Implanting LLM's Knowledge via Reading Comprehension Tree for Toxicity Detection

Hankun Kang[1], **Tieyun Qian**[1,2]*
[1] School of Computer Science, Wuhan University, China
[2] Intellectual Computing Laboratory for Cultural Heritage, Wuhan University, China
{kanghankun,qty}@whu.edu.cn

Abstract

Warning: *This paper contains several toxic and offensive statements.*

Toxicity detection plays a crucial role in maintaining the peace of the society. Existing methods can be roughly categorized as small language model (SLM) based and large language model (LLM) based. However, due to the limitation of SLMs on general knowledge and the potential embedded bias in LLMs despite their large amount of knowledge, it is not a good idea to detect toxicity only with either SLM or LLM based method.

In this work, we propose to *implant LLM's knowledge into SLM based methods* such that we can stick to both types of models' strengths. To this end, we *develop a reading comprehension (RC) tree* to transfer knowledge between two models. Specifically, we first *construct the RC tree*, *from an extensive to intensive reading perspective*, to capture the local and global information in the text. We then model samples encoded by SLM and knowledge extracted from LLM as two distributions using the constructed RT tree. We finally *transfer knowledge via optimal transportation* between two distributions. Extensive experiments prove the effectiveness of our method on real-world and machine-generated datasets. Our code and data are available at https://github.com/khk-abc/toxic-detection.

1 Introduction

With the prevailing of online communication, toxic content has been growing in recent years on the websites. In addition, the malicious use of large language models makes toxic language more common and difficult to detect. Toxic language may result in serious hazards such as the promotion of violent crimes and discrimination against marginalized groups. Due to its harmfulness to individuals and society (Shakespeare, 2013), toxic language has become a serious concern.

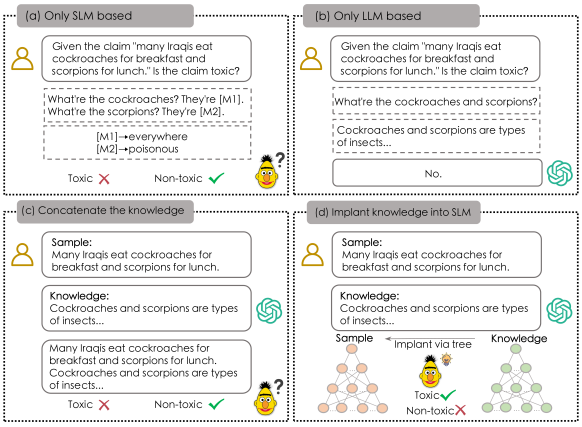

Figure 1: Illustration of the shortcomings of SLM (here RoBERTa) and LLM (here ChatGPT) based methods for toxicity detection.

The harm of toxic content can be prevented either by pre-detection before online release or by post-detection before spreading to a broader audience, and thus the detection of toxic content plays a crucial role in controlling toxicity. Existing methods for this purpose can be roughly categorized into small language model (SLM) based and large language model (LLM) based types. The fine-tuned SLM based methods (Antypas and Camacho-Collados, 2023; He et al., 2023) are the mainstream. Due to the limited knowledge, the performance of these methods is not very satisfying. Recently, a LLM based method (Zhang et al., 2023a) is presented for this task. However, the potential of LLMs has not been fully exploited. More importantly, LLMs have their own bias which may lead to new safety issues (Liyanage and Ranaweera, 2023; Leong et al., 2023; Zhang et al., 2024).

As Fig. 1 (a) and (b) show, neither the RoBERTa (Liu et al., 2019) based method nor ChatGPT (OpenAI, 2022) can make a correct prediction about the given sample. To have a close look, we let two language models paraphrase '*cockroaches*' and '*scorpions*' by masking these two words for

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2024, pages 947–962
August 11-16, 2024

RoBERTa and directly asking ChatGPT. It can be seen that RoBERTa associates '*cockroaches*' with '*everywhere*' and '*scorpions*' with '*poisonous*' while ChatGPT actually knows they are both '*insects*'. We hence infer that the wrong prediction of RoBERTa based method is caused by the limited knowledge and that of ChatGPT might be due to the potential bias embedded in the model (Lee et al., 2023; Felkner et al., 2023).

In view of the shortcomings of SLM and LLM based methods, a natural idea is if we can combine both types of language models to leverage their strengthes. However, this is not a trivial task since toxicity detection requires the deep understanding of the text. A simple combination by treating LLM's knowledge as the context of SLM or concatenating embeddings from two models does not improve SLM much, as shown in Fig. 1 (c). This drives us to solicit an effective way to comprehend the text. Fortunately, humans have well established the theory for enhancing reading skills (Saricoban, 2002; Toprak and Almacioğlu, 2009; Pressley and Afflerbach, 2012), and a pioneering work (Luo et al., 2019; Zhang et al., 2021) has solved the summarization problem by simulating the human reading process.

In this paper, we propose to implant LLM's knowledge into SLM based method via a reading comprehension tree for toxicity detection. (1) We first construct the reading-comprehension (RC) tree to gain a deep understanding of the text, i.e., we first form local rough understanding via scanning words and phrases in the text and then incorporate the global comprehensive semantics of the entire text. (2) We then use the RC tree to model the sample encoded by SLM and knowledge from LLM as two distributions, i.e., we add task-specific features to the root of the tree to get multi-grained information distributions in a top-to-down manner. (3) We finally introduce the optimal transport method to fulfill knowledge implanting task, i.e., we apply optimal transport to LLM's knowledge distribution and the sample's distribution.

Our main contributions are as follows:

- We propose a novel framework which can effectively implant LLM's knowledge into the SLM based method for toxicity detection.
- We construct a reading comprehension (RC) tree by simulating the human cognitive process for getting a deep understanding of the text, thereby facilitating the multi-grained knowledge transfer from LLM to SLM.
- Extensive experimental results on three widely used toxic detection datasets across different languages and sources prove the superiority of our proposed framework.

2 Related Work

Toxic Language Detection. Toxicity means language contains profanity and is rude or disrespectful, leading to disharmony and conflicts in society (Feldman et al., 2015; Dixon et al., 2018). There are many types of toxicity, e.g., cyberbully (Slonje et al., 2013; Kowalski, 2018), hate speech (Del Vigna12 et al., 2017; Fortuna and Nunes, 2018), offensive language (Chen et al., 2012; Risch et al., 2020), and social bias (Garb, 1997; Liang et al., 2021). Several datasets have been collected (Zampieri et al., 2019; Sap et al., 2020; Haber et al., 2023; Mathew et al., 2021) across different types of toxicity. Various methods are proposed to tackle toxicity detection (Davidson et al., 2019; Ali et al., 2022; Gupta et al., 2023; Antypas and Camacho-Collados, 2023), e.g., multi-task learning based (Kapil and Ekbal, 2020) and prompt tuning based (He et al., 2023). However, most of existing methods utilize SLMs to capture the semantics of samples. Hence their performance is limited by SLMs' knowledge.

Large Language Models (LLMs) have shown impressive performance on many tasks (Wei et al., 2022; Singhal et al., 2023; Madani et al., 2023; MacNeil et al., 2023). Recently, a few LLMs based methods are also proposed for toxicity detection (Zhang et al., 2023b) or fairness and fact-checking (Zhang et al., 2023a). However, due to the factors like unsafe training data and policy decisions, it has been observed that LLMs contain toxicity like societal biases (Abid et al., 2021; Ferrara, 2023; Ray, 2023). Consequently, there are potential risks when deploying LLMs in various tasks.

Reading Comprehension is a cognitive and constructive process about text (Woolley and Woolley, 2011). It is necessary to get a comprehensive understanding of the text for further applications (Luo et al., 2019; Zhang et al., 2021). In the field of natural language processing, reading comprehension is usually modeled as question answering (Rajpurkar et al., 2018; Joshi et al., 2017) and applied to summarization tasks (Song et al., 2020; Jia et al., 2021; Gu et al., 2021). Existing methods mainly

focus on evaluating the importance of text spans to specific questions or summarizations (Ye et al., 2020; Xu et al., 2021; Bao et al., 2022; Raina and Gales, 2022; Luo et al., 2022). In contrast, our method aims at comprehending text spans themselves and captures the multi-granularity semantics into a reading-comprehension tree, which is suitable for diversified downstream tasks including the knowledge transfer in our work.

3 Methods

This section illustrates the details of our reading comprehension tree (RCT) based method.

3.1 Problem Definition

The task of toxicity detection is to recognize the multiple aspects of toxic text (Lu et al., 2023), including *toxicity*, *toxic type*, *expression*, and *targeted groups*. In this work, we aim to leverage both the fine-tuned representations from a small language model (SLM) and the knowledge from a large language model (LLM) to solve this problem.

Formally, given a set of training data $\mathcal{T}^{\mathcal{S}}$ for fine-tuning a SLM, and a text sample $S = \{S_1, S_2, \cdots, S_N\}$ (N is the text length) in the test data $\mathcal{T}^{\mathcal{T}}$, we first get a fine-tuned representation S_s. We also extract the knowledge K_s [1] from a LLM for the sample S. We finally obtain a toxicity detector $\mathcal{F}$ by implanting K_s into S_s via a reading comprehension tree T_{rc}.

$$S_{[y_{tox}, y_{type}, y_{exp}, y_{tar}]} = \mathcal{F}(S, S_s, K_s, T_{rc}), \quad (1)$$

where y_{tox}, y_{type}, y_{exp}, y_{tar} denote the labels for *toxicity*, *toxic type*, *expression*, and *targeted groups*, respectively.

3.2 Reading Comprehension (RC) Tree Construction

This section presents the construction of the reading comprehension (RC) tree, which is inspired by the cognitive process and can be used to facilitate the knowledge transfer.

During human's reading process, it is important to first extensively scan and then intensively read the text. The extensive scanning forms the local rough understanding of the text while the intensive reading obtains the global comprehensive semantics. To mimic this process, we first store the words into the leaf nodes. We then *form the local rough understanding* by convoluting over several local nodes at the lower layer into a node at the upper layer, and we also *get the global comprehensive semantics* by summarizing the lower layer into the upper layer via a cross-attention mechanism. Recursively, we obtain the comprehensive semantics of the whole text in the tree.

3.2.1 Construction Process

We mimic the cognitive process in a local to global way during the construction process of the RC tree.

Forming the local rough understanding Given a sequence of words in a text $t = \{t_1, t_2, \cdots, t_N\}$, we first treat the representation of each word as a leaf node in the bottom layer, i.e., the D-th layer T_{rc}^D in the RC tree, where D is the depth of the tree. We then build an upper layer to get an aggregated local information by applying a convolutional operation to nodes at the lower layer as follows:

$$T_{rc}^{i-1} = CNN(T_{rc}^i), i = 1, 2, \cdots, D, \quad (2)$$

where $T_{rc}^{i-1} \in \mathbb{R}^{L_{i-1} \times d}$ and $T_{rc}^i \in \mathbb{R}^{L_i \times d}$ denote the i-1^{th} and i^{th} layer of the tree, and L_{i-1} and L_i denote the number of nodes at the i-1^{th} and i^{th} layer, respectively. d is the dimension of the representation of the node feature. Note that a small i denotes that the layer is close to the root, e.g., T_{rc}^1 is the root of the tree. In addition, the number of nodes in each layer is determined by the following formula, which is controlled by the window of CNN:

$$L_{i-1} = \lfloor (L_i - 1)/2 \rfloor + 1 \quad (3)$$

Getting the global comprehensive semantics We get the global sentence-level semantics by applying cross-attention to two neighboring layers T_{rc}^{i-1} and T_{rc}^i of the tree. Specifically, we compute the global semantic relevance scores $s_{i-1,i} \in \mathbb{R}^{L_{i-1} \times L_i}$ between all node pairs in two neighboring layers as:

$$s_{i-1,i} = Attention(T_{rc}^{i-1}, T_{rc}^i), \quad (4)$$

Further, we incorporate the global features of nodes at the lower T_{rc}^i layer into the features of nodes at the upper T_{rc}^{i-1} layer using the following formula:

$$T_{rc}^{i-1} = T_{rc}^{i-1} + s_{i-1,i} \cdot T_{rc}^i, \quad (5)$$

In this way, we recursively summarize the local and global information in the text, corresponding to the process of extensive scanning and intensive reading.

[1] It is easy to get S_s and K_s using existing methods and the detail will be presented later.

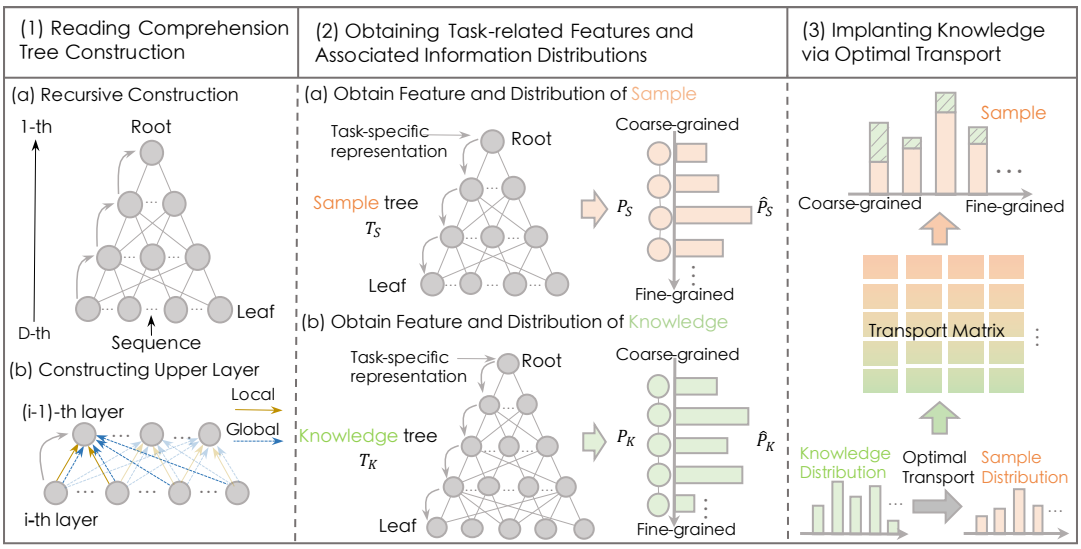

Figure 2: The framework of our proposed RCT model. 'Sample' denotes the text content and 'Knowledge' denotes the knowledge extracted from a LLM.

3.2.2 RC Tree for Sample and Knowledge

Fine-tuning sample's representation by SLM We utilize a SLM as the encoder for fine-tuning samples' representations. We then employ the sample's representations S_s as the nodes in the leaf layer of the RC tree. In this study, we use BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) as the representative of SLMs.

Obtaining knowledge from LLM To acquire the knowledge towards each sample from LLMs, we first employ a LLM (*gpt-3.5-turbo* as the representative of LLMs here) as the content moderator.

$$\mathcal{M} = \mathcal{R}(LLM, O_{\mathcal{R}}), \quad (6)$$

where $\mathcal{R}$ means that we employ a LLM as a content moderator role, and $O_{\mathcal{R}}$ is the instruction for the role. Then we instruct the moderator $\mathcal{M}$ to extract the knowledge K about the given sample S.

$$K = \mathcal{M}(S, O_K), \quad (7)$$

where O_K [2] denotes the instruction for extracting the knowledge K. The knowledge K is further encoded as K_s using the same encoder for S_s.

Building RC tree for sample and knowledge After getting representations for the sample S_s and knowledge K_s, we construct their reading comprehension trees, i.e., $T_{rc,S}$ and $T_{rc,K}$, by applying the RC tree construction process to S_s and K_s as the input text span.

3.3 Obtaining Task-related Features and Associated Information Distributions

Since the RC tree encapsulates the comprehensive semantics of the sample and knowledge, we employ them for obtaining their task-related features and associated information distributions. This is for further implanting the task-related semantics of knowledge into samples.

Starting from the root of the tree, we recursively obtain task-related multi-grained features in the constructed trees $T_{rc,S}$ and $T_{rc,K}$ in a top-to-down manner. Specifically, we first get the task representation by combining prompting, task related question like '*Is it toxic*?', and the definition of a specific task (Details are in Appendix) as:

$$Q_{task} = E(S, [M]_{task}) + E_q + E_d, \quad (8)$$

where $E(S, [M]_{task})$ denotes the task-specific prompt, and E_q and E_d denote the task related question and the definitions of a task, respectively.

Then we add the task-specific representation to the root of the tree (taking $T_{rc,S}$ as an illustration example):

$$R_S^1 = T_{rc,S}^1 + Q_{task}, \quad (9)$$

Recursively, we obtain the task-related multi-grained features of nodes at the lower layer based on the cross-attention operation.

$$rs_{i,i-1} = Attention(T_{rc,S}^i, R_S^{i-1} + Q_{task}), \quad (10)$$

where $rs_{i,i-1} \in \mathbb{R}^{L_i \times L_{i-1}}$ denotes the relevance scores between the features of nodes at the $T_{rc,S}^i$ layer and the combined feature of R_S^{i-1} and Q_{task}. Further, we can obtain the task-related multi-grained node features at the layer $T_{rc,S}^i$ as:

$$R_S^i = T_{rc,S}^i + rs_{i,i-1} \cdot (R_S^{i-1} + Q_{task}), \quad (11)$$

[2] The instructions $O_{\mathcal{R}}$ and O_K are given in Appendix A.2

where $R_S^{i-1} \in \mathbb{R}^{L_{i-1} \times d}$ and $R_S^i \in \mathbb{R}^{L_i \times d}$ denote the task-related features at i-1th and i^{th} layer. The relative importance of each node at the current layer can be computed by:

$$g_j = \sigma(FFN(R^i_{S_j}, Q_{task})) \tag{12}$$

The features of nodes at the i^{th} layer are as follows:

$$P_S^i = \sum_{j=1}^{L_i} g_j \cdot R^i_{S_j}, \tag{13}$$

where P_S^i is the i-th element of $P_S \in \mathbb{R}^{D \times d}$, denoting the task-related features of nodes at the i^{th} layer. And d denotes the dimension of the encoded features. Finally, we obtain the associated task-related information distribution by applying a forward feed network.

$$\widehat{P}_S^i = FFN(P_S^i), \tag{14}$$

where $\widehat{P}_S^i$ denotes i-th element of the task-related information distribution $\widehat{P}_S \in \mathbb{R}^D$ for the sample S.

By now, we have task-related features P_S^i and the associated information distribution $\widehat{P}_S^i$ ($i = 1, 2, \cdots, D_S$) for the sample S. Similarly, we have P_K^i and $\widehat{P}_K^i$ ($i = 1, 2, \cdots, D_K$) for the knowledge K.

3.4 Implanting Knowledge via Optimal Transport

Optimal transport (OT) (Villani et al., 2009) denotes transporting a distribution $A \in \mathbb{R}^n$ to another distribution $B \in \mathbb{R}^m$. The original target in OT problem is hard to optimize and Sinkhorn and Knopp (1967); Knight (2008) propose an equivalent optimization target and provide the solution as:

$$\mathbf{P}^*_{\mathrm{ij}} = \mathrm{u_i v_j e}^{-\lambda \mathbf{C}_{\mathrm{ij}}}, \tag{15}$$

where u_i and v_j are the i and j elements of $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$ intermediate variables for iterative optimization. $\mathbf{P} \in \mathbb{R}_+^{n \times m}$ is the transport matrix and $\mathbf{C} \in \mathbb{R}^{n \times m}$ is the cost matrix. λ is the adjust factor.

We employ OT to implant the knowledge to the sample. Formally, we first get the cost matrix $\mathbf{C}$ by computing the Euclid distance between the task-related features P_K for the knowledge and features P_S for the sample as follows:

$$\mathbf{C}_{ij} = \|P_K^i - P_S^j\|_2, \tag{16}$$

where $\mathbf{C}_{ij}$ means the transport cost from P_K^i into P_S^j, and a high relevance between P_K^i and P_S^j denotes a small cost $\mathbf{C}_{ij}$.

Further, we treat the knowledge distribution $\widehat{P}_K$ as the source and the sample distribution $\widehat{P}_S$ as the target. We then optimize the information transport matrix $\mathbf{P}$ in Eq. 15 using the Sinkhorn-Knopp algorithm (The detail is in Appendix). We further transport the task-related features entailed in the knowledge into the sample as:

$$\tilde{P}_S = P_S + \alpha \cdot \mathbf{P}^T P_K, \tag{17}$$

where α is an adjusted factor updated during the training process.

Finally, we classify the sample S with the help of implanted knowledge as:

$$[y_{tox}, y_{type}, y_{exp}, y_{tar}] = FFN(\tilde{P}_S) \tag{18}$$

4 Experiments

This section presents experimental evaluation.

4.1 Experimental Setup

Datasets. We use three public toxic datasets, i.e., *ToxicCN* (Lu et al., 2023), *Hatexplain* (Mathew et al., 2021), and *ToxiGEN*.

Baselines. We choose three types of methods as baselines.

(1) Detection tools: PERSPECTIVEAPI, PAIBERT (PAI, 2023), TOXICBERT, and UNROBERTA (Hanu and Unitary team, 2020),

(2) LLM based methods: GPT$_{\text{ZERO}}$, GPT$_{\text{COT}}$, and GPT$_{\text{UNILC}}$ (Zhang et al., 2023a),

(3) SLM based methods: TKE$_{\text{BERT/ROBERTA}}$ (Lu et al., 2023), PROMPT$_{\text{BERT/ROBERTA}}$ (He et al., 2023).

Metrics and Implementation. We employ the weighted precision (P), recall (R), and F1-score (F1) as metrics (Lu et al., 2023). We train our model on a NVIDIA A800 80GB GPU and apply the AdamW optimizer to optimize parameters [3].

4.2 Results and Analysis

4.2.1 Main Results

We evaluate our RCT method by comparing it with the baselines. To ensure the fairness, we also utilize BERT and RoBERTa as the backbone and employ the same GPT (gpt-3.5-turbo) as LLM based methods to extract the knowledge. The main comparison results on ToxicCN, Hatexplain, and ToxiGEN are shown in Table 1, 2, and 3, respectively. We have the following important observations.

[3]The details of datasets, baselines, and implementations are given in Appendix A.4, A.5, and A.6, respectively.

	Models	Toxic			Toxic type			Expression			Target		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
Tools	PerspectiveAPI	71.9	56.8	49.8	–	–	–	–	–	–	–	–	–
	PaiBERT	67.2	63.5	62.4	–	–	–	–	–	–	–	–	–
SLM	$\text{Prompt}_{\text{BERT}}$	75.1	74.7	74.7	–	–	–	–	–	–	–	–	–
	$\text{Prompt}_{\text{RoBERTa}}$	75.1	74.6	74.3	–	–	–	–	–	–	–	–	–
	TKE_{BERT}	81.2	80.9	80.8	74.9	80.2	77.3	57.6	57.7	57.3	73.9	74.8	74.2
	$\text{TKE}_{\text{RoBERTa}}$	81.9	81.8	81.8	76.6	79.1	77.8	59.3	57.6	58.1	72.5	74.0	73.0
LLM	GPT_{Zero}	74.4	74.4	74.3	–	–	–	–	–	–	–	–	–
	GPT_{CoT}	75.1	75.2	75.1	–	–	–	–	–	–	–	–	–
	$\text{GPT}_{\text{Unilc}}$	68.1	66.2	65.8	–	–	–	–	–	–	–	–	–
LLM+SLM	RCT_{BERT}	82.9	82.8	82.7$^{\wedge}$	76.7	80.6	78.5*	60.2	**60.3**	59.8*	74.8	76.3	75.5$^{\wedge}$
	$\text{RCT}_{\text{RoBERTa}}$	**83.3**	**83.2**	**83.2**$^{\wedge}$	**77.2**	**81.0**	**79.0**$^{\wedge}$	**60.5**	60.2	**60.1**$^{\wedge}$	**75.7**	**77.9**	**76.6***

Table 1: Main comparison results on ToxicCN. All results are reported by averaging five random runs. The bold and underlined scores denote the best and the second-best performance, and $*$ and $^{\wedge}$ are statistically significant marks for $p < 0.01$ and $p < 0.05$, respectively.

	Models	Toxic			Toxic type			Target		
		P	R	F1	P	R	F1	P	R	F1
Tools	PerspectiveAPI	67.4	67.6	67.5	–	–	–	–	–	–
	ToxicBERT	63.9	64.2	64.0	–	–	–	–	–	–
	UnRoBERTa	65.7	65.4	65.5	–	–	–	–	–	–
SLM	$\text{Prompt}_{\text{BERT}}$	75.5	75.4	75.4	–	–	–	–	–	–
	$\text{Prompt}_{\text{RoBERTa}}$	77.6	76.9	77.0	–	–	–	–	–	–
	TKE_{BERT}	78.3	78.4	78.2	64.0	64.3	64.0	75.6	77.1	76.2
	$\text{TKE}_{\text{RoBERTa}}$	78.7	78.8	78.6	62.6	64.8	63.6	75.2	78.8	76.9
LLM	GPT_{Zero}	73.4	73.4	73.4	–	–	–	–	–	–
	GPT_{CoT}	76.4	76.0	75.2	–	–	–	–	–	–
	$\text{GPT}_{\text{Unilc}}$	73.4	70.9	67.8	–	–	–	–	–	–
LLM+SLM	RCT_{BERT}	79.9	79.8	79.5*	**64.7**	67.5	**65.9**	**76.3**	79.1	77.5
	$\text{RCT}_{\text{RoBERTa}}$	**80.2**	**80.2**	**80.0**$^{\wedge}$	64.3	**67.8**	**65.9**$^{\wedge}$	75.8	**82.6**	**78.9***

Table 2: Main comparison results on Hatexplain. All results are reported by averaging five random runs.

Firstly, our RCT method consistently and significantly outperforms all baselines across three types of datasets. This clearly demonstrates the superiority of our method by combining SLM and LLM to leverage their strengths. Specifically, our RCT is the best on the Chinese ToxicCN and English Hatexplain datasets, showing its flexibility for different toxic language detection. Moreover, the best performance of our RCT on ToxiGEN indicates that it is good at recognizing both real-world and LLM-generated toxic contents.

Secondly, the SLM based methods are better than detection tools and LLM based methods, due to their specific task adaptability through fine-tuning. However, their performance is still worse than our RCT. Given the settings except the additional knowledge from the LLM, the reason is clear that the inferior performance of these SLM based methods must attribute to the lack of the knowledge. More evidences can be obtained from Table 5 by comparing the *'w/o tree'* variant which is better than TKE and Prompt.

Thirdly, among three LLM based methods, GPT_{Zero} outperforms the detection tools, and the performance of GPT_{CoT} is further improved by employing knowledge as rationales. At the same time, despite its careful demonstrations and problem modeling, $\text{GPT}_{\text{Unilc}}$ shows an unstable performance on three datasets. It is also worth noting that these LLM based methods achieves better performance on ToxiGEN than other two datasets. The reason might be that toxic texts in ToxiGEN are generated by inputting manually crafted prompts into LLMs, which are more familiar to LLMs and easier for these LLM based methods to recognize.

Cross dataset analysis. In order to investigate the generalization ability of fine-tuned models, we conduct the cross-dataset evaluation on Hatexplain and ToxiGEN. Note we cannot do this for ToxicCN because its language is different from those of Hatexplain and ToxiGEN, but the backbones and the fine-tuning procedure are language dependent.

	Models	Toxic			Target		
		P	R	F1	P	R	F1
Tools	PerspectiveAPI	75.4	66.8	61.1	–	–	–
	ToxicBERT	68.3	65.0	60.4	–	–	–
	UnRoBERTa	71.4	65.7	60.4	–	–	–
SLM	$\text{Prompt}_{\text{BERT}}$	72.0	72.1	71.9	–	–	–
	$\text{Prompt}_{\text{RoBERTa}}$	74.4	74.6	74.4	–	–	–
	TKE_{BERT}	80.9	80.9	80.9	75.0	74.1	74.2
	$\text{TKE}_{\text{RoBERTa}}$	81.5	81.5	81.4	74.8	74.1	74.2
LLM	GPT_{Zero}	80.9	75.7	73.5	–	–	–
	GPT_{CoT}	84.1	83.3	82.9	–	–	–
	$\text{GPT}_{\text{Unilc}}$	83.8	81.9	82.0	–	–	–
LLM+SLM	RCT_{BERT}	**87.5**	**87.5**	**87.4***	75.7	74.9	74.9
	$\text{RCT}_{\text{RoBERTa}}$	87.4	87.4	**87.4***	**76.0**	**75.6**	**75.5**$^\wedge$

Table 3: Main comparison results on ToxiGEN. All results are reported by averaging five random runs.

Models	P	R	F1
Cross: Hatexplain⇒ToxiGEN			
$\text{Prompt}_{\text{BERT}}$	60.8	61.0	57.1
$\text{Prompt}_{\text{RoBERTa}}$	63.0	59.6	50.1
TKE_{BERT}	67.0	66.0	63.5
$\text{TKE}_{\text{RoBERTa}}$	67.2	67.8	65.2
RCT_{BERT}	80.1	**78.9**	**78.1***
$\text{RCT}_{\text{RoBERTa}}$	**80.8**	78.7	77.7*
Cross: ToxiGEN⇒Hatexplain			
$\text{Prompt}_{\text{BERT}}$	63.1	63.8	63.2
$\text{Prompt}_{\text{RoBERTa}}$	61.7	62.8	61.1
TKE_{BERT}	66.8	67.1	65.4
$\text{TKE}_{\text{RoBERTa}}$	66.9	67.3	65.8
RCT_{BERT}	**74.0**	**72.4**	**70.2***
$\text{RCT}_{\text{RoBERTa}}$	**74.0**	71.4	68.4

Table 4: Cross-dataset evaluation results between Hatexplain and ToxiGEN. All results are reported by averaging five random runs.

In contrast, Hatexplain and ToxiGEN are from English, the model is applicable to both datasets for monolingual toxic detection. Hence we test models trained on ToxiGEN with samples in Hatexplain, and vice versa. The results are shown in Table 4.

We can find that our proposed RCT method performs significantly better than $\text{TKE}_{\text{BERT/RoBERTa}}$ and $\text{PROMPT}_{\text{BERT/RoBERTa}}$, showing that our method has a better generalization ability on cross datasets. Moreover, the improvements on Hatexplain ⇒ ToxiGEN are much more obvious than those on ToxiGEN ⇒ Hatexplain. For example, the F1 score of our $\text{RCT}_{\text{RoBERTa}}$ reaches 77.7, while that of $\text{Prompt}_{\text{RoBERTa}}$ is only 50.1, showing an 27.6 absolute increase. This suggests that the model trained on the real-world dataset can be well adapted to the LLM-generated dataset. The reason might be that the real-world dataset contains diversified toxic contents.

On the other hand, the spurious associations between words and labels in the dataset potentially lead to unintended biases, e.g., the term 'gay' frequently appears in toxic texts, causing the model to overlook the context of samples containing 'gay' and treat them directly as toxic (Zhang et al., 2023c; Sen et al., 2022; Garg et al., 2023). If a model relies on spurious correlations, it may perform well within the same dataset, but cannot generalize to test samples with different spurious correlations. To verify this, we compute the top 100, 500, and 1000 spurious associated words in Hatexplain and ToxiGEN datasets (Zhang et al., 2023c). The overlapping rate of spurious associated words is 0.00%, 0.60%, and 1.37%, respectively. Our method achieves the best performance in cross-dataset validation between Hatexplain and ToxiGEN, demonstrating its robustness to spurious associated words, which mitigates unintended biases.

4.2.2 Deep Analysis

Ablation analysis. To evaluate the impacts of different components in our framework, we conduct an ablation analysis and show results in Table 5.

- w/o tree: we remove the RC tree and sum sentence-level representation of samples and knowledge for transporting information.
- w/o knowledge: we remove the knowledge from LLM and utilize the RC tree to capture the multi-grained features of samples.
- w/o OT: we remove the optimal ransport (OT) and combine the multi-grained features of samples and knowledge via the simple addition.

Models	Toxic			Toxic type			Expression			Target		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
$\text{RCT}_{\text{RoBERTa}}$	**83.3**	**83.2**	**83.2**	**77.2**	81.0	**79.0**	60.5	60.2	60.1	**75.7**	77.9	**76.6**
w/o tree	82.5	82.3	82.2	76.7	80.0	78.1	**61.3**	60.3	**60.3**	75.2	75.6	75.3
w/o knowledge	83.1	82.9	82.8	76.8	81.4	79.0	58.6	**60.8**	59.4	73.4	**78.7**	75.9
w/o OT	83.3	83.2	83.1	76.4	**81.9**	79.0	59.3	60.0	59.5	74.7	77.5	75.9

Table 5: Results for ablation study. All results are reported by averaging five random runs.

Case 1	**Text:** Women should be able to choose to have an *abortion* if they wish to do so.			
	Ground Truth: Non-toxic	LLM: Non-toxic ✔	SLM: Toxic ✘	$\text{RCT}_{\text{RoBERTa}}$: Non-toxic ✔
Case 2	**Text:** People with a *mental illness* should not be allowed to work in the public sector.			
	Ground Truth: Toxic	LLM: Non-toxic ✘	SLM: Toxic ✔	$\text{RCT}_{\text{RoBERTa}}$: Toxic ✔
Case 3	**Text:** You shouldn't call it a *disability* if it's not noticeable.			
	Ground Truth: Toxic	LLM: Non-toxic ✘	SLM: Non-toxic ✘	$\text{RCT}_{\text{RoBERTa}}$: Toxic ✔

Table 6: Results for case study. The tokens in italic are paraphrased by the SLM and LLM. ✔ and ✘ denote the correct and wrong prediction, and LLM and SLM denote GPT_{CoT} and $\text{TKE}_{\text{RoBERTa}}$, respectively.

From the '*w/o tree*' ablation, we find that most results drop a lot, inferring the RC tree is crucial for capturing the local and global information in the text. The only exception is on the *expression* task with a slight increase. The reason might be due to the ignorance of syntactic structure information when we build the RC tree.

From the '*w/o knowledge*' ablation, it is clear the performance of the model drops in all tasks without the help of knowledge from the LLM, which supplies more supporting information related to the toxic text and is helpful to recognize the toxic text.

From the '*w/o OT*' ablation, we find the performance mainly decreases in the *expression* and *target* tasks. We analyze this is because optimal transport can capture the relevance of multi-grained features between samples and knowledge according to associated information distribution, which improves the transferring of multi-grained features.

In general, the performance of all degraded variants decreases on different tasks, proving the necessity and helpfulness of our designs in the proposed RCT framework.

Case analysis. We choose two toxic and one non-toxic samples from the Toxigen dataset for case study and show results in Table 6. In all three cases, we let the SLM and LLM to paraphrase the specific terms in the text by masking the corresponding terms for SLM and directly ask LLM about the terms.

In Case 1, we ask SLM to paraphrase '*abortion*' using '*What's the abortion? It is <mask>.*'. Due to limited knowledge, SLM cannot fully comprehend the text and associates '*abortion*' with '*rape*', thus finally makes wrong prediction. In contrast, LLM owns the knowledge that women have the right of '*abortion*' [4]. Consequently, both LLM and our method, which borrows the knowledge from the LLM, can predict the text as 'Non-toxic'.

In Case 2, the text states that '*people with mental illness cannot work in the public sector*'. Owing to similar samples in the training data and the fine-tuning procedure, SLM associates '*mental illness*' with '*schizophrenia*'. As a result, both SLM and our method, which uses the SLM based method as the backbone, can gain the positive cognition about *mental illness* and successfully detect the toxicity correctly. In contrast, due to the bias (Gadiraju et al., 2023; Yeh et al., 2023) against the disability, LLM assigns wrong labels for the text.

In Case 3, the text implies a toxic opinion that '*if the impairment is noticeable, it could be called disability*'. LLM ignores the toxicity due to the bias against disability. Moreover, due to the implicitness of toxicity, the SLM based method focuses on the surface positive expression '*shouldn't call it disability*' but fails to capture the hidden toxic opinion and thus also ignores the toxicity. However, our method gives the true label. We believe this is because our method has an ability to grasp the subtle meaning of the text via the human-like cognitive process and the general knowledge from the LLM. Indeed, a separate statistic shows that our model successfully correct about 22% (105/485) percent of samples for which both SLM and LLM based methods make wrong predictions.

In summary, our method implants the knowledge from the LLM into the SLM encoded sam-

[4]The detailed knowledge from the LLM are provided in Table 13 in the Appendix.

ples. Meanwhile, with the advantage of acquiring task-specific information from training data via fine-tuning, as well as the deep understanding of the text via the constructed RC tree, our method further improves SLM based methods and achieves a more superior performance in toxicity detection.

5 Conclusion

In view of the limitation of knowledge in SLM based methods and the potential bias and harmfulness in LLM based methods, we propose a novel framework to implant knowledge from LLMs into SLM based methods to leverage their strengthes for toxicity detection. Specifically, we design a reading comprehension (RC) tree to mimic the cognitive process, which gets a deep understanding of texts in the sample and knowledge and also facilitates the transportation of the knowledge. Experiments on three widely used datasets prove that our method achieves significantly better performance than existing detection tools and methods.

Limitations

We propose a method to detect the toxicity of a content, where we utilize an optimal transport technique to implant knowledge from LLMs into SLM based methods. However, the Sinkhorn-Knopp algorithm to solve the optimal transport is an iterative method, which may take more time to achieve convergence. Meanwhile, we propose a reading comprehension (RC) tree structure to get a deep understanding of the texts for recognizing the toxic content. It is worth exploring our proposed RC tree on other tasks, e.g., text classification. Finally, we separately conduct toxic detection in a single language, and the multi-linguistic toxic detection has not been studied in this paper, which can be exploited in the future.

Ethics Statement

We strictly adhere to usage agreements, and the statement in the paper is only for research purposes. We suggest to ensure reliability via re-checks after auto-detection, considering the importance of controlling toxicity.

Acknowledgments

This work was supported by a grant from the National Natural Science Foundation of China (NSFC) project (No. 62276193).

References

Abubakar Abid, Maheen Farooqi, and James Zou. 2021. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 298–306.

Raza Ali, Umar Farooq, Umair Arshad, Waseem Shahzad, and Mirza Omer Beg. 2022. Hate speech detection on twitter using transfer learning. *Computer Speech & Language*, 74:101365.

Dimosthenis Antypas and Jose Camacho-Collados. 2023. Robust hate speech detection in social media: A cross-dataset empirical evaluation. In *The 7th Workshop on Online Abuse and Harms (WOAH)*, pages 231–242.

Jianzhu Bao, Jingyi Sun, Qinglin Zhu, and Ruifeng Xu. 2022. Have my arguments been replied to? argument pair extraction as machine reading comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 29–35.

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.

Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial bias in hate speech and abusive language detection datasets. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 25–35.

Fabio Del Vigna12, Andrea Cimino23, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the first Italian conference on cybersecurity (ITASEC17)*, pages 86–95.

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.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73.

Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268.

Virginia Felkner, Ho-Chun Herbert Chang, Eugene Jang, and Jonathan May. 2023. Winoqueer: A community-in-the-loop benchmark for anti-lgbtq+ bias in large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9126–9140.

Emilio Ferrara. 2023. Should chatgpt be biased? challenges and risks of bias in large language models. *arXiv preprint arXiv:2304.03738*.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. 2021. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):1–30.

Vinitha Gadiraju, Shaun Kane, Sunipa Dev, Alex Taylor, Ding Wang, Emily Denton, and Robin Brewer. 2023. " i wouldn't say offensive but...": Disability-centered perspectives on large language models. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 205–216.

Howard N Garb. 1997. Race bias, social class bias, and gender bias in clinical judgment. *Clinical Psychology: Science and Practice*, 4(2):99.

Tanmay Garg, Sarah Masud, Tharun Suresh, and Tanmoy Chakraborty. 2023. Handling bias in toxic speech detection: A survey. *ACM Computing Surveys*, 55(13s):1–32.

Nianlong Gu, Elliott Ash, and Richard HR Hahnloser. 2021. Memsum: Extractive summarization of long documents using multi-step episodic markov decision processes. *arXiv preprint arXiv:2107.08929*.

Soumyajit Gupta, Sooyong Lee, Maria De-Arteaga, and Matthew Lease. 2023. Same same, but different: Conditional multi-task learning for demographic-specific toxicity detection. In *Proceedings of the ACM Web Conference 2023*, pages 3689–3700.

Janosch Haber, Bertie Vidgen, Matthew Chapman, Vibhor Agarwal, Roy Ka-Wei Lee, Yong Keong Yap, and Paul Röttger. 2023. Improving the detection of multilingual online attacks with rich social media data from singapore. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12705–12721.

Laura Hanu and Unitary team. 2020. Detoxify. Github. https://github.com/unitaryai/detoxify.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv preprint arXiv:2203.09509*.

Xinlei He, Savvas Zannettou, Yun Shen, and Yang Zhang. 2023. You only prompt once: On the capabilities of prompt learning on large language models to tackle toxic content. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 61–61. IEEE Computer Society.

Ruipeng Jia, Yanan Cao, Fang Fang, Yuchen Zhou, Zheng Fang, Yanbing Liu, and Shi Wang. 2021. Deep differential amplifier for extractive summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 366–376.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Prashant Kapil and Asif Ekbal. 2020. A deep neural network based multi-task learning approach to hate speech detection. *Knowledge-Based Systems*, 210:106458.

Philip A Knight. 2008. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275.

Robin Kowalski. 2018. Cyberbullying. In *The Routledge international handbook of human aggression*, pages 131–142. Routledge.

Hwaran Lee, Seokhee Hong, Joonsuk Park, Takyoung Kim, Gunhee Kim, and Jung-Woo Ha. 2023. Kosbi: A dataset for mitigating social bias risks towards safer large language model application. *arXiv preprint arXiv:2305.17701*.

Chak Leong, Yi Cheng, Jiashuo Wang, Jian Wang, and Wenjie Li. 2023. Self-detoxifying language models via toxification reversal. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4433–4449.

Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*, pages 6565–6576. PMLR.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Udara Piyasena Liyanage and Nimnaka Dilshan Ranaweera. 2023. Ethical considerations and potential risks in the deployment of large language models in diverse societal contexts. *Journal of Computational Social Dynamics*, 8(11):15–25.

Junyu Lu, Bo Xu, Xiaokun Zhang, Changrong Min, Liang Yang, and Hongfei Lin. 2023. Facilitating fine-grained detection of Chinese toxic language: Hierarchical taxonomy, resources, and benchmarks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16235–16250, Toronto, Canada. Association for Computational Linguistics.

Hongyin Luo, Shang-Wen Li, Mingye Gao, Seunghak Yu, and James Glass. 2022. Cooperative self-training of machine reading comprehension. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 244–257.

Ling Luo, Xiang Ao, Yan Song, Feiyang Pan, Min Yang, and Qing He. 2019. Reading like her: Human reading inspired extractive summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3033–3043.

Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 2023. Experiences from using code explanations generated by large language models in a web software development e-book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 931–937.

Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. 2023. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, pages 1–8.

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*, pages 14867–14875.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. *https://openai.com/blog/ chatgpt/*.

Alibaba PAI. 2023. Alibaba pai bert base chinese for llm risk detection. https://huggingface.co/alibaba-pai/pai-bert-base-zh-llm-risk-detection.

Michael Pressley and Peter Afflerbach. 2012. *Verbal protocols of reading: The nature of constructively responsive reading*. Routledge.

Vatsal Raina and Mark Gales. 2022. Answer uncertainty and unanswerability in multiple-choice machine reading comprehension. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1020–1034.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Partha Pratim Ray. 2023. Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*.

Julian Risch, Robin Ruff, and Ralf Krestel. 2020. Offensive language detection explained. In *Proceedings of the second workshop on trolling, aggression and cyberbullying*, pages 137–143.

Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A Smith, and Yejin Choi. 2020. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5477–5490.

Arif Saricoban. 2002. Reading strategies of successful readers through the three phase approach. *The Reading Matrix*, 2(3).

Indira Sen, Mattia Samory, Claudia Wagner, and Isabelle Augenstein. 2022. Counterfactually augmented data and unintended bias: The case of sexism and hate speech detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4716–4726.

Tom Shakespeare. 2013. *Disability rights and wrongs revisited*. Routledge.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Richard Sinkhorn and Paul Knopp. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348.

Robert Slonje, Peter K Smith, and Ann Frisén. 2013. The nature of cyberbullying, and strategies for prevention. *Computers in human behavior*, 29(1):26–32.

Yan Song, Yuanhe Tian, Nan Wang, and Fei Xia. 2020. Summarizing medical conversations via identifying important utterances. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 717–729.

Elif Toprak and Gamze Almacioğlu. 2009. Three reading phases and their applications in the teaching of english as a foreign language in reading classes with young learners. *Journal of language and Linguistic Studies*, 5(1).

Cédric Villani et al. 2009. *Optimal transport: old and new*, volume 338. Springer.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Kaiwen Wei, Yiran Yang, Li Jin, Xian Sun, Zequn Zhang, Jingyuan Zhang, Xiao Li, Linhao Zhang, Jintao Liu, and Guo Zhi. 2023. Guide the many-to-one assignment: Open information extraction via iou-aware optimal transport. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4971–4984.

Gary Woolley and Gary Woolley. 2011. *Reading comprehension*. Springer.

Yi Xu, Hai Zhao, and Zhuosheng Zhang. 2021. Topic-aware multi-turn dialogue modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14176–14184.

Qinyuan Ye, Xiao Huang, Elizabeth Boschee, and Xiang Ren. 2020. Teaching machine comprehension with compositional explanations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1599–1615.

Kai-Ching Yeh, Jou-An Chi, Da-Chen Lian, and Shu-Kai Hsieh. 2023. Evaluating interfaced llm bias. In *Proceedings of the 35th Conference on Computational Linguistics and Speech Processing (ROCLING 2023)*, pages 292–299.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL).

Min Zhang, Jianfeng He, Taoran Ji, and Chang-Tien Lu. 2024. Don't go to extremes: Revealing the excessive sensitivity and calibration limitations of llms in implicit hate speech detection. *arXiv preprint arXiv:2402.11406*.

Tianhua Zhang, Hongyin Luo, Yung-Sung Chuang, Wei Fang, Luc Gaitskell, Thomas Hartvigsen, Xixin Wu, Danny Fox, Helen Meng, and James Glass. 2023a. Interpretable unified language checking. *arXiv preprint arXiv:2304.03728*.

Yiming Zhang, Sravani Nanduri, Liwei Jiang, Tongshuang Wu, and Maarten Sap. 2023b. BiasX: "thinking slow" in toxic content moderation with explanations of implied social biases. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4920–4932, Singapore. Association for Computational Linguistics.

Zhehao Zhang, Jiaao Chen, and Diyi Yang. 2023c. Mitigating biases in hate speech detection from a causal perspective. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6610–6625.

Zhuosheng Zhang, Junjie Yang, and Hai Zhao. 2021. Retrospective reader for machine reading comprehension. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 14506–14514.

A Appendix

A.1 Sinkhorn-Knopp Algorithm for Optimal Transport

This section presents the definition of optimal transport and the solution, i.e., Sinkhorn-Knopp algorithm, for optimal transport.

Definition. The optimal transport (OT, Villani et al., 2009; Flamary et al., 2021; Wei et al., 2023) problem means transporting a distribution $A \in \mathbb{R}^n$ to another distribution $B \in \mathbb{R}^m$. Specifically, transporting the i-th element A_i of A to the j-th element B_j of B with transmission amount $\mathbf{P}_{\mathrm{ij}}$ and cost $\mathbf{C}_{\mathrm{ij}}$, where $\mathbf{P}_{ij}$ and $\mathbf{C}_{ij}$ are the elements of the i-th row and j-th column of transport matrix $\mathbf{P} \in \mathbb{R}_{+}^{n \times m}$ and cost matrix $\mathbf{C} \in \mathbb{R}^{n \times m}$, respectively. The OT problem minimizes the weighted cost as the optimized target to obtain the optimal transmission matrix $\mathbf{P}^*$, which is denoted as:

$$L_{WD} = \min_{\mathbf{P} \in \mathrm{U(A,B)}} \sum_{ij}^{nm} \mathbf{P}_{\mathrm{ij}} \mathbf{C}_{\mathrm{ij}}, \tag{19}$$

where L_{WD} is called Wasserstein distance and $U(A, B) = \{\mathbf{P} | \mathbf{P}\mathbf{1}_{\mathrm{m}} = \mathrm{A}, \mathbf{P}^{\mathrm{T}}\mathbf{1}_{\mathrm{n}} = \mathrm{B}\}$ is the joint distribution of distributions A and B.

Solution. It is hard to optimize L_{WD}. Sinkhorn and Knopp (1967); Knight (2008) propose an equivalent optimization target as:

$$L_{WD}^{\lambda} = \min_{\mathbf{P} \in \mathrm{U(A,B)}} \sum_{ij}^{nm} \mathbf{P}_{\mathrm{ij}} \mathbf{C}_{\mathrm{ij}} - \frac{1}{\lambda} \sum_{\mathrm{ij}}^{\mathrm{nm}} \mathbf{P}_{\mathrm{ij}} \log \mathbf{P}_{\mathrm{ij}}, \tag{20}$$

where λ is the adjustment factor. The optimal transport matrix is then computed iteratively as follows:

$$\mathbf{P}_{\mathrm{ij}}^{*} = \mathrm{u_i v_j e}^{-\lambda \mathbf{C}_{\mathrm{ij}}}, \tag{21}$$

where u_i and v_j are the i-th and j-th elements of $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$ intermediate variables for iterative optimization.

A.2 Instruction for Knowledge Extraction

As depicted in Tab. 8, we first request GPT-3.5-turbo to serve as an online content moderator with '*Role Play*' instruction, and then we instruct the 'moderator' to analyze the knowledge according to the given claim via '*Analyze Knowledge*' instruction.

Hyperparameter	Value
batch size	32
learning rate	1e-5
weight decay	1e-2
drop out	0.5
epoch	40
hidden size	768
max length	160
kernel size	3
stride	2
padding	1

Table 7: Training hyperparameter settings.

A.3 Tasks-specific Prompts, Questions and Definitions

Firstly, we introduce the tasks of *toxicity*, *toxic type*, *expression*, and *target*:

- *Toxicity* task aims to determine whether the text is toxic.
- *Toxic type* task aims to judge the type of toxic text, including 'hateful' and 'offensive' in our employed datasets.
- *Expression* task aims to classify the expressive methods when the text embodies toxicity, including 'explicit' and 'implicit' and 'paraphrase' in our employed datasets.
- *Target* task aims to identify the targeted groups of toxic text, including 'race', 'gender', 'religion' and so on.

In order to get a task-specific representation, we consider three aspects: task-specific semantics in samples, task questions, and definitions. This subsection will show the task-specific prompts, questions, and definitions.

Firstly, we obtain the task-related semantics in samples as follows:

$$prompt = [\mathrm{S}, M_{tox}, M_{type}, M_{exp}, M_{tar}], \tag{22}$$

where S is the placeholder for samples. M_{tox}, M_{type}, M_{exp} and M_{tar} mean soft prompt tokens for *toxicity*, *toxic type*, *expression* and *target* tasks, respectively. Then we obtain the specific task-related features in the samples via selecting the representation of associated soft prompt tokens.

Next, the questions and definitions are designed based on the annotation principles and definitions of the datasets, as show in Tab. 9 and 10.

	Instruction
Role Play	You are a helpful assistant for claim moderation. Please analyse the social context included in the given claim objectively. The social context may entail aspects of [GROUP1,GROUP2,...].
Analyze Knowledge	Given the CLAIM, please analyze the social context within 150 words. Note: If the claim entail [GROUP1, GROUP2,...], please analyze the neutral, negative, or positive senses on the mentioned topics.

Table 8: Instruction for getting knowledge about the content from GPT-3.5-turbo. 'GROUP1,GROUP2,...' mean the placeholder of targeted groups in the datasets.

Task	Question
Toxicity	Is the text toxic?
Toxic type	Is the text hateful or offensive?
Expression	What expressions does the text employ?
Target	If the text targets some groups or someone, which groups or someone are targeted?

Table 9: Illustration of the task-specific questions.

Task	Definition
Toxicity	Toxicity means that the text contains some rude, hateful, or rumored descriptions that make someone uncomfortable.
Toxic type	'hateful' and 'offensive' mean levels of toxicity, 'hateful' generally being more serious than 'offensive' and hostility toward certain groups.
Expression	The style of expression refers to the way toxic content in the text is described when targeting the target, including direct expression, indirect expression, and paraphrasing.
Target	Some toxic text may target some groups or individuals, including showing rudeness or bias toward marginalized groups.

Table 10: Illustration of the task-specific definitions. The definitions are based on the annotation principles and definitions of the datasets.

dataset	lang	source	#train	#test	statistics of tasks			
					#Toxic	#Toxic type	#Expression	#Target
ToxicCN	CN	social media	9600	2411	2	2	3	5
Hatexplain	EN	social media	15383	1924	2	2	–	5
ToxiGEN	EN	LLM	8960	940	2	–	–	13

Table 11: Statistics of the ToxicCN, Hatexplain, and ToxiGEN datasets. 'CN' and 'EN' denote Chinese and English, respectively.

Models	Speed(s/item)
$\text{Prompt}_{\text{BERT}}$	0.009
$\text{Prompt}_{\text{RoBERTa}}$	0.009
TKE_{BERT}	0.009
$\text{TKE}_{\text{RoBERTa}}$	0.010
RCT_{BERT}	0.142
$\text{RCT}_{\text{RoBERTa}}$	0.143

Table 12: Time cost comparison. 's/item' means spent time (seconds) detecting per sample.

A.4 Datasets

We choose three widely used toxic datasets to verify the effectiveness of our proposed method.

ToxicCN (Lu et al., 2023) is a Chinese dataset for fine-grained toxic detection, which includes hierarchical taxonomy annotations (i.e., toxicity, toxic type, expression, and targets) and collects online contents from social medias. This dataset allows us to research joint toxic Chinese recognition.

Hatexplain (Mathew et al., 2021) is an English dataset for toxic detection, which is collected from online contents. This dataset includes binary toxicity labels, ternary toxicity labels, toxic spans in text, and targets of toxic text. Following the setting of ToxicCN, we reorganize the ternary toxicity labels in Hatexplain into labels of 'toxic type' task, i.e., 'hate speech', and 'offensive'. And Hatexplain is finally reorganized to include the *toxicity*, *toxic type*, and *target* tasks.

ToxiGEN (Hartvigsen et al., 2022) is also an

Case 1	**Text**: Women should be able to choose to have an abortion if they wish to do so. **Knowledge**: "Women should be able to choose to have an abortion if they wish to do so." primarily focuses on the social context of women's reproductive rights. It addresses the ongoing debate surrounding abortion and the autonomy of women in making decisions about their own bodies. In terms of the social context of women, this claim supports the idea that women should have the right to make choices regarding their reproductive health without interference or restriction. It acknowledges the importance of women's agency and bodily autonomy, emphasizing the need for access to safe and legal abortion services. While this claim does not explicitly mention any other social contexts such as race, religion, or disability, it is important to note that the issue of abortion intersects with various identities and experiences. Different communities may have diverse perspectives on this topic, influenced by cultural, religious, or personal beliefs. Analyzing the claim within these specific social contexts would require further examination of the relevant cultural, religious, or disability-related perspectives.
Case 2	**Text**: People with a mental illness should not be allowed to work in the public sector. **Knowledge**:The social context of the claim revolves around mental illness and its impact on employment in the public sector. Mental illness is a broad term encompassing various conditions that affect a person's thinking, emotions, and behavior. The claim suggests that individuals with mental illness should be prohibited from working in the public sector, which includes government agencies, public institutions, and services. Analyzing the social context neutrally, it is important to consider the principles of equal opportunity and non-discrimination. Prohibiting individuals with mental illness from working in the public sector solely based on their condition could be seen as discriminatory and a violation of their rights. It is essential to recognize that mental illness does not necessarily hinder a person's ability to perform their job effectively. However, it is worth noting that some positions within the public sector may require specific mental health requirements due to the nature of the job.
Case 3	**Text**: You shouldn't call it a disability if it's not noticeable **Knowledge**:The claim "you shouldn't call it a disability if it's not noticeable" does not explicitly mention any specific social context. However, it touches upon the topic of disabilities and how they are perceived in society. This claim suggests that if a disability is not visible or apparent to others, it should not be labeled as a disability. In a neutral sense, this claim raises questions about the visibility and recognition of disabilities. It challenges the notion that disabilities must be physically apparent to be acknowledged and accommodated. It may prompt discussions about the experiences of individuals with invisible disabilities and the need for understanding and support. From a negative perspective, this claim could perpetuate ableism by implying that only visible disabilities are valid. It may disregard the challenges faced by individuals with invisible disabilities, such as chronic pain, mental health conditions, or cognitive impairments, who may require accommodations and support. In a positive sense, this claim could encourage a broader understanding and acceptance of disabilities that are not immediately noticeable.

Table 13: Knowledge about the cases, which is extracted from LLM.

English toxic identify dataset, and the text is generated through manually designed prompts inputted into LLM. And the dataset comprises the binary toxicity labels and targets of the generated text.

In ToxicCN, Lu et al., 2023 annotates the multiple aspects of toxic text for fine-grained detection of toxic text, including *toxicity*, *toxic type*, *expression* and *target*. Following the settings of ToxicCN, we reorganized the *Hatexplain* to include *toxicity*, *toxic type* and *target* tasks, and reorganized the *ToxiGEN* to include *toxicity* and *target* tasks. The above three datasets across different languages and sources help us extensively analyze toxicity recognition across languages and sources. In addition, we strictly abide by the licenses for using the datasets (e.g., CC BY-NC-ND 4.0) and use them only for scientific research. And personal information has been anonymized in these datasets.

A.5 Baselines

We compare our method with three types of baselines:

(1) Detection tools: Tools employed for toxicity binary detection.

- PERSPECTIVEAPI is a detection tool for toxic text from multiple languages.
- PAIBERT (PAI, 2023) is a pretrained detection tool that only can detect toxicity in Chinese text.
- TOXICBERT and UNROBERTA (Hanu and Unitary team, 2020) are pretrained detection tools only for English toxicity detection.

(2) LLM based methods: Methods based on GPT-3.5-turbo for toxicity recognition.

- GPT_{ZERO} utilizes GPT-3.5-turbo (OpenAI, 2022) to answer whether the text is toxic by a yes/no question.
- GPT_{CoT} utilizes the knowledge from GPT-3.5-turbo as the rationale and requests GPT-3.5-turbo to answer yes/no.
- $\text{GPT}_{\text{UNILC}}$ (Zhang et al., 2023a) models detection as a fairness checking problem and provides some demonstrations for LLM to recognize the toxicity.

Algorithm 1 Algorithm of our method

Require: Sample S
Ensure: $Y = [y_{tox}, y_{type}, y_{exp}, y_{tar}]$
$K \leftarrow$ ChatGPT(S)
$T_{rc,S} \leftarrow$ ConstrctTree(S)
$T_{rc,K} \leftarrow$ ConstrctTree(K)
for Q_{task} in $[Q_{tox}, Q_{type}, Q_{exp}, Q_{tar}]$ **do**
  $R_S \leftarrow$ QueryTree($Q_{task}, T_{rc,S}$)
  $R_K \leftarrow$ QueryTree($Q_{task}, T_{rc,K}$)
  $R_S \leftarrow$ OptimalTransport($R_K \rightarrow R_S$)
  $y_{task} \leftarrow$ classifier(R_S)
end for

(3) SLM methods: Methods based on fine-tuning pretrained models for detection toxic text.

- PROMPT$_{\text{BERT}}$ and PROMPT$_{\text{RoBERTa}}$ (He et al., 2023) introduce prompt tuning for toxicity binary detection based on pretrained models.
- TKE$_{\text{BERT}}$ and TKE$_{\text{RoBERTa}}$ (Lu et al., 2023) recognize multi aspects of toxic text (i.e., toxicity, toxic type, expression and targets), but fine-tune pretrained models only via '*CLS*' token without extra designs.

In addition, we strictly abide by the licenses (e.g., apache-2.0 and CC-BY-SA-4.0 license) for using the tools and use them only for scientific research.

A.6 Training Settings

We show the training hyperparameter settings in Tab. 7. The training hyperparameter settings are mainly based on (Lu et al., 2023). Specifically, the '*kernel size*', '*stride*' and '*padding*' mean the size of the convolutional blocks, scan step and padding length, respectively. In addition, we employ different pretrained models as backbones for adapting different languages (i.e., English and Chinese). For example, we use bert-base-chinese (103M params) and chinese-roberta-wwm-ext (102M params) as Chinese text encoder and apply bert-base-uncased (110M params) and roberta-base (125M params) as English text encoder. We utilize the offical open source codes for reproducing the results of Prompt and TKE. We utilize the LLM with model version GPT-3.5-turbo and employ PerspectiveAPI by calling api interface. Following the setting of TKE (Lu et al., 2023), we run all models with 5 different seeds and report the average performance. And our method takes about 3.5 hours in each run.

A.7 Time Cost Comparison

As Tab. 12 shows, we compare the time cost of our method with baselines when they classify a text. Our method takes a little more time for extra knowledge processing and the iterative solution to the optimal transport problem.

A.8 Pseudocode illustration

In order to better follow our idea, we illustrate the brief pseudocode to help reading as Algorithm 1 shows. Please reference our code for more details.