Explaining Toxic Text via Knowledge Enhanced Text Generation

Anonymous ACL submission

Abstract

Warning: This paper contains content that is offensive and may be upsetting.

Biased or toxic speech can be harmful to various demographic groups. Therefore, it is not only important for models to detect these speech, but to also output explanations of why a given text is toxic. Previous literature has mostly focused on classifying and detecting toxic speech, and existing efforts on explaining stereotypes in toxic speech mainly use standard text generation approaches, resulting in generic and repetitive explanations. Building on these prior works, we introduce a novel knowledge-informed encoder-decoder framework to utilize multiple knowledge sources to generate implications of biased text. Experiments show that our knowledge informed models outperform prior state-of-the-art models significantly, and can generate detailed explanations of stereotypes in toxic speech compared to baselines, both quantitatively and qualitatively.

1 Introduction

The toxic speech detection and classification problem has seen increasing interest in recent years. However, it is not only important for AI agents to recognize and classify toxic speech, but to also explain why it is toxic. For instance, debiasing methods that use information about toxic language may benefit from additional information given by detailed explanations of toxicity in text (Ma et al., 2020). Furthermore, detailed explanations of toxicity may facilitate human interaction with toxicity detection systems (Rosenfeld and Richardson, 2019). They can also help humans who work with toxicity classifiers use more information about the input when making decisions about toxic speech. To elucidate, consider the following offensive joke: “What type of punch do you use against a kindergartner? A sandy-hook.”. While the literal text is not toxic, the implied meaning is offensive, particularly to those affected by school shootings. An AI agent capable of generating the implied meaning could thus provide additional information to downstream actors. Note that, we use the term biased and toxic interchangeably in this work.

Existing work largely addresses the problem of detecting and classifying toxic speech (Waseem and Hovy, 2016; Founta et al., 2018; Davidson et al., 2017). As mentioned earlier, explanations of toxicity can help with downstream tasks such as debiasing or decision making by humans, thus there has been increasing demand for explainable machine learning classifiers (Ribeiro et al., 2016; Došilović et al., 2018). Recent work around explainable toxicity classification introduced Social Bias Frames (Sap et al., 2020), a formal framework which combines explanations of toxicity along with toxicity classifications along multiple dimensions. However the explanations generated from the current state-of-the-art methods tend to be generic, without much detail. For instance, explanations may focus on certain toxic components of the input but ignore others, or include irrelevant stereotypes about the minority group affected.

To fill this gap, our work proposes to leverage different types of knowledge to provide rich context and background for toxicity explanation. Specifically, we introduce a novel framework to utilize three distinct knowledge sources. Prior work (Yu et al., 2020a) divides knowledge broadly into internal and external knowledge, where internal knowledge is knowledge embedded in the input text, and external knowledge is derived from sources outside the input. Building upon these, we leverage expert knowledge that comes from high-quality expert annotations of the input, and explicit knowledge from knowledge graphs and bases, as such symbolic knowledge can provide relevant information to the output text (Yu et al., 2020a; Mou et al., 2016). While knowledge graphs and bases deterministically retrieve and restructure knowledge from raw text sources, large pretrained gen-
Figure 1: MixGEN takes 3 types of knowledge sources and outputs an implied explanation for the input post.

2 Related Work

Prior work on knowledge enhanced text generation (Yu et al., 2020b) can be viewed across two different knowledge sources.

2.1 Internal Knowledge

Internal knowledge includes knowledge that is available within the input. For instance, Latent Dirichlet Allocation (LDA) (Blei et al., 2003) can learn topics from inputs, which can then be incorporated into text generation models (Cao et al., 2015; Guo et al., 2019). Keywords may also be extracted from input text using techniques like TF-IDF, PMI or independent classifiers. In one such work, Song et al. (2019) extract emotion oriented keywords using an independent emotion classifier to enhance dialogue generation. Similarly, Mou et al. (2016) use PMI to find relevant keywords for short text conversation. Forbes et al. (2020) develop conceptual formalisms that rely on annotations about the input to generate text. In this work, we denote the use of independent annotations on input to enhance text generation as expert knowledge, as such annotations often come from human experts.

2.2 External Knowledge

Knowledge graphs and bases are commonly used as a form of external knowledge. Zhang et al. (2019) use knowledge graph embeddings to model conversation flow, while Guan et al. (2020) use triples extracted from knowledge graphs to enhance story generation. Finally, Lian et al. (2019) develop probabilistic mechanisms to select knowledge from knowledge bases for response generation. Knowledge from conventional sources are deterministically created, in that they simply restructure raw text and store them in a knowledge base or graph. We refer to this type of external knowledge as explicit knowledge. On the other hand, there has been increasing interest in the use of large pretrained generative models as a source of knowledge. Heinzerling and Inui (2021) argue that large pretrained models can in fact serve as knowledge bases, while Davison et al. (2019) argue that pretrained models can accurately assess the validity of knowledge mined from raw text. While pretrained models are also trained on raw texts, similar to knowledge bases and graphs, they draw from this knowledge probabilistically and thus are a distinct approach. We denote this type of external knowledge as implicit knowledge.

2.3 Toxic Text Understanding

Prior work around toxicity understanding mainly focuses on detection (Schmidt and Wiegand, 2017). Early approaches include using n-grams (Waseem and Hovy, 2016; Sood et al., 2012; Perera and Fernando, 2021) as well as word clustering (Xiang et al., 2012; Zhong et al., 2016). Recently, knowledge enhanced approaches have also been used for toxicity detection. For instance, Dinakar et al. (2012) use ConceptNet to detect anti-LGBT bullying. The use of meta-information, such as information about the user (Dadvar et al., 2012), has proven to be useful, depending on the type of information used. Sap et al. (2020) use Social Bias Frames to produce both toxicity classifications and explanations of toxicity. Similarly, our approach attempts to explain toxicity by leveraging different sources of knowledge to provide more context and grounding for the models to generate explanations. Different from many prior works, we synthesize these diverse knowledge sources in a unified framework to utilize the unique contribution from each...
individual knowledge source.

3 Knowledge Enhanced MixGen

This section presents our selected three different types of knowledge—expert knowledge, explicit knowledge and implicit knowledge, and our MixGen models for toxicity explanation.

3.1 Expert Knowledge

Expert knowledge is sourced from annotations of the input. For instance, in the Social Bias Frames dataset (Sap et al., 2020), such expert knowledge include human judgements towards the lewdness, offensiveness, intent to offend, and group targeted categories. This type of expert knowledge provides useful insights and heuristics for the toxicity explanation task, if they are available.

We incorporate expert knowledge into the generation process using the join embedding technique (Pryzant et al., 2020) along with toxicity classification models. The join embedding architecture uses attention weights from the toxicity classifiers to inform the text generation model about parts of the input post relevant to toxicity classification, thus providing a heuristic for the related toxicity explanation task. Formal details of the architecture can be found in Appendix A.1.

We denote these models with the naming convention, “EXPERT [FEATURE]”, where “[FEATURE]” is the categorical variable we use for the join embedding. We replace “[FEATURE]” with “ALL” when we train on all features.

3.2 Explicit Knowledge

Explicit knowledge is sourced from some knowledge base or graph. Common sources include ConceptNET, DBpedia, WikiData, etc. (Auer et al., 2007; Vrandečić and Krötzsch, 2014). We opt to use ConceptNet (Speer et al., 2017), since it contains commonsense knowledge (Liu and Singh, 2004). Commonsense knowledge incorporates everyday concepts, especially knowledge regarding social groups and situations.

Following Chang et al. (2020), given a BART model and an input, we extract ranked triples related to the input post and keep the top k triples per post where we vary the k ∈ {3, 5, 10, 15, 20, 25}. We experiment with both concatenation and attention based methods to incorporate the top k triples, but settle on a concatenation based approach due to its simplicity and the lack of performance gains from the attention based approach. Results and analysis for both the concatenation and attention based approaches are provided in Appendix 12. We denote these models with the naming convention, “EXPLICIT (k)”, where K denotes the number of triples used.

3.3 Implicit Knowledge

Implicit knowledge is obtained from some text generator, such as a large pretrained generative model. Prior work such as Heinzerling and Inui (2021), argue that large pretrained models can in fact serve as knowledge bases. Implicit knowledge grants models a probabilistic view of external raw text sources related to a given scenario or input, since generative models tend to generate based on statistical correlations found in their training corpora (Razniewski et al., 2021).

To use implicit knowledge, we first train a BART model to generate the target minority group from the input post. Following Sheng et al. (2019), we use the predicted target minority corresponding to each input post and a set of prompts to induce biased prompt completions from GPT models. We may use multiple prompts per input post, where the number of prompt completions generated is governed by a hyperparameter, k. Then we train an independent BART model to generate these biased prompt completions, given the origin input post. This BART model is then retrained to produce the implied stereotype, given the input post. Again, we provide a formal description in Appendix A.3.

We denote these models with “IMPLICIT [GPT/GPT-2] (k)”, where “GPT” and “GPT-2” correspond to the model used for prompt completion, while K corresponds to the number of biased prompt completions generated per input post.

3.4 MixGen Models

We introduce a simple and effective approach to combine all three knowledge sources as input for our MixGen family of models, and generate the final stereotype by integrating complementary knowledge from these sources. In our design, we take inspiration from Mixture of Experts models (Masoudnia and Ebrahimpour, 2012), which combine base expert model outputs into a final output using a gating mechanism. Here, we rely on attention mechanisms over the knowledge informed model outputs to serve as the gating mechanism.

We build two variants. The first variant is called MixGen CONCAT which uses concatenation to
with separator tokens and uses the concatenation as aforesaid knowledge types. We con-
pute post, and each informed by one of the produce the implied stereotype given the in-
trained models, 
ctional self attention mechanisms of the MultiView architecture to see whether the addi-
attention over input tokens, we experiment with the knowledge informed models (Chen and Yang, 
020). Since the BART model already uses self 
views to perform self attention over outputs of 
model to output the explanation. We test with six models, two from each knowledge source.

### Explicit Knowledge

- Expert knowledge sourced from knowledge bases and graphs. Restructured, deterministic interpretations of raw text sources.
- Many knowledge bases and graphs exist.
- Easy to query due to symbolic representation.
- Explainable since knowledge source is known.

### Implicit Knowledge

- External knowledge sourced from large pretrained models. Probabilistically generated from raw text sources.
- Easy to retrieval.
- Probabilistic, thus increasing diversity in retrieval.
- Low explainability.
- Low quality, since the knowledge is implicitly learned.

**Table 1:** Some pros and cons about different types of knowledge used for toxicity explanation.

![Diagram](Image)

**Figure 2:** The MIXGEN model takes in the output of multiple trained knowledge models, concatenates them with separator tokens and uses the concatenation as input to a BART model to output the explanation. We test with six models, two from each knowledge source.

combine outputs from the knowledge informed models, as shown in Figure 2. The second vari-
ant is called MIXGEN MULTI VIEW which uses views to perform self attention over outputs of 
the knowledge informed models (Chen and Yang, 2020). Since the BART model already uses self 
attention over input tokens, we experiment with the MultiView architecture to see whether the addi-
tional self attention mechanisms of the MultiView model causes changes in performance.

For MIXGEN CONCAT, suppose we have k trained models, \( M_1, \ldots, M_k \), each trained to 
produce the implied stereotype given the input post, and each informed by one of the aforementioned knowledge types. We con-
catenate the outputs of each knowledge based model, \( M_i \), to produce a new input string. Thus if each model \( M_i \) outputs \( "s[\text{OUT}_i]\)”, we get the following concatenated input string: \( "s[\text{OUT}_1][\text{SEP}]s[\text{OUT}_2]\cdots[\text{SEP}]s[\text{OUT}_k]" \). Now, let \( M \) be a standard pretrained BART model. We train model \( M \) to produce the implied stereotype using \( "s[\text{OUT}_1][\text{SEP}]s[\text{OUT}_2]\cdots[\text{SEP}]s[\text{OUT}_k]" \) as input. Note that the knowledge based models, \( M_1, \ldots, M_k \) are fixed when training \( M \). Model \( M \) serves as the final MIXGEN CONCAT model.

**MIXGEN MULTI VIEW** uses the MultiView architecture proposed by Chen and Yang (2020). In this case, the outputs of \( M_1, \ldots, M_k \) are treated as separate views. If each model \( M_i \) outputs \( "s[\text{OUT}_i]" \) given the input post, then for each model \( M_i \) we configure the corresponding view as the string \( "v_1[s[\text{OUT}_1][\text{SEP}]v_2s[\text{OUT}_2]\cdots[\text{SEP}]v_ks[\text{OUT}_k]]" \), where \( v_1, v_2, \ldots, v_k \) are view tokens. Here, \( v_1 \) is always the first token in the view string and \( v_2, \ldots, v_k \) surround \( M_i \)’s output.

We configure \( k \) such views (one for each model) and pass each into the BART MultiView model as a set of views corresponding to the original input post. The BART MultiView model is then trained to produce the corresponding output stereotype. For details on the MultiView architecture, please see Chen and Yang (2020).

### 3.5 Training

We utilize the BART encoder decoder framework throughout (Lewis et al., 2020). We use batch gradi-
ent descent when training. For a batch \( B \) with padded input sequences \( X_i \) of length \( N_s \) and corre-
sponding padded target sequences \( Y_i \) of length \( N_t \), along with knowledge \( K_i \) from some knowledge...
We compare our models with BART, and state of the art baselines from Sap et al. (2020):

- **GPT**: Following Sap et al. (2020), we train the GPT pretrained model from huggingface to generate the toxicity classifications, the Target Minority, and the Implied Stereotype as a string, when prompted with the input post.
- **GPT-2**: We train with the same setting as the GPT Baseline, but use the GPT-2 pretrained model from huggingface.
- **BART**: We train a standard pretrained BART model to generate the implied stereotype when given the input post.

### 4.3 Evaluation Metrics

We use BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004) and BERTScore (Zhang et al., 2020) to evaluate our models and take the maximum score for each hypothesis over all of the corresponding references. We use BERTScore since it looks for semantic similarity, unlike the other two metrics. We do not use human evaluation, since generated stereotypes are minimal in length compared to other text generation tasks. Moreover, we perform manual analyses of the results in Section 5.

### 4.4 Results on SBIC

Our results on both dev and test are described in Table 3. Here we focus on dev since both sets of results track similar trends. We observe that the MixGEN models outperform all other models. After MixGEN, the model using Implicit Knowledge sources perform best. These are followed by the model using Explicit Knowledge, in turn followed by model using Expert Knowledge.

Both models with explicit and with implicit knowledge outperform expert language models. The models using implicit knowledge tend to perform best overall (BLEU: from 0.650 to 0.683, ROUGE-L: from 0.624 to 0.659, BERTScore: from 0.759 to 0.800). This is likely because implicit knowledge is less structured and hence easier to induce bias from (Petroni et al., 2019). On the other hand, while our source (ConceptNET) for explicit knowledge may be biased (Mehrabi et al., 2021), retrieved stereotypes are often mixed with general, unbiased facts.

The MixGEN models outperform every other model. This makes intuitive sense since MixGEN synthesizes multiple types of knowledge. Unexpectedly, MixGEN MULTIVIEW model does not improve performance (the absolute difference is within 0.002 across all scores) over MixGEN CONCAT. This is likely due to the fact that the MultiView model was intended to capture meta-sequences in text (Chen and Yang, 2020), whereas in our setting the input is not sequential. We also note that the MixGEN models perform better than the source models, despite their input being sourced from the source models. Thus, despite differing performance, models from different knowledge sources are likely providing some distinct and complementary information.

### 4.5 Results on Implicit Hate Speech Corpus

Results on the implicit hate corpus (ElSherief et al., 2021) are given in Table 4. All models (including baselines) generally perform worse than they do on the SBIC dataset. This is likely because the implicit hate corpus contains one reference per post, in contrast to the SBIC dataset (see Table 2). While the Expert knowledge model performs worse than
We categorize the types of errors made by models and challenge type is given in Appendix 13. An additional full set of examples for each error and challenge types below are given in Table 5. We report performance of baseline models (first three rows) and our representative models from each knowledge source. A superscript of * indicates statistically significant (p-value < 0.05) improvements over the GPT and BART baselines, while a superscript of ** indicates statistically significant improvements over all baselines. We use Wilcoxon’s signed rank test with a one sided alternative hypothesis to compute p values (Wilcoxon, 1992).

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>ROUGE-L</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT</td>
<td>0.597</td>
<td>0.579</td>
<td>0.712</td>
</tr>
<tr>
<td>GPT-2</td>
<td>0.617</td>
<td>0.601</td>
<td>0.733</td>
</tr>
<tr>
<td>BART Base</td>
<td>0.495</td>
<td>0.467</td>
<td>0.624</td>
</tr>
</tbody>
</table>

Table 3: We report performance of baseline models on the implicit hate corpus (ElSherief et al., 2021). BERTS. stands for BERTScore. A superscript of * indicates statistically significant (p-value < 0.05) improvements over the BART baseline.

<table>
<thead>
<tr>
<th>Model</th>
<th>dev BLEU</th>
<th>ROUGE-L</th>
<th>BERTScore</th>
<th>test BLEU</th>
<th>ROUGE-L</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPERT (GROUP)</td>
<td>0.630*</td>
<td>0.604*</td>
<td>0.765*</td>
<td>0.637**</td>
<td>0.608*</td>
<td>0.776*</td>
</tr>
<tr>
<td>EXPLICIT (20)</td>
<td>0.650**</td>
<td>0.624**</td>
<td>0.770**</td>
<td>0.645**</td>
<td>0.617**</td>
<td>0.773**</td>
</tr>
<tr>
<td>IMPLICIT GPT-2 (15)</td>
<td>0.683**</td>
<td>0.659**</td>
<td>0.800**</td>
<td>0.689**</td>
<td>0.662**</td>
<td>0.811**</td>
</tr>
<tr>
<td>MIXGEN CONCAT</td>
<td>0.692**</td>
<td>0.665**</td>
<td>0.807**</td>
<td>0.696**</td>
<td>0.669**</td>
<td>0.817**</td>
</tr>
<tr>
<td>MIXGEN MULTIVIEW</td>
<td>0.691**</td>
<td>0.664**</td>
<td>0.806**</td>
<td>0.694**</td>
<td>0.666**</td>
<td>0.816**</td>
</tr>
</tbody>
</table>

Table 4: This table shows the performance of our models on the implicit hate corpus (ElSherief et al., 2021). BERTS. stands for BERTScore. A superscript of * indicates statistically significant (p value < 0.05) improvements over the BART baseline.

The baseline, the other models perform slightly better. This is likely because the Expert model relies on toxicity classifications, which weren’t available in the implicit hate corpus. We believe our models can be generalized to text generation tasks on other datasets, but they likely need multiple reference points where the implicit hate corpus only has one.

5 Error Analysis and Ablation Studies

We perform analyses and ablation studies on model results on the SBIC dataset. We do not perform these on the implicit hate dataset, since we have too few references per example. Examples of the error and challenge types below are given in Table 5. An additional full set of examples for each error and challenge type is given in Appendix 13.

5.1 Error Analysis

We categorize the types of errors made by models on a small sample of 200 examples from the dev set and provide the distributions in Figure 3. We provide the error categories below.

1. Non-Existent Stereotype: Model generates a stereotype when the reference stereotype is an empty string.

2. Ignores Stereotype: Model does not generate a stereotype when the reference stereotype is a non-empty string.

3. Incorrect Target Minority: Model uses the incorrect target minority.

4. Incorrect Stereotype: Model uses the correct target minority but generates an incorrect or overly general stereotype.

Figure 3: Distribution on the four error types across knowledge types and baselines.

The baseline GPT models tend to make more errors of every type, except that the expert model makes more errors of type 4. The expert model likely focuses on tokens that trigger toxicity classifications, which makes it less likely to focus on other relevant tokens. In the second example of
Figure 4: Distribution of challenges across knowledge types and baselines. We leave out cases of where models detect non-existent stereotypes from challenge type 2.

Table 5, the token “black” may be triggering the Expert Model, causing an error of type 4.

The knowledge enhanced models rarely fail to generate a stereotype (type 2 error). On the other hand, they often detect non-existent stereotypes (type 1 error), but less often than the baselines. In the third example of Table 5, “contraceptive” may be incorrectly triggering the implicit and explicit models, while the expert model is not triggered.

5.2 Challenges in Stereotype Generation

We categorize the various challenges faced by our text generation models and provide distributions in Figure 4 and counts in Appendix 15. We analyze the same sample of 200 examples from Section 5.1.

1. Misunderstands Post: The model fundamentally misunderstands the post and generates an irrelevant stereotype as a result.

2. High Sensitivity: The model is highly sensitive to trigger words, which causes it to detect non-existent stereotypes or generate stereotypes based on the triggers alone.

3. Localized Generation: The model focuses only on parts of the input and generates stereotypes based on those parts, rather than on the entire input post.

4. Does not Draw Connections: The model clearly considers the entire input post, but does not draw connections between the different parts of the post.

5. Misunderstands Sarcasm or Irony: The model takes a more literal interpretation of a sarcastic or ironic post, causing it to output text that has the opposite meaning of the reference stereotype.

6. Ignores Stereotype: The model does not generate a stereotype despite a non-empty reference stereotype.

Interestingly, the MixGEN model tends to misunderstand sarcasm and irony at a slightly higher rate than the other knowledge model types. In the fourth example of 5, MixGEN and the Implicit Model take literal interpretations of the input post. The Implicit Model type has difficulty with drawing connections over the input (challenge type 4). An example is given in the 6th row of Table 5, where the model does not draw a connection between the target minority and the stereotypes present.

5.3 How MixGENSynthesizes Knowledge

Table 5 provides examples of MixGEN synthesizing knowledge across sources. In the third example, MixGEN produces an empty string even though two of the more reliable sources (explicit and implicit knowledge) produce non-empty strings. In the fifth example, MixGEN clearly combines parts of all the knowledge sources, while in the sixth example MixGEN produces a more accurate stereotype than any of the other models. Thus, it seems that MixGEN does not simply attempt to copy the correct model, but actually possess a deeper understanding of the knowledge types it synthesizes.

5.4 Implicit Knowledge Ablation Study

Table 6 contains results of ablations on the Implicit Knowledge models. We vary the amount \((k)\) and source (GPT and GPT-2) of implicit knowledge. The model which sources knowledge using GPT only outperforms the model sourcing knowledge using GPT-2 when \(k = 3\). When \(k = 15\), the latter model sourcing GPT-2 knowledge outperforms the model sourcing knowledge from GPT. It is possible that GPT-2 is less biased than GPT, thus benefiting our model for low \(k\), but that these benefits decrease as \(k\) increases. With greater \(k\), both models have a greater chance of exhibiting bias.

5.5 MixGEN CONCAT Ablation Study

In Table 7, we look at ablations on the number of knowledge informed models, for the MixGEN CONCAT model. Let \(k\) be the number of knowledge informed models. The variant using \(k = 6\)
models performs best. Since the models variants within each knowledge type only vary by some hyperparameter, information gain probably saturates as the number of models increases; however performance decreases when $k = 9$. Since we tend to include models with lower performance for larger $k$, worse performing models are likely counter pro-

### 6 Conclusion

In this paper, we propose a novel framework MIX-GEN to generate the stereotypes present in toxic social media posts, using multiple knowledge sources. We categorized three different sources of knowledge and synthesize the sources of knowledge using the MIXGEN models. While the knowledge models perform as well as baselines, models built on different knowledge types vary in strengths and weaknesses. For instance, the expert model suffers from high sensitivity to trigger words, while the implicit models may not draw connections over complex inputs. The MIXGEN models takes this into account and minimizes the number of examples on which it has errors and/or faces challenges. We conclude that mixture and ensemble methods as simple as concatenation can leverage the complementary nature of distinct knowledge sources to produce high quality text generations.

On the other hand, the MIXGEN model requires significant computational power. One needs to train models across knowledge sources, and then train the MIXGEN model itself. Future work may alleviate this burden by considering end to end solutions, or more efficient knowledge retrieval techniques.
Ethical Considerations

Models such as the one proposed in this paper, which output toxicity classifications of text or speech and reasoning behind such classifications should be used with care. Considerations of algorithmic fairness should be taken into account (Corbett-Davies et al., 2017), as well as cultural differences (Oliva et al., 2020) and racial biases (Xia et al., 2020) which can lead to misclassifications of offensiveness. Care should be taken to avoid political bias in training datasets, when training these models for deployment purposes (Wich et al., 2020). Finally, concerns about censorship should be taken seriously (Heins, 2014).

References


Ziqiang Cao, Sujuan Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In AAAI.


Lili Mou, Yiping Song, Rui Yan, Ge Li, L. Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In COLING.


Simon Razniewski, Andrew Yates, Nora Kassner, and Gerhard Weikum. 2021. Language models as or for knowledge bases.


A Formal Details for Knowledge Incorporation

A.1 Expert Knowledge

A.1.1 Incorporating Expert Knowledge using the Join Embedding Technique

Given the BERT classifier with $m$ attention heads, an input with length $n$, and a BART model with hidden dimension $d$, we pass the input to the BERT classifier to retrieve the attention head outputs of the last layer, namely $a_1, \ldots, a_m$, each in $\mathbb{R}^n$. We also pass the input to the BART model and retrieve the BART encoded input, namely $H \in \mathbb{R}^{n \times d}$. Let $v_1, \ldots, v_m$ in $\mathbb{R}^d$ be trainable weight vectors. For the $j$-th row vector, $h_j$ of $H$, we compute an enriched hidden state $h_j'$ as follows:

$$(h_j')^T = h_j^T + \sum_{i=1}^{m} a_{ij}v_i \quad (2)$$

We then pass the enriched hidden state through the BART decoder to generate the output stereotype. To combine knowledge from multiple variables, we sum the enriched hidden state in Equation (2) over each variable.

A.2 Explicit Knowledge

A.2.1 Incorporating Explicit Knowledge using Concatenation

In this section, we provide formal details on how we incorporate explicit knowledge from ConceptNet using concatenation. In order to retrieve the top $k$ triples (varying $k \in \{3, 5, 10, 15, 20, 25\}$), we first extract nouns, verbs, and adjectives from the input as our query tokens. We then query ConceptNet for triples associated with the query tokens and sort the triples by the product of the query’s IDF weight and the triple’s edge weight.

We then translate the triples into English (Robyn Speer, 2019). For example, if the entities “car” and “vehicle” are connected by the edge relation “IsA”, the translation would be “Car is a vehicle”. We concatenate the translations to the input post to form a new input. Formally, let the input be $“s_{post}”$ and $“s_{[trp,i]}”$ be the sentence derived from the $i$-th triple. The modified input is then $“s_{post} [SEP] s_{[trp,1]} [SEP] s_{[trp,2]} \cdots [SEP] s_{[trp,k]}”$. We then pass the modified to the BART model which generates the implied stereotype.

The concatenation based approach allows the model to encode the external knowledge into its own embedding space.
A.2.2 Incorporating Explicit Knowledge using Attention

In this section, we provide formal details on how we incorporate explicit knowledge from Concept-NET using attention and the fusion layer described in Chang et al. (2020). This is an alternative method we tried in addition to the method described in Section A.2.1.

We first accumulate the top $k$ triples (varying $k \in \{5, 10, 20\}$) associated with a post, using the method described in Appendix A.2.1. We then concatenate the numberbatch embeddings (each of dimension $p$) of the two entities in each of the $k$ triples vertically to produce vectors in $\mathbb{R}^p$. We then horizontally stack the concatenations to produce a matrix, $H_{KG} \in \mathbb{R}^{k \times 2p}$. The encoded input generated by the BART model can be represented by the matrix $H_B \in \mathbb{R}^{n \times d}$, where $n$ is the input length and $d$ is the hidden size of the BART model. We compute knowledge-aware attention over the input as follows:

\[
Q = H_B \ast W_1 + b_1^T \quad (3)
\]
\[
K = H_{KG} \ast W_2 + b_2^T \quad (4)
\]
\[
V = H_{KG} \ast W_3 + b_3^T \quad (5)
\]
\[
H_{KG}^B = \text{softmax} \left( \frac{QK^T}{\sqrt{d}} \right) \quad (6)
\]

where $W_1 \in \mathbb{R}^{d \times d}, W_2, W_3 \in \mathbb{R}^{2p \times d}$ and $b_1, b_2, b_3 \in \mathbb{R}^{d}$. The knowledge-aware matrix is $H_{KG}^B \in \mathbb{R}^{n \times d}$. Finally, we concatenate the original encoded input and the knowledge-aware matrix, $H_{KG}^B$, and perform an affine transformation:

\[
H' = (H_B \oplus H_{KG}^B) \ast W_4 + b_4^T \quad (7)
\]

where $\oplus$ denotes column-wise concatenation, $W_4 \in \mathbb{R}^{2d \times d}$ and $b_4 \in \mathbb{R}^{d}$. $H' \in \mathbb{R}^{n \times d}$ is the new hidden state, which fuses the old hidden state and the new knowledge-aware hidden state. $H'$ is passed into the BART decoder.

A.3 Implicit Knowledge

A.3.1 Incorporating Implicit Knowledge using GPT

In this section, we provide formal details on incorporating implicit knowledge using a pretrained GPT model.

Let $M_{im}$ be a BART model trained to predict the target minority of a given input post. Let the target minority string predicted by $M_{im}$ be “$s_{[tm]}$” and let “$s_{[pr]}$” be a prompt. We provide a list of prompts in Table 8. We then prompt the GPT models with the following string: “The $s_{[tm]}$ $s_{[pr]}$”. The GPT models complete the prompt with a generated string, “$s_{[gpt]}$”, so that the final string is “The $s_{[tm]}$ $s_{[pr]}$ $s_{[gpt]}$”. For GPT generations, we choose hyper-parameters based on methods in Patrick von Platen (2020). For each input, we randomly select various prompts to generate $k$ sentences of the form “The $s_{[tm]}$ $s_{[pr]}$ $s_{[gpt]}$”, varying $k \in \{3, 15\}$. Note that if there is no predicted target minority, $s_{[tm]}$ is the empty string and no sentence is generated using GPT. In this case, the input is paired with just the empty string.

We then pair the input with each of the $k$ generated sentences (or the empty string if there is no predicted target minority) and train a BART model $M$ to predict the generated sentences. Model $M$ is then used as a pretrained model and is retrained to predict the implied stereotype given the same input. The retrained BART model is our final implied stereotype generator.

### Table 8: Prompts for the GPT/GPT-2 biased sentence generators.

<table>
<thead>
<tr>
<th>Prompt</th>
<th>904</th>
</tr>
</thead>
<tbody>
<tr>
<td>were known for</td>
<td>905</td>
</tr>
<tr>
<td>were described as</td>
<td>906</td>
</tr>
<tr>
<td>were regarded as</td>
<td>907</td>
</tr>
<tr>
<td>were thought of as</td>
<td>908</td>
</tr>
<tr>
<td>were well-known for</td>
<td>909</td>
</tr>
<tr>
<td>were seen as a</td>
<td>910</td>
</tr>
<tr>
<td>have a reputation for</td>
<td>911</td>
</tr>
<tr>
<td>would frequently worked as</td>
<td>912</td>
</tr>
<tr>
<td>have a job as</td>
<td>913</td>
</tr>
<tr>
<td>have a part-time job as</td>
<td>914</td>
</tr>
<tr>
<td>earned money by</td>
<td>915</td>
</tr>
<tr>
<td>started working as</td>
<td>916</td>
</tr>
<tr>
<td>have various hobbies such as</td>
<td>917</td>
</tr>
<tr>
<td>would regularly engage in</td>
<td>918</td>
</tr>
<tr>
<td>frequently talked about</td>
<td>919</td>
</tr>
<tr>
<td>behaved as though</td>
<td>920</td>
</tr>
<tr>
<td>liked to</td>
<td>921</td>
</tr>
</tbody>
</table>


### B Implementation Details

#### B.1 Implementation Details for BART Encoder Decoder Models

We train our BART models using a learning rate of $5e^{-7}$ or $6$ for 3 epochs with a batch size of 2 or 4, depending on the size of the input. This excludes the BERT classifier models, whose settings are given in Appendix B. The BART models have 406M parameters and all training is done on an Nvidia TITAN V GPU, with 12 GB memory, a boost clock speed of 1455 MHz, 640 Tensor cores and 5120 CUDA cores. Training under this regime takes approximately 90 - 120 minutes. We remove all URLs, “RT” strings, and “@” mentions from the input post. We train these models on just a single seed and results are reported on just that seed, as we had limited time to train and test our models. The baseline GPT-2 and GPT models are trained for 5 epochs, as in the original paper by Sap et al. (2020). Following the paper, we perform minimal preprocessing to the input text before training and testing and only remove all URLs. During inference, we pass batches of input from the dev and test sets to the generate method of the huggingface BART model class. We use beam search for generation, with a beam width of 10 and a length penalty of 5.0.

#### B.2 Implementation Details for BERT Classifier Models

We train base BERT models, which have 110M parameters. Training takes approximately 30 - 40 minutes on the GPUs described in B. We train these models on just a single seed, as results did not vary much as the seed varied and we had limited time to train and test our models.

<table>
<thead>
<tr>
<th>Model</th>
<th>LR</th>
<th>Batch Size</th>
<th>Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offensiveness</td>
<td>$5e^{-6}$</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>Intent to Offend</td>
<td>$5e^{-6}$</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>Lewdness</td>
<td>$5e^{-7}$</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>Group Targeted</td>
<td>$5e^{-6}$</td>
<td>32</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 9: Training Settings for BERT Classifier Models. LR stands for Learning Rate.

BERT Classifier Model training settings are given in Table 9 and Table 10.

### C Further Ablation Studies

In this section, we look at ablation studies conducted on the Expert Knowledge models and the Explicit Knowledge models.

#### C.1 Expert Knowledge Ablation Study

Recall that we use the last attention layer of a BERT classifier with the join embedding architecture introduced in Pryzant et al. (2020) to enhance the hidden states of the BART model performing stereotype generation. We perform ablation studies by replacing the classifier over a few different annotated categories, namely Offensiveness, Intent to Offend, Lewdness, and Group Targeted. We also train a model that uses all of the classifiers. The results for the ablations are in Table 11.

It is important to note the relative performance of the models. The Expert Knowledge model using the Group Targeted BERT classifiers performs better than the other single classifier models, and performs on par with the Expert Knowledge model leveraging all of the classifiers. It’s likely that the tokens used by the BERT model to identify whether a minority group is targeted aligns closely with the stereotypes. This makes intuitive sense, since the set of posts targeting some minority group likely has some stereotype mentioned in the post and vice versa. The same cannot be said for the other categories. This intuition is further strengthened by the fact that the Expert Knowledge model using all the classifiers does not perform much better than the Expert Knowledge model using just the Group Targeted classifier. Thus the other classifiers may be contributing little additional knowledge to the stereotype generation task.

#### C.2 Explicit Knowledge Ablation Study

In this section, we specifically discuss the number of knowledge triples we use when training explicit knowledge models and note trends in the BERTScore as $k$ varies. The results are in Table 12. We discuss an additional attention based model not mentioned in the main paper. The detailed methodology for this model is given in Appendix A.2.2.

We note that when concatenated directly to the input, performance increases as $k$ increases up to a point and then starts to decrease. We believe that this occurs because the usefulness of knowledge initially increases then decreases as $k$ increases. In

---

*The rest of the content is not shown as it exceeds the character limit.*
D Analysis

We provide a few examples of stereotype generation, comparing and contrasting the different model types in Table 13.

D.1 Error Analysis

Table 14 provides the counts for the Error Analysis. Clearly, the M1xGEN models minimize the total number of errors made, although M1xGEN may not have the minimal number of errors in each category.

D.2 Challenges in Stereotype Generation

Table 15 provides the counts for the Challenges faced by each model. Clearly, the M1xGEN models minimize the total number of challenges faced, although M1xGEN may not have faced the minimal number of challenges within each category.

Table 10: GPT/GPT-2/BERT Classifier Performance on Dev and Test sets.

<table>
<thead>
<tr>
<th>Models</th>
<th>dev BERTScore</th>
<th>test BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT</td>
<td>0.712</td>
<td>0.713</td>
</tr>
<tr>
<td>GPT-2</td>
<td>0.733</td>
<td>0.741</td>
</tr>
<tr>
<td>BART Base</td>
<td>0.624</td>
<td>0.638</td>
</tr>
<tr>
<td><strong>EXPERT (OFFENSIVE)</strong></td>
<td>0.759</td>
<td>0.767</td>
</tr>
<tr>
<td><strong>EXPERT (INTENT)</strong></td>
<td>0.761</td>
<td>0.764</td>
</tr>
<tr>
<td><strong>EXPERT (LEWD)</strong></td>
<td>0.757</td>
<td>0.764</td>
</tr>
<tr>
<td><strong>EXPERT (GROUP)</strong></td>
<td><strong>0.765</strong></td>
<td><strong>0.776</strong></td>
</tr>
<tr>
<td><strong>EXPERT (ALL)</strong></td>
<td>0.765</td>
<td>0.770</td>
</tr>
</tbody>
</table>

Table 11: We report BERTScores of baseline models (first three rows) and the expert knowledge models. The Expert Knowledge Models leverage annotations of categorical variables on the input to enhance stereotype generation. The **All** variant leverages all of the categorical variables.

<table>
<thead>
<tr>
<th>Model</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT</td>
<td>0.712</td>
<td>0.713</td>
</tr>
<tr>
<td>GPT-2</td>
<td>0.733</td>
<td>0.741</td>
</tr>
<tr>
<td>BART Base</td>
<td>0.624</td>
<td>0.638</td>
</tr>
<tr>
<td><strong>EXPLICIT (INPUT) (3)</strong></td>
<td>0.761</td>
<td>0.768</td>
</tr>
<tr>
<td><strong>EXPLICIT (INPUT) (5)</strong></td>
<td>0.748</td>
<td>0.757</td>
</tr>
<tr>
<td><strong>EXPLICIT (INPUT) (10)</strong></td>
<td>0.764</td>
<td>0.770</td>
</tr>
<tr>
<td><strong>EXPLICIT (INPUT) (15)</strong></td>
<td>0.709</td>
<td>0.772</td>
</tr>
<tr>
<td><strong>EXPLICIT (INPUT) (20)</strong></td>
<td>0.770</td>
<td>0.773</td>
</tr>
<tr>
<td><strong>EXPLICIT (INPUT) (25)</strong></td>
<td>0.752</td>
<td>0.763</td>
</tr>
<tr>
<td><strong>EXPLICIT (ATTN) (5)</strong></td>
<td>0.761</td>
<td>0.768</td>
</tr>
<tr>
<td><strong>EXPLICIT (ATTN) (10)</strong></td>
<td>0.759</td>
<td>0.765</td>
</tr>
<tr>
<td><strong>EXPLICIT (ATTN) (20)</strong></td>
<td>0.768</td>
<td>0.769</td>
</tr>
</tbody>
</table>

Table 12: We report BERTScores of baseline models (first three rows) and the explicit knowledge models.

particular, as \( k \) increases, many of the latter triples tend to be synonymous with earlier triples or unrelated to the original input. We also note that incorporating knowledge as attention tends to produce a better BERTScore, while not performing as well on other metrics. We suspect that this is due to the numberbatch embeddings capturing more semantic meaning than the BART embeddings. The numberbatch embeddings draw from a variety of sources, in addition to the graph structure itself and they perform well on benchmarks measuring word similarity.
<table>
<thead>
<tr>
<th>Error/Challenge Type</th>
<th>Input Post</th>
<th>Generated Stereotype</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Type 1 (Non-Existence Stereotype)</td>
<td>what do you call it when you fuck an apple? an inside job</td>
<td>makes light of child abuse</td>
<td>The post does not refer or imply anything about children in any way.</td>
</tr>
<tr>
<td>Error Type 2 (Ignores Stereotype)</td>
<td>why do bitches hate so much tho? what’s wrong with congratulating and giving people props sometimes?</td>
<td>“”</td>
<td>The post clearly refers to women in a derogatory manner.</td>
</tr>
<tr>
<td>Error Type 3 (Incorrect Target Minority)</td>
<td>if harvey and irma are god’s fidget spinners, wouldn’t god be autistic?</td>
<td>trivializes harm to victims</td>
<td>The post refers to autistic folks, but the generate stereotype doesn’t capture it.</td>
</tr>
<tr>
<td>Error Type 4 (Incorrect Stereotype)</td>
<td>why don’t black people go on cruises? because they already fell for that shit once.</td>
<td>black people deserve to die</td>
<td>The post refers to African American history (those arriving on slave ships), yet the generated stereotype does not mention this.</td>
</tr>
<tr>
<td>Challenge Type 1 (Misunderstands Post)</td>
<td>if the operation to remove your tonsils is called a tonsillectomy, then what is the name of the operation for a female do male sex change? addadictomy</td>
<td>women are only good for sex</td>
<td>The generate stereotype has nothing to do with the original post.</td>
</tr>
<tr>
<td>Challenge Type 2 (High Sensitivity)</td>
<td>why do jews have big noses? to smell the money.</td>
<td>jews were killed in the holocaust</td>
<td>The token “jew” triggers an irrelevant generation.</td>
</tr>
<tr>
<td>Challenge Type 3 (Localized Generation)</td>
<td>how do you know the asian food is undercooked? when it starts barking.</td>
<td>asians have no food</td>
<td>The post focuses on the first part of the sentence, but ignores the remainder.</td>
</tr>
<tr>
<td>Challenge Type 4 (Does not Draw Connections)</td>
<td>enjoy having multiculturalism shoved down your throat, fighting for israel and having a black or asian as your new best friend.</td>
<td>non-whites are inferior</td>
<td>The generated stereotype understands that the stereotype is about non-whites, but does not connect it to the broader stereotype of non-white cultures having a negative impact.</td>
</tr>
<tr>
<td>Challenge Type 5 (Misunderstands Sarcasm/Irony)</td>
<td>dating rules for women * don’t be morbidly obese *</td>
<td>women shouldn’t be skinny</td>
<td>The generated stereotype inverts the sarcasm of the original post and thus fails to capture the stereotype.</td>
</tr>
<tr>
<td>Challenge Type 6 (Ignores Stereotype)</td>
<td>fight anyone that tells you there’s nothing wrong with being fat, and help your body and bones and lose weight, seriously.</td>
<td>“”</td>
<td>The input post targets those who are obese, yet the model does not generate a stereotype.</td>
</tr>
</tbody>
</table>

Table 13: Examples of generation from each of the error types described in Sections 5.1 and 5.2

<table>
<thead>
<tr>
<th>Error Type</th>
<th>GPT</th>
<th>GPT-2</th>
<th>Expert</th>
<th>Explicit</th>
<th>Implicit</th>
<th>MIXGEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>46</td>
<td>37</td>
<td>44</td>
<td>30</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>37</td>
<td>41</td>
<td>24</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>99</td>
<td>110</td>
<td>122</td>
<td>130</td>
<td>137</td>
</tr>
</tbody>
</table>

Table 14: This table contains the counts for the error analysis. Though it is not quite an error and isn’t mentioned in the main paper, Error Type 5 accounts for the examples in which the model outputs an accurate stereotype.
Table 15: This table contains the counts for the error analysis. Though it is not quite a challenge and isn’t mentioned in the main paper, Challenge Type 7 accounts for accurate generations by the model. Challenge Type 2 is partitioned into two sub-categories, one which includes only cases where the model detects non-existent stereotypes and another which does include such cases.