Enhancing LLM-based Hatred and Toxicity Detection with Meta-Toxic Knowledge Graph

Yibo Zhao, Jiapeng Zhu, Can Xu, Yao Liu[1*], **Xiang Li** [1*]
East China Normal University, Shanghai, China

Abstract

The rapid growth of social media platforms has raised significant concerns regarding online content toxicity. When Large Language Models (LLMs) are used for toxicity detection, two key challenges emerge: 1) the absence of domain-specific toxicity knowledge leads to false negatives; 2) the excessive sensitivity of LLMs to toxic speech results in false positives, limiting freedom of speech. To address these issues, we propose a novel method called *MetaTox*, leveraging graph search on a meta-toxic knowledge graph to enhance hatred and toxicity detection. First, we construct a comprehensive meta-toxic knowledge graph by utilizing LLMs to extract toxic information through a three-step pipeline. Second, we query the graph via retrieval and ranking processes to supplement accurate, relevant toxicity knowledge. Extensive experiments and case studies across multiple datasets demonstrate that our *MetaTox* boosts overall toxicity detection performance, particularly in out-of-domain settings. In addition, under in-domain scenarios, we surprisingly find that small language models are more competent. Our code is available at https://github.com/YiboZhao624/MetaTox.

Disclaimer: *This paper describes toxic and discriminatory content that may be disturbing to some readers.*

1 Introduction

Online social media platforms have become a major source of information for people worldwide. Meanwhile, they also provide a communication tool for spreading toxic content including harassment, trolling, cyberbullying, and hate speech, which poses a serious and continual threat to the harmony of society (Simpson, 2013) and harms children's mental health (Simpson, 2019). It thus becomes a critical task to detect the toxic content , while cautiously avoiding hindering freedom of expression.

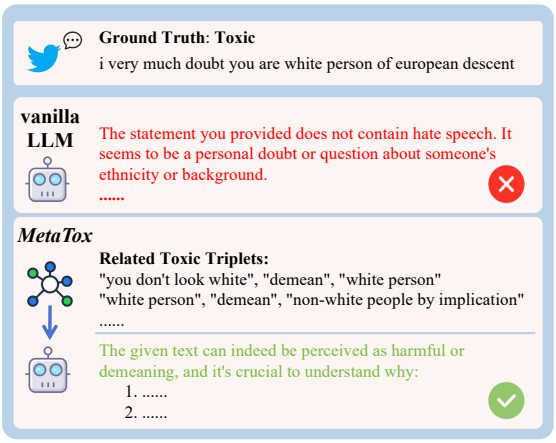

Figure 1: A showcase of our method *MetaTox*, which enhances the LLM to detect toxicity by injecting speech-related toxic knowledge in form of triplets.

To address the problem, previous work can be mainly categorized into three types: (1) rule-based methods (Chen et al., 2012; Gitari et al., 2015) relying on the pre-defined rules; (2) embedding-based methods (Warner and Hirschberg, 2012; Djuric et al., 2015; Park and Fung, 2017) leveraging text representations for classification; (3) transformer-based methods (D'Sa et al., 2020; Luu and Nguyen, 2021) employing transformer architectures with small-scale parameters. The first two types of approaches struggle with semantic understanding, limiting their ability to detect implicit toxicity. While methods of the third type can capture more complex semantics, their reliance on domain-specific training data results in significant performance degradation on out-of-domain data. However, due to their lightweight nature and ease of training, they are widely applied.

The rapid development of large language models (LLMs) has provided new insights into the detection of toxic contents. Therefore, we try to explore whether LLM-based methods can outperform small

*Corresponding author: liuyao@cc.ecnu.edu.cn, xiangli@dase.ecnu.edu.cn

Findings of the Association for Computational Linguistics: ACL 2025, pages 24747–24760

models in this domain. Some studies (Yang et al., 2023; Zhang et al., 2024a) leverage LLMs as data augmentation tools to enhance toxicity detection. However, due to LLMs' lack of domain-specific knowledge in hatred and toxicity, the performance of these methods is still limited. Further, recent research (Zhang et al., 2024b) highlights another issue: LLMs are extremely sensitive to groups or topics that may raise fairness concerns, such as race, gender, and religion, often leading to false positive misjudgment, which impairs the freedom of speech. For example, the benign post "*forget this white nationalist mess. i'm america first...*" is misjudged as toxic by LLM possibly because the model is too sensitive to the "*white nationalist*".

To supplement external domain knowledge for LLMs, Retrieval Augmented Generation (RAG) (Lewis et al., 2020) has been widely adopted. However, naive RAG-based methods heavily rely on the semantic similarity between the embeddings of queries and documents. When query contents are implicitly toxic with vague semantics, retrieved documents could be irrelevant and impair the model performance. Inspired by the recent success of GraphRAG (Edge et al., 2024), as well as the inherent nature of toxic content targeting specific groups, regions, and other entities, we organize external knowledge into a meta-toxic knowledge graph, and dig out toxicity knowledge highly relevant to speech via graph retrieval. The retrieved triplets or paths, which represent concentrated toxicity knowledge derived from the originally fragmented toxic corpus, capture rich meta-toxic information. This process facilitates more accurate detection of toxicity, thereby alleviating the issue of false positive misjudgments.

In this paper, we propose *MetaTox*, which utilizes a meta-toxic knowledge graph and LLMs for hate and toxic speech detection. We first construct a meta-toxic knowledge graph from existing toxic benchmark datasets by designing a three-step pipeline: *rationale reasoning*, *triplet extraction*, and *entity resolution*. Specifically, rationale reasoning aims to reason about what contents trigger toxicity in the speech; triplet extraction involves extracting toxic entities and relations, with a self-checking strategy to ensure the quality of these triplets; and entity resolution merges nodes and relations with similar semantics respectively. Subsequently, given a downstream potentially toxic speech, we query the knowledge graph by *retrieval* and *ranking* for toxicity detection. Retrieval returns relevant paths to the speech, while ranking further filters noise and refines the triplets in paths. We then leverage the extracted triplets that serve as toxic prompts to boost the LLMs' capability in toxicity detection. **The entire process is training-free**. A showcase of our method is depicted in Figure 1. Our contributions are summarized as follows.

- We construct a novel meta-toxic knowledge graph. To our best knowledge, it is the first domain-specific knowledge graph for hate and toxic speech detection. We will open-source the knowledge graph upon paper acceptance.
- We propose an effective **training-free** method *MetaTox* for hate and toxic speech detection. In particular, *MetaTox* addresses the notorious false positive misjudgment issue, decreasing the ethical risk of hurting the freedom of speech.
- We conduct extensive experiments to evaluate the performance of MetaTox. Surprisingly, we find that small language models are more suitable for in-domain toxicity detection, while our method is superior in out-of-domain settings. This further sheds light on model selection in various toxicity detection scenarios.

2 Related Work

Previous methods can mainly be classified into rule-based, embedding-based, and transformer-based, and LLM-based approaches.

Rule-based methods (Chen et al., 2012; Gitari et al., 2015) rely on predefined rules that match specific patterns within the text. (Liu and Forss, 2015) uses the insults and swears words to form a dictionary for hatred and toxicity detection. Embedding-based approaches, on the other hand, combine text representations obtained by methods like word2vec (Djuric et al., 2015; Park and Fung, 2017). Transformer-based methods (D'Sa et al., 2020; Luu and Nguyen, 2021), typically perform fine-tuning on pre-trained models like BERT. Recent studies have explored the use of LLMs, which typically serve as a powerful data augmentation tool. Some methods (Lee et al., 2024) directly prompt LLMs to generate explanatory information about the text, which is then combined with the original text to train a detection model. Furthermore, some approaches (Yang et al., 2023; Zhang et al., 2024a) employ structured reasoning methods such as Chain-of-Thought (CoT) (Wei et al., 2024) and Tree-of-Thought (ToT) (Yao et al., 2024) to infer hidden semantics of input text. Some studies

aim to enhance the robustness of hatred and toxicity detection. For example, TextAttack (Morris et al., 2020) improves robustness through data augmentation, while Bespalov et al. (2023) proposes the ToxicTrap modular, leveraging adversarial training to strengthen model resilience.

However, previous approaches have either overlooked related real-world knowledge or relied solely on distilling the limited internal knowledge of LLMs. In contrast, our method is the first to build a hate-related KG and introduce it into hatred and toxicity detection, offering interpretive knowledge injection and multi-hop inference ability.

3 Methodology

In this section, we present *MetaTox*, our proposed method for enhancing hatred and toxicity detection using a meta-toxic knowledge graph. *MetaTox* consists of two stages: (1) construction of the meta-toxic knowledge graph through a three-step pipeline, and (2) querying the graph to enhance the downstream task of binary classification for toxicity detection. We begin with the data collection process, which lays the foundation for constructing the knowledge graph.

3.1 Data Collection

For meta-toxic knowledge graph construction, we leverage three well-established English datasets: HateXplain (Mathew et al., 2021), ToxicSpans (Pavlopoulos et al., 2021), and IHC (ElSherief et al., 2021). Since our goal is to enhance toxicity detection by supplementing the LLM with accurate and curated toxicity knowledge, the knowledge graph should serve as a domain-specific corpus that reflects only toxic content. Therefore, we exclusively retain toxic samples labeled as "toxic", "hate" or "offensive" from the training sets of these datasets to construct the knowledge graph.

For toxicity detection, we use the test sets from the three datasets, aligning each sample's label with either "toxic" or "non-toxic" to formalize a binary classification task. This enables us to evaluate the LLM's performance improvement in detecting potentially toxic speech with the support of our meta-toxic knowledge graph.

3.2 Graph Construction

We propose a three-step process to build the meta-toxic knowledge graph based on solely toxic samples: (1) *rationale reasoning*, which involves identifying the contents that trigger toxicity in speech, thereby uncovering implicit toxic meanings; (2) *triplet extraction*, where toxic entities and relations are extracted using a self-checking mechanism to ensure the triplets are correctly formatted and toxic; (3) *entity resolution*, which merges semantically similar entities and relations to reduce noise and shrink the graph. The three steps are illustrated in Figure 2 and described in detail below.

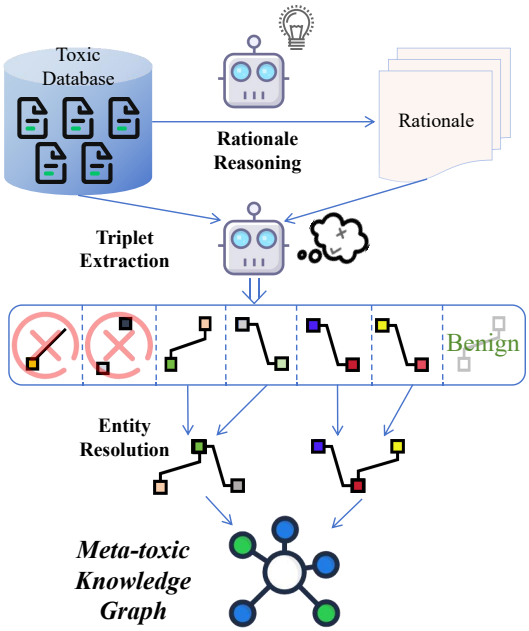

Figure 2: The pipeline of meta-toxic knowledge graph construction, including rationale reasoning, triplet extraction, and entity resolution.

3.2.1 Rationale Reasoning

To construct the meta-toxic knowledge graph from initial toxic speech, it is essential to first identify the core toxic elements, i.e. entities and relations, within the speech. However, toxic semantics are often implicit and abstract, involving concepts such as race, gender and religion, rather than specific named entities. This introduces challenges in directly extracting these elements in a single step.

To address this, we employ a rationale reasoning step prior to triplet extraction, which applies LLMs to articulate why the speech is considered toxic, with relevant elements leading to this conclusion naturally incorporated into the rationale. For example, by informing the LLM that "*white lives matter event*" belongs to hate speech, the LLM explains: "*a counter-movement to the Black Lives Matter movement, shows potential harm to the Black Lives Matter movement*". This explanation not only provides the context that triggers toxicity but also highlights the toxic entities in-

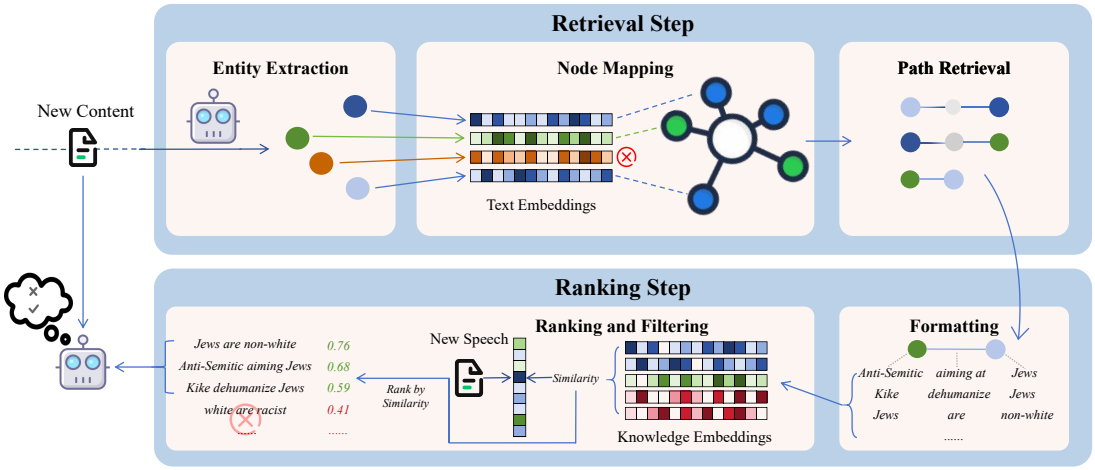

Figure 3: Graph query pipeline of *MetaTox*. We propose five steps including entity extraction, node mapping, path retrieval, formatting, and ranking and filtering to inject accurate knowledge to the LLM.

volved. Hence, this rationale reasoning step can be interpreted as a form of data augmentation, which draws out implicit toxic elements from the speech and makes them more explicit to better guide the LLM's reasoning logic towards toxicity, thereby reducing difficulties for subsequent triplets extracting. We leverage in-context learning to enhance LLM's ability to generate high-quality rationales, using carefully designed prompts. A detailed description of the prompt can be found in Appendix A.1.

It is important to emphasize that while LLMs exhibit excessive sensitivity and struggle to correctly judge when used for toxic content detection, our approach explicitly informs the model that the given text is toxic and leverages the model for reasoning rather than direct judgment or generation. This equates to **providing the model with prior knowledge, where the model only needs to deduce the reasons for the toxicity from the text.** A similar approach has also been adopted and validated in studies such as STaR (Zelikman et al., 2022), which guide models in generating new rationales based on correct answers, thereby helping them learn to solve increasingly difficult problems using the generated rationales.

3.2.2 Triplet Extraction

In this step, we take the toxic speech and the corresponding rationales as input and prompt the LLM to extract toxic triplets, represented in the Subject-Predicate-Object (SPO) format, such as "(*white lives matter, is against, black lives matter*)". Specifically, we implement a template that instructs the LLM to extract triplets triggering hatred, given two exemplars to guide the LLM in understanding the extraction process and outputting triplets in a unifying SPO format. It is worth noting that the extracted relations are not constrained to a predefined set but are determined by the LLM. Specifically, we include a few illustrative examples of potential relations in the prompt template and instruct the LLM to generate triplets using relations that best reflect the underlying semantics, rather than selecting from a fixed list. The details of the prompts are shown in Appendix A.2.

However, we observed two main issues with the extracted triplets. First, due to occasional imprecision in following instructions, the LLM may generate triplets that deviate from the standard SPO format, like omitting a subject or object. Second, the triplets may fail to capture toxic semantics, possibly due to inherent ambiguity in hate speech.

Thus, we propose a self-checking mechanism to refine the extracted triplets, as illustrated with pictorial cases in Figure 2. For triplets that are incorrectly formatted, we discard them using regular expressions. For triplets that fail to capture toxic semantics, we urge the LLM to filter out non-toxic triplets with few-shot prompting, ensuring that only those triplets capable of evoking hatred and toxicity are retained. Details of the filtering prompt are provided in Appendix A.3.

After applying the self-checking mechanism, we obtain a curated set of toxic triplets that explicitly reflect toxic semantics, effectively leveraging external domain knowledge.

3.2.3 Entity Resolution

In this step, we take the extracted triplet elements (entities and relations) as input, applying a clus-

tering algorithm to merge these elements and ultimately generate a meta-toxic knowledge graph with entities resolved. Unlike traditional entity resolution methods for knowledge graphs, our approach simplifies the standardization process by assigning identical names to entities within the same cluster, rather than relying on specific Internationalized Resource Identifiers (IRIs).

To perform clustering on the triplet elements, we first use a pre-trained BERT model (Devlin et al., 2019) to encode the textual attributes of entities and relations into embeddings. We then apply a clustering algorithm to group similar entities and relations respectively, based on their textual embeddings. For deduplication, we determine the name that appears most frequently within each cluster and assign it as the unified name for all elements in that cluster. In other words, we use clustering combined with a voting scheme to merge similar elements. This method not only helps resolve slight spelling variations, such as "*Jew*" and "*jew*", but also standardizes elements with different names referring to the same concept, such as "*LGBT*" and "*LGBTQ+*". To prevent over-merging, we set a relatively high similarity threshold.

The benefits of entity resolution are twofold. On the one hand, it reduces noise by eliminating unnecessary spelling differences and retaining representative expressions supported by most elements. On the other hand, it decreases the overall size of the graph, which in turn enhances the efficiency of graph retrieval.

3.3 Graph Query

Given a potentially toxic speech, we devote to querying on the meta-toxic knowledge graph to supplement speech-related toxic triplets, thereby providing the LLM with accurate, domain-specific guidance when judging the hatred and toxicity of the given speech. Briefly, we divide the query process into *Retrieval* (Entity Extraction, Node Mapping, Path Retrieval) and *Ranking* (Formatting, Ranking and Filtering). The overall query procedure is illustrated in Figure 3.

Entity Extraction aims to identify various entities in the given speech, treating them as candidate toxic entities for further validation. Given that explicit toxic entities may be absent, we employ the LLM to extract as many relevant entities as possible, including both specific named entities and broader concepts such as race, gender, and religion. The detailed instruction is shown in Appendix A.4.

Node Mapping faces the challenge that entities extracted from the speech may not exactly match the names of nodes in the meta-toxic knowledge graph. To resolve this, we formulate it as a dense retrieval task, where the most semantically similar node is mapped to each extracted entity based on textual embeddings generated by a pre-trained BERT model. At this point, candidate toxic entities are identified within the knowledge graph. The Faiss library (Douze et al., 2024) is applied to optimize retrieval efficiency.

Path Retrieval extracts coherent toxic knowledge contained from the graph based on the paths that connect previously mapped nodes through one or multi-hop relations. While a straightforward retrieval approach would be to extract all neighbors for each mapped node, this may result in recalling excessive unrelated triplets that could degrade toxicity detection performance, particularly when sensitive entities such as "*Jew*", "*Nigger*", "*Nazi*". To mitigate the noise introduced by unnecessary entities or relations, we enumerate pairwise combinations of the mapped nodes and retrieve the shortest path for each node pair. After splitting all the paths into SPO triplets, we take the union of them as the candidate retrieved triplets, reflecting toxicity knowledge potentially related to the text.

Formatting transforms the retrieved triplets into candidate knowledge with natural language by concatenating the SPO elements of each triplet, like "*white lives matter is against black lives matter*". After ranking and filtering each remaining knowledge will ultimately be sent to the LLM following this format as part of the prompts.

Ranking and Filtering is introduced to rank the candidate knowledge and discard irrelevant ones for denoising, as LLMs are sensitive to the prompt content. Specifically, we rank the candidate knowledge by sorting the cosine similarity between each candidate's knowledge and the input speech in descending order. This prioritizes toxicity knowledge that is more relevant to the speech. Additionally, we filter out less related knowledge based on similarity, which are considered as adverse noise.

After querying the meta-toxic knowledge graph, the ultimate toxicity knowledge is retrieved as supplementary information. We then insert the retrieved knowledge into the prompt template utilized for enhancing LLM's ability to detect hatred and toxicity. Two exemplars are designed to inform the LLM on utilizing external toxicity knowledge. The prompting template is presented in Appendix A.5.

4 Experiments

As outlined in Section 3.1, we construct three meta-toxic knowledge graphs based on the HateXplain, ToxicSpans, and IHC datasets, respectively, with Qwen2.5-14B-Instruct (Qwen) (Qwen Team, 2024). Then Qwen and Llama3.1-8B-Instruct (Llama) (AI@Meta, 2024) are employed to generate predictions. We evaluate *MetaTox* from three perspectives: **(1) Graph Evaluation**, which provides qualitative and quantitative analysis on the constructed graphs; **(2) Toxicity Prediction**, where *MetaTox* is compared to baseline methods under both in-domain and cross-domain settings to evaluate effectiveness and robustness of *MetaTox*; **(3) Case Studies**, which offer in-depth analyses by guiding LLMs to output reasoning paths, providing insights into how *MetaTox* enhances interpretability and reasoning abilities. To validate the rationale behind our pipeline design, we have further devised an ablation study, the details of which can be found in Appendix B.

4.1 Graph Evaluation

For the constructed KG, to verify whether the triplets contain hate-related semantics and adhere to the required format, we conduct a manual evaluation. To conserve human resources, we sampled 100 triplets from each of the three constructed KGs. Additionally, to mitigate annotators' potential bias towards classifying all triplets as hate-related, we introduce an equal proportion of triplets generated by LLMs that do not contain hate-related information, blending them in a 1:1 ratio.

We assembled a team of 5 annotators who are master's and PhD students, proficient in English, and have some relevant background knowledge. We paid the annotators a wage higher than the minimum wage in their local area. Before annotating, we informed them about the task and ensured that they fully understood and consented to annotate hate speech triplets.

Given the complexity of annotating hate speech triplets, we specifically emphasized three rules: 1) A triplet is considered correct only if it meets the required format and contains hate-related content. 2) When the correctness of a data point was unclear, annotators were instructed to search online first. 3) After completing the annotation, the five annotators should discuss to reach an agreement. The final accuracy of HateXplain, IHC, ToxicSpans KGs is 89%, 87% and 82%, respectively. Most of the triplets are correct, while the main types of errors are format issues and the use of pronouns without specifying the entities they referred to.

Table 1: Data Summary. † means that it has been reduced by 21.52% compared to before the merge; ‡ means 3.25% off.

	Toxic Samples	**Entities**	**Triplets**
IHC	7,373	20,043	25,534
HateXplain	10,273	24,442	33,276
ToxicSpans	9,905	21,667	30,347
Merged	27,551	51,917†	86,350‡

To illustrate the scale information of the KGs, we also conducted statistical analysis on the graph. We analyze the graph properties, particularly the number of nodes and relations, to assess the effectiveness of our data mining approach. Additionally, we examine the merging ratio by integrating meta-toxic knowledge graphs derived from different datasets, demonstrating how the meta-toxic knowledge graph expands. After merging three datasets, we observe a 21.52% reduction in the number of entities and a 3.25% reduction in the number of triplets in the merged graph. This suggests that the targets of toxic speech exhibit significant overlap across datasets, supporting the transferability of our approach. The detailed quantitative results are presented in Table 1. To facilitate the understanding of the knowledge graph structure, we illustrate sampled triplets in Appendix C.

4.2 Toxicity Prediction

We conduct experiments in both in-domain and cross-domain scenarios, comparing our method with vanilla LLM and the naive RAG-enhanced LLM for toxicity detection. Further, to assess whether LLMs can outperform fine-tuned small language models , we also include HateBERT (Caselli et al., 2021) and BERT (Devlin et al., 2019) as baselines. It's worth noting that HateBERT requires an initial retraining phase of 18 days and 2 million steps, followed by supervised fine-tuning (SFT) on downstream tasks. For the vanilla LLM, we directly input the test speech into the LLM and prompt it to provide the prediction. For the naive RAG method, we retrieve the top-2 most similar speeches from the training set as additional knowledge. To ensure a fair comparison, the training set used for the RAG method is the same as the data

Table 2: Results on using meta-toxic knowledge graph built from the **same** dataset, best results are highlighted.

Backbone Model		Qwen2.5-14B-Instruct			Llama3.1-8B-Instruct			HateBERT	BERT
Method		Vanilla LLM	RAG	*MetaTox*	Vanilla LLM	RAG	*MetaTox*	SFT	SFT
HateXplain	Acc.↑	70.95	73.13	73.39	63.36	69.59	68.87	77.96	**78.17**
	F1 ↑	64.04	70.18	72.48	48.11	62.02	62.50	**81.96**	81.34
	AUC↑	76.36	83.70	84.02	79.77	80.50	82.27	88.82	**90.44**
	FPR↓	66.62	48.72	32.10	88.75	40.69	38.74	31.33	**24.94**
IHC	Acc. ↑	66.34	66.71	73.65	50.79	64.29	69.04	78.35	**78.72**
	F1 ↑	66.32	66.67	69.95	48.03	64.23	68.55	**71.52**	70.57
	AUC↑	64.38	66.79	70.10	62.97	66.74	63.75	79.70	**79.74**
	FPR↓	48.42	43.24	**12.32**	77.63	51.58	34.23	17.49	14.19

Table 3: Results on using meta-toxic knowledge graph built from **another** dataset, best results are highlighted.

Backbone Model		Qwen2.5-14B-Instruct			Llama3.1-8B-Instruct			HateBERT	BERT
Method		Vanilla LLM	RAG	*MetaTox*	Vanilla LLM	RAG	*MetaTox*	SFT	SFT
HateXplain	Acc.↑	70.95	71.21	**73.28**	63.36	65.96	68.24	59.30	57.38
	F1↑	64.04	64.97	**72.38**	48.11	54.32	61.63	69.11	45.78
	AUC↑	76.36	81.29	**83.80**	79.77	81.50	81.67	72.63	58.57
	FPR↓	66.62	64.32	**32.10**	88.75	80.95	67.13	66.11	86.32
IHC	Acc. ↑	66.34	63.36	**73.42**	50.79	52.84	69.04	45.86	40.36
	F1 ↑	66.32	63.34	**69.42**	48.03	51.01	68.85	45.06	37.66
	AUC↑	64.38	67.21	**69.78**	62.97	61.15	63.63	39.88	33.27
	FPR↓	48.42	43.24	**11.64**	77.63	72.97	34.23	61.86	84.23

used to construct our meta-toxic knowledge graph.

To apply LLMs for classification, we follow the evaluation approach from MMLU (Hendrycks et al., 2021), where classification is determined by comparing the logit values of candidate options "a" (toxic) and "b" (non-toxic). Our evaluation metrics include classification accuracy (Acc.), F1 score (F1), and precision-recall area under the curve (AUC). Additionally, we calculate the false positive rate (FPR) to assess how effectively our method reduces false positives.

4.2.1 In-domain Setting

For the in-domain setting, the graph construction and toxicity detection are performed on the training and test sets of the same dataset, respectively.

As illustrated in Table 2, *MetaTox* demonstrates performance improvements across the HateXplain and IHC datasets with two backbone models, consistently outperforming baseline methods. The results reveal key insights into the effectiveness of our approach. For HateXplain, where toxic semantics are relatively explicit, the related knowledge is easier to retrieve by naive RAG, improving the performance to some extent. However, *MetaTox* outperforms naive RAG, primarily due to the more concise and relevant knowledge provided by KG.

In contrast, for IHC, where toxic semantics are more implicit, overall performance is lower compared to HateXplain. Using Qwen, while naive RAG yields only marginal improvement, *MetaTox* achieves a remarkable performance boost. This discrepancy may stem from the fact that documents retrieved by naive RAG often contain excessive unrelated information. In contrast, *MetaTox*, through the combined contributions of knowledge graph construction and query on the graph, not only retrieves the most related knowledge but also organizes it into concise triplets, significantly improving the model's understanding of the provided knowledge. When using the less powerful Llama for detection, *MetaTox* achieves performance slightly better than naive RAG, reflecting the advantage of its more effective knowledge integration, even with limited model capacity. In brief, our method effectively enhances the performance for in-domain scenario.

Compared to HateBERT and fine-tuned BERT, surprisingly, we find that **a simple BERT-based model, after training, can outperform a 14B LLM under in-domain scenarios**. The key distinction lies in the way of capturing toxicity: our method explicitly presents hate-related information with KG triplets, whereas HateBERT implicitly recognizes toxicity patterns by SFT. We speculate that small models' performance advantage attributes to supervised learning, which enables them to memorize toxicity knowledge within the

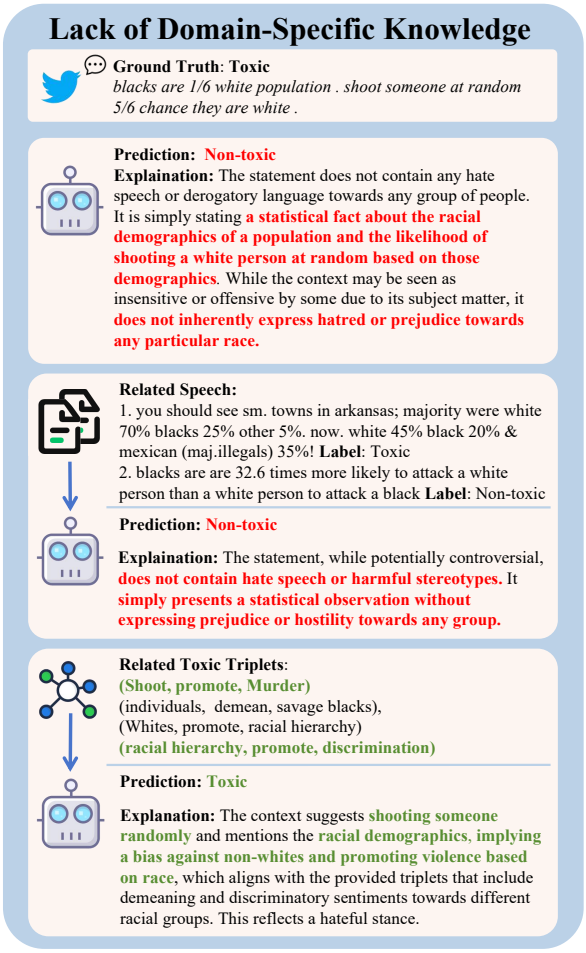

Figure 4: Example of false negative misjudgment caused by lack of domain-specific knowledge. From top to bottom, each block shows the ground truth label and context, followed by the predictions and reasoning of the vanilla LLM, naive RAG, and *MetaTox*.

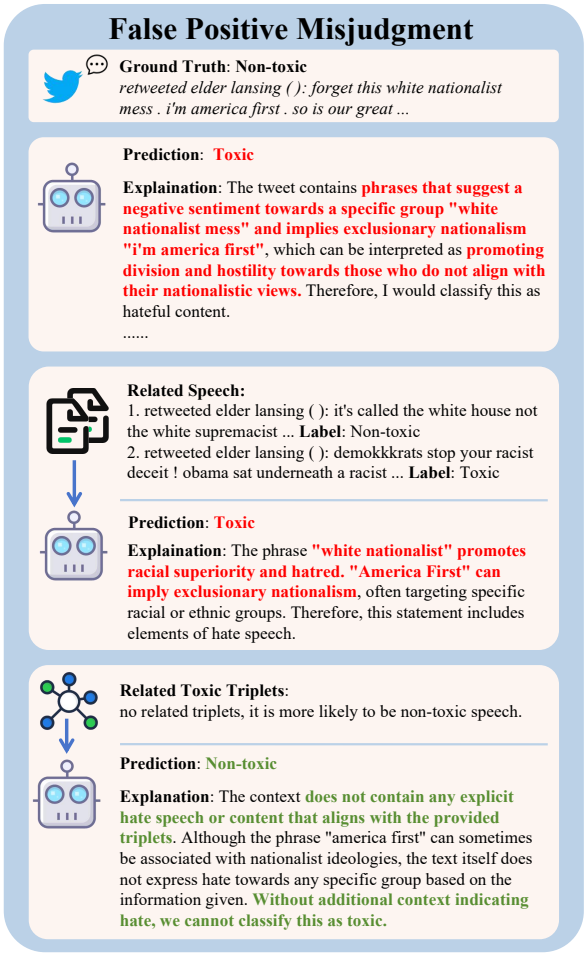

Figure 5: Example of false positive misjudgment caused by LLMs' extreme sensitivity towards certain phrases. The blocks are distributed the same as Figure 4.

training data (Chu et al., 2025). While HateBERT performs well on static datasets, toxic detection requires rapid updates in fast-evolving social media environments. This makes our approach, which constructs a knowledge graph that is easier to edit and expand, more suitable for such tasks. Therefore, we conduct experiments in a cross-domain setting as discussed below.

4.2.2 Cross-domain Setting

For the cross-domain setting, the knowledge graph is constructed using the training set of ToxicSpans, while detection is performed on test sets of HateXplain and IHC. Correspondingly, naive RAG is only allowed to retrieve documents from Toxicspans.

The experimental results are presented in Table 3. Notably, we observe a performance drop with naive RAG on IHC and negligible improvement on HateXplain. This can be attributed to the different facets of toxic semantics between ToxicSpans and test datasets. As analyzed in Section 4.2.1, the text representation-based method, HateBERT and BERT, struggles in OOD scenarios, resulting in a significant performance drop. HateBERT performs slightly better than BERT, likely due to its extensive retraining on a relatively large toxicity corpora, RAL-E dataset. In contrast, *MetaTox* still maintains a notable improvement in performance, because by incorporating external accurate toxicity knowledge, LLMs can uncover the underlying, invariant toxicity essence concealed within the diverse expressions of toxic content, therefore enhancing generalization.

It is worth noting that our method outperforms other LLM-based approaches across all scenarios, achieving the lowest or near-lowest false positive rate, effectively mitigating the risk of infringing on freedom of speech caused by false positives.

4.3 Case Study

To further analyze how *MetaTox* incorporates domain-specific knowledge and mitigates false positive errors, we present two case studies with the reasoning explanations provided by the LLM.

As shown in Figure 4, the original speech implicitly promotes toxicity by stirring up antagonism between blacks and whites. The word "*shoot*" also emphasizes the racial disparities. The vanilla LLM incorrectly focuses on numbers like "*1/6*" and "*5/6*", which misleads the LLM to interpret the speech as a statistical analysis rather than a toxic trigger. When enhancing with naive RAG, the retrieved relevant texts contain statistics like "*70%*", and "*32.6 times*", indicating that the retriever fails to understand the intended meaning of the text. This further deepened the model's misunderstanding, leading to misjudgments. On the contrary, *MetaTox* correctly identifies pivotal elements like "*shoot*", and "*racial hierarchy*", guiding the model to correctly interpret the speech as promoting race-based violence and predict correctly.

As shown in Figure 5, the original context emphasizes unity by stating "*I'm America first,*" while suggesting ignoring toxic content presented by "*white nationalist mess*". However, the vanilla LLM misclassifies it as toxic because it overly focuses on phrases like "*America First*" and "*white nationalist mess*", incorrectly interpreting them as indicators of toxicity. When enhanced with naive RAG, the LLM retrieves two related retweets for the same post. However, since both retweets focus on racial issues, even though they express different opinions, they fail to shift the LLM's focus away from racial attributes, resulting in a false positive misjudgment. In contrast, *MetaTox* effectively filters the retrieved triplets, directing the model to follow the prompt instructions that "*when there are no related triplets, the text is more likely to be non-toxic*", which significantly reduces the false positive misjudgment.

5 Conclusion

In this paper, we proposed a novel method called *MetaTox* to address both false positive and false negative misjudgments caused by the lack of domain-specific knowledge and LLMs' extreme sensitivity. First, we leverage LLMs to extract toxic content through a three-step pipeline, which builds the meta-toxic knowledge graph. Next, we query the graph with retrieval and ranking processes to provide additional, relevant toxic knowledge. Extensive experiments and detailed case studies across various datasets show that *MetaTox* significantly lowers the false positive rate while improving overall toxicity detection performance in OOD scenarios. In addition, under in-domain scenarios, we surprisingly find that small language models are more competent.

Acknowledgement

This work was supported by Shanghai "Science and Technology Innovation Action Plan" Project under Grant No.23511100700 and National Natural Science Foundation of China under Grant 42375146.

Limitation

Our study still has several limitations. First, due to the computational constraint, we did not conduct experiments with larger LLMs, suggesting that the full potential of our method remains to be further explored. Future research could leverage more powerful LLMs to construct even more comprehensive meta-toxic knowledge graphs. Additionally, our method is currently limited to binary classification in English-language scenarios. Future work can extend our approach to multi-modal, multilingual, and multi-cultural tasks, thereby broadening its applicability across diverse contexts.

Ethical Statement

Our research focuses on toxic speech detection, primarily addressing two key issues faced by LLMs in this task: false negatives caused by a lack of domain-specific knowledge, and false positives resulting from excessive sensitivity to certain content. Our primary goal is to facilitate more accurate detection of toxicity, thereby alleviating the issue of false positive misjudgments and protecting freedom of speech, which aims to contribute to the creation of a more harmonious online environment. We construct a meta-toxic knowledge graph filled with toxic content. However, there is also a risk of potential misuse of our proposed method. Specifically, our methods might be misused to generate toxic speech. It is essential to emphasize that our work focuses on detecting toxic speech while safeguarding freedom of speech, instead of censorship. The datasets we used are all existing open-source datasets, aligning with their intention for scientific research. We also adhered to the MIT license for HateXplain and IHC datasets, and the CC0-1.0 license for the ToxicSpans dataset.

References

AI@Meta. 2024. Llama 3 model card.

Dmitriy Bespalov, Sourav Bhabesh, Yi Xiang, Liutong Zhou, and Yanjun Qi. 2023. Towards building a robust toxicity predictor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 581–598, Toronto, Canada. Association for Computational Linguistics.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *2012 international conference on privacy, security, risk and trust and 2012 international confernece on social computing*, pages 71–80. IEEE.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *Preprint*, arXiv:2501.17161.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Ashwin Geet D'Sa, Irina Illina, and Dominique Fohr. 2020. Bert and fasttext embeddings for automatic detection of toxic speech. In *2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)*, pages 1–5.

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*.

Mai ElSherief, Caleb Ziems, David Muchlinski, Vaishnavi Anupindi, Jordyn Seybolt, Munmun De Choudhury, and Diyi Yang. 2021. Latent hatred: A benchmark for understanding implicit hate speech. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 345–363, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Nayeon Lee, Chani Jung, Junho Myung, Jiho Jin, Jose Camacho-Collados, Juho Kim, and Alice Oh. 2024. Exploring cross-cultural differences in English hate speech annotations: From dataset construction to analysis. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4205–4224, Mexico City, Mexico. Association for Computational Linguistics.

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 *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Shuhua Liu and Thomas Forss. 2015. New classification models for detecting hate and violence web content. In *2015 7th international joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3K)*, volume 1, pages 487–495. IEEE.

Son T Luu and Ngan Luu-Thuy Nguyen. 2021. Uit-isenlp at semeval-2021 task 5: Toxic spans detection with bilstm-crf and toxicbert comment classification. *arXiv preprint arXiv:2104.10100*.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 14867–14875.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the*

2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 119–126, Online. Association for Computational Linguistics.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*.

John Pavlopoulos, Jeffrey Sorensen, Léo Laugier, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection. In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*, pages 59–69.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Robert Mark Simpson. 2013. Dignity, harm, and hate speech. *Law and Philosophy*, 32(6):701–728.

Robert Mark Simpson. 2019. 'won't somebody please think of the children?'hate speech, harm, and childhood. *Law and Philosophy*, 38(1):79–108.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the second workshop on language in social media*, pages 19–26.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Yongjin Yang, Joonkee Kim, Yujin Kim, Namgyu Ho, James Thorne, and Se-Young Yun. 2023. HARE: Explainable hate speech detection with step-by-step reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5490–5505, Singapore. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488. Curran Associates, Inc.

Jiang Zhang, Qiong Wu, Yiming Xu, Cheng Cao, Zheng Du, and Konstantinos Psounis. 2024a. Efficient toxic content detection by bootstrapping and distilling large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(19):21779–21787.

Min Zhang, Jianfeng He, Taoran Ji, and Chang-Tien Lu. 2024b. Don't go to extremes: Revealing the excessive sensitivity and calibration limitations of LLMs in implicit hate speech detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12073–12086, Bangkok, Thailand. Association for Computational Linguistics.

A Prompts

A.1 Rationale Reasoning Prompt

Rationale Reasoning Prompt

<USER>: You are a helpful assistant aiming at hateful speech detection. For each message, give the reason why it is hateful. Here are 2 examples:
Example 1:
<USER>: Context 1: The Democrat party is like a dog sniffing its own tail. It stinks.
<ASSISTANT>: Analysis 1: The context is considered hateful because it uses a derogatory comparison to dehumanize a political group (the Democrat party) by likening it to a dog, which implies foolishness or absurdity. Additionally, the phrase 'it stinks' conveys a strong negative judgment, suggesting disdain and contempt. Such language can foster hostility and division, contributing to a culture of hate rather than constructive discourse.
Example 2:
<USER>: Context 2: The working middle class is an easy target! Have fun bleeding us dry with more taxes & Sisyphean social programs, elite upper class robber barons!
<ASSISTANT>: Analysis 2: The context is considered hateful because it employs inflammatory language by labeling the upper class as 'robber barons',which evokes historical connotations of greed and exploitation. This framing dehumanizes individuals within that socioeconomic group and fosters animosity. Additionally, the phrase 'bleeding us dry' expresses a strong sense of victimization and resentment, promoting division between classes rather than encouraging constructive dialogue. Such rhetoric can perpetuate class hatred and animosity.
<USER>: New Context: context

<ASSISTANT>: Analysis:

A.2 Triplet Extraction Prompt

Triplet Extraction Prompt

<USER>: You are a helpful assistant skilled in inductive reasoning and summarization. For each message, you will receive an original context and an analysis. Your task is to extract triplets that trigger hate.
In general, the context is more important and the analysis is just for assisting. The output should start with 'Output' and be with the format (head object A, predicate, tail object B) like (King, is, man). Also, there is a special condition that if the two phrase or word co-appear, they will lead to hate, you can use the "co-appear" as the relation and the two phrases or words as the head object and the tail object. There are some recommanded relations, you can use them if they are suitable: is, demean, promote, refer to, imply, link, co-appear, insult, downplay, etc. For entities, it is better to use the original words instead of the conclusion of the analysis. Here are two examples:
Example 1:
<USER>: Input 1: Context: the persecution of christians isn't as bad as the persecution of muslims.
Analysis: The context is considered hateful because it downplays the suffering of one group (Christians) by comparing it to the suffering of another (Muslims).
This minimizes the severity of the persecution faced by Muslims and can be seen as a form of victim-blaming.
Furthermore, the implication that Christians are being persecuted is not universally recognized or acknowledged, and this statement can be seen as perpetuating a narrative that is divisive and can be hurtful to those who experience persecution.
<ASSISTANT>: Output 1:
(persecution of Christians, is not as bad as, persecution of Muslims),
(comparing to suffering of Muslims, downplay, suffering of Christians)
Example 2:
<USER>: Input 2: Context: just shows how many retarded jerks we have living here buying everything he has. yea, that means your kids you stupid clueless parents. wise up.
Analysis: The context is considered hateful because it uses derogatory language to insult individuals based on their socioeconomic status (r̈etarded jerks)̈ and their ability to make informed decisions (c̈lueless parents)̈.
The use of the word'retarded' is particularly egregious as it's a term that has been historically used to demean people with intellectual disabilities.
This language is not only hurtful but also discriminatory, perpetuating stigma and hate towards those who may be vulnerable.
The final sentence, ẅise up,ïs a condescending and aggressive statement that further escalates the hate speech.
<ASSISTANT>: Output 2:
(retarded jerk, co-appear, parents),
(retarded jerk, insult, individuals),
(wise up, condescend, aggressive statement)
<USER>: New Input: context

<ASSISTANT>: Output:

A.3 Self-checking Prompt

Self-checking Prompt

<USER>: Please select the triplet evoking the hate from the following triplets began with 'Input:' and within the []. In general, the triplet should be in form like (Object A, relation, Object B), where A is a noun phrase or word, relation is a predicate(verb or a linking verb).If the input triplets don't fit the format, you should rewrite the input triplets first.
The number of triplet is not fixed, can be one or several, you are required to select all hate triggered triplets. Finally, output the hate triggered triplets in [] only.
Here are two examples:
Example 1:
<USER>: Input 1: [(downplays suffering of Christians, by comparing to, suffering of Muslims),(persecution of Christians, is not as bad as, persecution of Muslims),(minimizes severity of persecution, faced by, Muslims),(implication that Christians are being persecuted, is not universally recognized, acknowledgment)]
<ASSISTANT>: Output 1: [(persecution of Christians, is not as bad as, persecution of Muslims),(comparing to suffering of Muslims, downplay, suffering of Christians),]
Example 2:
<USER>: Input 2: [(retarded jerks, shows, socioeconomic status),(clueless parents, buying everything, he has),(derogatory language, insult, individuals),(wise up, condescend, aggressive statement)]
<ASSISTANT>: Output 2: [(retarded jerk, insult, individuals), (wise up, condescend, aggressive statement)]
<USER>: Input: context
<ASSISTANT>: Output:

A.4 NER Prompt

Named Entity Recognition Prompt

Extract and list all distinct entities from the following text. Only return the entities as a comma-separated list like following template:
Input: <sentence>
Output: <entity1>,<entity2>, <entity3> ...
Example:
Text: u are the best gift for Muslim as you r a pig.
Entities: gift,Muslim,pig
Text: text
Entities:

A.5 Query Prompt

Query Prompt

<Instructions>: You are a helpful assistant skilled in inductive reasoning. For each message, you will receive a context and related triplets as reference. All the triplets will lead to hate. If the context contains the same meaning with one of the triplets, it is hateful. Otherwise, it is more possible to be benign, but you should leverage your knowledge to judge it. Your task is to determine whether the context is hateful or not with a single aplphabet letter among given answer choices a and b. You should only output a or b.
a: {{Hateful}}
b: {{Non-hatefule}}
Here are 2 examples:
Example 1:
<USER>: TRIPLETS: (niggers, is described as, stupid),
CONTEXT: Niggers are stupid.
<ASSISTANT>: a
Example 2:
<USER>: TRIPLETS: (white woman, shows, racism), (racism, aiming at, black man)
CONTEXT: That old white lady condemns any type of racism against black men.
<ASSISTANT>: b
New Input:
<USER>: TRIPLETS: {triplets}
CONTEXT: {context}
<ASSISTANT>:

B Ablation Study

To validate the effectiveness and rationality of each step in our retrieval pipeline, we designed an ablation study. We selected Qwen-14B-Instruct as the backbone model and conducted the experiment on the HateXplain dataset. Specifically, we compared the settings without rerank & filter and those replacing shortest path search with 1-hop neighbor search. The results are presented in Table 4.

By comparing the results of the first and second rows, we find that rank & filter significantly improves performance. Similarly, comparing the second and third rows shows that shortest path search outperforms 1-hop neighbor search. This may be because the elements involved in shortest path search are more relevant, reducing input noise.

Table 4: The ablation results on the HateXplain dataset based on Qwen2.5-14B-Instruct.

method	Acc.	F1
MetaTox(w/ rank & filter, shortest path)	**73.39**	**72.48**
w/o rank & filter, w/ shortest path	70.17	71.73
w/o rank & filter, w/ 1-hop subgraph	65.38	69.18

C Examples in Knowledge Graph

```
{'nodes': [
    'nigger', 'stupid','Jews',
    'Muslims', 'terrorists',
    'intellectually inferior',
    'immigrants', 'demean country',
    'violence', 'culture',
    'feminist', 'man-hating',
    'useless', 'shoplifter'
],
'triplets' : [
    ['nigger','is','stupid'],
    ['stupid','coappear','Muslims'],
    ['Muslims','are','terrorists'],
    ['nigger','implies','intellectually inferior'],
    ['immigrants','demean','country'],
    ['nigger','is responsible for','violence'],
    ['Muslims','demean','sincerity'],
    ['feminist','are','man-hating'],
    ['immigrants','are','useless'],
    ['immigrants','ruin','culture'],
    ['nigger','is','shoplifter']
]}
```

Listing 1: Examples in Knowledge Graph.