decomposer+Check	**Decompose Time (s)**	**Verification Time (s)**	Total Time (s)
Qwen2-7B + Qwen2-7B	$0.2882x$	$1.1037x$	$1.3919x$
Qwen2-7B + MiniCheck-DeBERTa	$0.2882x$	$0.3803x$	$0.6685x$
Qwen2-7B + nli-deberta	$0.2882x$	$0.1821x$	$0.4703x$
Ours + nli-deberta	$0.0897x$	$0.1821x$	$0.2718x$
Ours + Ours	$0.0897(x+1)$	$0.0073(x+1)+0.00013x$	$0.0970x+0.0969$

Table 5: Verification time formulas for different Decomposer + Checker combinations under a fixed reference setting. Here, x denotes the number of newly generated documents to be verified, and the reference document is fixed to a single document. The table decomposes the total time into decomposition and verification components for each method.

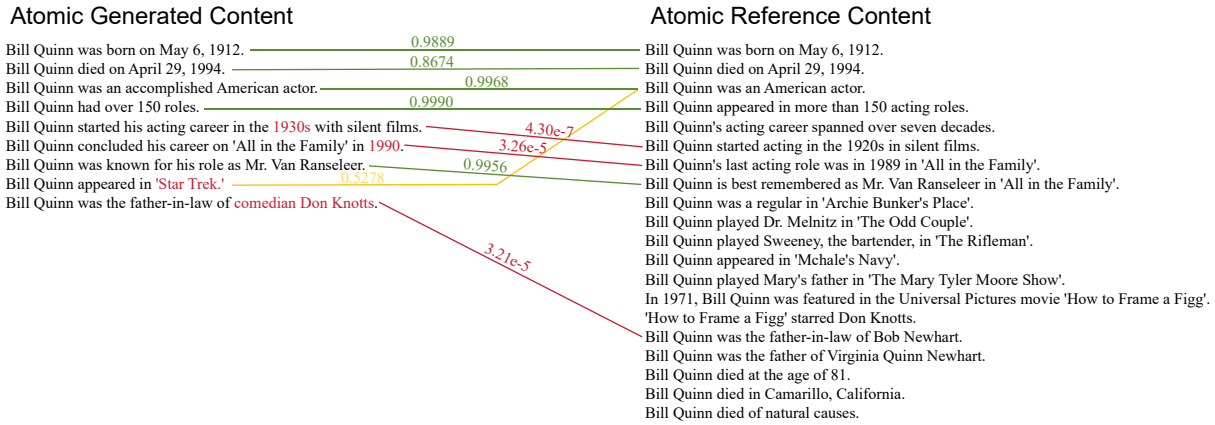

Figure 5: Fact-to-fact alignment case study. Green = correct matches; Red = unsupported/hallucinated facts; Yellow = false positive error.

tor, score = 0.5278) illustrates a known challenge in embedding-based verification systems. We attribute this misalignment to multiple factors: (1) **representation bias**, where pre-trained embedding models tend to map semantically or contextually related entities (e.g., *Star Trek* and *actor*) to nearby regions in the embedding space, even when they are factually unrelated; (2) **insufficient hard negative examples** in the training data, limiting the model's ability to disambiguate rare or long-tail facts. This case highlights a potential limitation of our current framework and points to promising future research directions such as hard negative mining.

Overall, this example demonstrates that our framework provides an interpretable and structured reasoning path by explicitly aligning generated atomic facts to reference facts. The low scores assigned to unsupported or incorrect facts showcase the system's ability to filter factual inconsistencies, while the incorrect prediction offers insight into current limitations and highlights potential directions for future improvement.

Figure 6: Few-shot prompt used for sentence-level atomic fact decomposition.

Sentence-level Atomic Fact Decomposition Prompt

Decompose the following sentences into atom facts if possible, response only the decomposition.
Rely solely on the provided text.
Do not infer or assume additional information.
Do not include any additional information.
Just be faithful to the text.
Examples:

Input: Elisha Brown (25 May 1717 - 20 April 1802) served as Deputy Governor of Rhode Island from 1770 to 1772.
Output: Elisha Brown was born on 25 May 1717. Elisha Brown died on 20 April 1802. Elisha Brown served as Deputy Governor of Rhode Island from 1770 to 1772.

Input: George Bovell is currently a professional swimmer and intends to compete in a record fifth Olympiad. Bovell is also respected for his voluntary giving back initiatives such as "The World Swim Against Malaria and Drowning" in Uganda, 2013, with his friend, Ugandan swimmer Max Kanyarezi.
Output: George Bovell is currently a professional swimmer. George Bovell intends to compete in a record fifth Olympiad. George Bovell is respected for his voluntary giving back initiatives such as "The World Swim Against Malaria and Drowning" in Uganda, 2013. George Bovell did this with his friend, Ugandan swimmer Max Kanyarezi.

Input: He now hosts the breakfast slot on 98FM.
Output: He now hosts the breakfast slot on 98FM.

Now expand this biographical statement with the same accuracy and style, ensuring the facts remain unchanged and no additional information is inferred.

Input: `{sentence}`
Output:

Figure 7: Zero-shot prompt used for non-factual sentence.

Non-factual Sentence Generation Prompt

type: type_info
type_dict = {
    "time": "Time content: Covers time, dates, periods, etc., related to when events occur.",
    "number": "Number content: Includes data, ratios, percentages, etc.",
    "entity": "Entity content: Involves specific entities such as names of people, places, organizations, etc.",
    "event": "Event content: Describes specific events, activities, actions, etc.",
}

Modify the input sentence by changing only the `{type}` content to make the sentence factually incorrect.
`{type_info}`
Ensure that the sentence structure and meaning remain consistent, but the facts related to the `{type}` content should be altered.
Just replace the original `{type}` content with a different, incorrect value.
And provide only the modified sentence as a response.
The sentence is: `{sentence}`
Your answer:

Figure 8: Zero-shot prompt used for positive sentence.

Similar Sentence Generation Prompt

Please take the following sentence and rewrite it using various of expressions, but keep the factual information the same.
Do not add any additional information that is not already mentioned in the original sentence.
And provide only the modified sentence as a response.
The sentence is: `{sentence}`
Your answer:

Figure 9: Prompt used for factuality verification.

Factuality Verification Prompt

Giving a fact and a paragraph, determine if the fact is supported by the paragraph:

Paragraph: `{paragraph}`

fact: `{fact}`

Answer (just yes or no):