DialoCONAN Counterspeech Generation Challenge
Prof. Tanmoy Chakraborty - ELL884

Project3 : C.Tech-Z
Om Prakash - 2023EET2177
Animesh Lohar - 2024EET2368

May 10, 2025

Contents

1 Introduction

The proliferation of offensive content on social media platforms necessitates automated systems capable of counterspeech generation to promote positive interactions and reduce hate speech. Recent advances in transformer-based architectures, such as BART [2], have enabled the development of generative models that can produce contextually relevant responses.

This report documents the process, challenges, and performance evaluation of fine-tuning such a model on a dataset derived from dialogue annotations for counterspeech. We analyze training logs, evaluate metrics, and provide mathematical insights into the evaluation strategies and observed issues.

2 Dataset and Preprocessing

2.1 Dataset Description

The dataset, sourced from `DIALOCONAN.csv`, contains dialogue annotations with fields such as `dialogue_id`, `turn_id`, `type`, and `text`. The critical class of interest is `type='CN'`, indicating counterspeech turns.

2.2 Preprocessing Pipeline

The preprocessing involves grouping dialogue turns by `dialogue_id`, sorting by `turn_id`, and constructing context-response pairs. Formally, for each dialogue $D = \{(t_i)\}$, where t_i has attributes $(\text{type}_i, \text{text}_i, \text{turn\_id}_i)$, the input X_j and output Y_j for a counterspeech turn t_j are:

$$X_j = \bigcup_{i<j} \text{text}_i, \quad \text{and} \quad Y_j = \text{text}_j \tag{1}$$

This creates a sequence-to-sequence learning problem, where the model learns to generate Y_j conditioned on X_j.

2.3 Implementation Snippet

```
def preprocess_data(file_path):
    df = pd.read_csv(file_path)
    dialogues = df.groupby('dialogue_id')
    inputs, outputs = [], []
    for dialogue_id, group in dialogues:
        turns = group.sort_values('turn_id')
        for _, row in turns.iterrows():
            if row['type'] == 'CN':
                context = turns[turns['turn_id'] < row['turn_id']]
                input_text = "\n".join(context['text'].tolist())
                inputs.append(input_text)
                outputs.append(row['text'])
    return pd.DataFrame({'input': inputs, 'output': outputs})
```

3 Model Architecture and Training

3.1 Model Choice

We utilize **BART** [2] due to its seq2seq capabilities, pretraining on corrupted text, and strong transfer learning performance. The architecture comprises an encoder $E(\cdot)$ and decoder $D(\cdot)$, with the generation probability:

$$P_\theta(Y|X) = \prod_{t=1}^{T} P_\theta(y_t|y_{<t}, X) \tag{2}$$

where y_t is the token at position t in the output sequence.

3.2 Training Hyperparameters

- Learning rate $\eta = 2 \times 10^{-5}$ - Batch size $B = 1$ with gradient accumulation over 8 steps - Epochs $N = 3$ - Beam search with num_beams $= 4$

3.3 Loss Function

The negative log-likelihood (NLL) loss:

$$\mathcal{L}(\theta) = -\sum_{(X,Y)} \log P_\theta(Y|X) \tag{3}$$

which is minimized during training via stochastic gradient descent.

4 Training Logs and Observations

4.1 Evaluation Metrics Over Epochs

The training logs are summarized as follows:

Table 1: Training and Evaluation Metrics

Epoch	Eval Loss	BLEU	ROUGE-L	BERTScore F1	Runtime (s)
1	5.757	0.0	0.0129	-0.525	104.7
2	4.001	0.0	0.0116	-1.041	19.8
3	3.673	0.0	0.0206	-0.526	16.0

4.2 Analysis of Metrics

The key observations:

- **BLEU** scores are consistently at zero, indicating negligible n-gram overlap between generated and reference responses.

- **ROUGE-L** scores are very low (1-2%), suggesting minimal subsequence overlap.
- **BERTScore F1** is negative, reflecting poor semantic similarity or empty outputs.
- High **eval_loss** indicates poor model calibration or trivial outputs.

4.3 Generated Outputs

Sample logs reveal warnings:

```
Warning: Empty candidate sentence detected; setting raw BERTscores to 0.
```

This indicates the model often produces empty or invalid responses, severely impacting evaluation scores.

5 Mathematical Insights into Evaluation Metrics

5.1 BLEU Score

Given candidate C and reference R sequences, BLEU computes an n-gram precision:

$$p_n = \frac{\sum_{n\text{-grams} \in C} \text{Count}_{clip}(n\text{-grams})}{\sum_{n\text{-grams} \in C} \text{Count}(n\text{-grams})} \tag{4}$$

The BLEU score aggregates over N n-gram precisions with weights w_n:

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{5}$$

where BP is the brevity penalty:

$$\text{BP} = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c \leq r \end{cases} \tag{6}$$

with c = length of candidate, r = length of reference.

5.2 ROUGE-L

ROUGE-L is based on the Longest Common Subsequence (LCS):

$$\text{LCS}(X, Y) = \max_{subsequence} \{\text{length of longest common subsequence}\} \tag{7}$$

The ROUGE-L score:

$$\text{ROUGE-L} = \frac{(1+\beta^2) \times \text{LCS}}{\text{len}(Y)} + \frac{(1+\beta^2) \times \text{LCS}}{\text{len}(Y')} \tag{8}$$

where β balances precision and recall.

5.3 BERTScore

BERTScore assesses semantic similarity via contextual embeddings $\mathbf{E}$:

$$\text{BERTScore} = \frac{1}{|Y|} \sum_{i=1}^{|Y|} \max_j \text{sim}(\mathbf{E}_{Y_i}, \mathbf{E}_{Y'_j}) \tag{9}$$

with cosine similarity:

$$\text{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \tag{10}$$

Negative or near-zero values indicate poor semantic overlap.

6 Generation Strategies and Challenges

6.1 Generation Function

The model generates responses using beam search:

$$\hat{Y} = \text{generate}(X) = \arg\max_Y P_\theta(Y|X), \quad \text{with } |Y| \leq 128 \tag{11}$$

6.2 Observed Failures

The main issues involve:

- **Empty responses**: leading to zero BERT scores.
- **Overly generic outputs**: due to insufficient training.
- **Limited diversity**: caused by small beam size and lack of sampling.

6.3 Mathematical Considerations

To improve diversity and avoid empty responses, techniques like nucleus sampling [3] can be employed, where the token probability distribution is truncated at a cumulative probability p:

$$\mathcal{V}_p = \left\{ y : \sum_{y' \leq y} P(y') \leq p \right\} \tag{12}$$

This encourages sampling from high-probability tokens while maintaining diversity.

7 Discussion and Recommendations

Based on the analysis, the following strategies can be adopted:

- **Improve Data Quality**: Ensure responses are meaningful and diverse.
- **Prompt Engineering**: Use explicit prompts like "Dialogue:" and "Counter:" to condition the model.
- **Training Duration**: Extend epochs and monitor validation metrics.
- **Decoding Strategies**: Use larger beam sizes, nucleus sampling, or temperature adjustments.
- **Evaluation**: Incorporate manual inspection and human-in-the-loop validation.

8 Conclusion

The current experimental results reveal significant challenges in generating meaningful counterspeech responses, as evidenced by evaluation metrics and sample outputs. The root causes are multifaceted, involving data, model training, and decoding strategies. Addressing these issues requires a holistic approach combining data augmentation, prompt design, training refinement, and advanced decoding.

Future work should focus on enhancing data quality, employing more sophisticated decoding techniques, and integrating human feedback to iteratively improve model responses.

9 References

References

[1] Helena Bonaldi1, Sara Dellantonio2, Serra Sinem Tekiroglu, Marco Guerini3. (2022). *DIALOCONAN : Human-Machine Collaboration Approaches to Build a Dialogue Dataset for Hate Speech Countering*

[2] Lewis, M., Liu, Y., Goyal, N., et al. (2019). *Bart: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.* arXiv preprint arXiv:1910.13461.

[3] Holtzman, A., Buys, J., Du, L., et al. (2019). The Curious Case of Neural Text Degeneration. *arXiv preprint arXiv:1904.09751.*

[4] Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics.*

[5] Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. *Text Summarization Branches Out.*

[6] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). BERTScore: Evaluating Text Generation with BERT. *International Conference on Learning Representations*.