EUREKA: EUphemism Recognition Enhanced Through KNN-based Methods and Augmentation

Sedrick Scott Keh[* 1]**, Rohit Bharadwaj**[* 2]**, Emmy Liu**[† 1]**,**
Simone Tedeschi[† 3,4]**, Varun Gangal** [1]**, Roberto Navigli** [3]

[1]Carnegie Mellon University, [2]Mohamed bin Zayed University of Artificial Intelligence,
[3]Sapienza University of Rome, [4]Babelscape, Italy
{skeh,mengyan3,vgangal}@cs.cmu.edu, rohit.bharadwaj@mbzuai.ac.ae
{tedeschi,navigli}@diag.uniroma1.it

Abstract

We introduce EUREKA, an ensemble-based approach for performing automatic euphemism detection. We (1) identify and correct potentially mislabelled rows in the dataset, (2) curate an expanded corpus called *EuphAug*, (3) leverage model representations of Potentially Euphemistic Terms (PETs), and (4) explore using representations of semantically close sentences to aid in classification. Using our augmented dataset and kNN-based methods, EUREKA[1] was able to achieve state-of-the-art results on the public leaderboard of the Euphemism Detection Shared Task, ranking first with a macro F1 score of 0.881.

1 Introduction

Euphemisms are mild or indirect expressions used in place of harsher or more direct ones. In everyday speech, euphemisms function as a means to politely discuss taboo or sensitive topics (Danescu-Niculescu-Mizil et al., 2013), to downplay certain situations (Karam, 2011), or to mask intent (Magu and Luo, 2018). The Euphemism Detection task is a key stepping stone to developing natural language systems that are able to process (Tedeschi et al., 2022; Liu et al., 2022; Jhamtani et al., 2021) and generate non-literal texts.

In this paper, we detail our methods to the Euphemism Detection Shared Task at the EMNLP 2022 FigLang Workshop[2]. We achieve performance improvements on two fronts:

1. **Data** – We explore various data cleaning and data augmentation (Shorten and Khoshgoftaar, 2019; Feng et al., 2021; Dhole et al., 2021) strategies. We identify and correct potentially mislabelled rows, and we curate a new dataset called

EuphAug by extracting sentences from a large unlabelled corpus using semantic representations of the sentences or euphemistic terms in the initial training corpus.

2. **Modelling** – We explore various representational and design choices, such as leveraging the LM representations of the tokens for euphemistic expressions (rather than the `[CLS]` token) and incorporating sentential context through kNN augmentation and deep averaging networks.

Using these methods, we develop a system called EUREKA which achieves a macro F1 score of 0.881 on the public leaderboard and ranks first among all submissions. We found the data innovations to be more significant in our case, indicating that euphemistic terms can be classified with some accuracy if potentially euphemistic spans are identified earlier in the pipeline.

2 Task Settings and Dataset

2.1 Task Settings

The task and dataset are specified by the Euphemism Detection Shared Task, which uses a subset of the euphemism detection dataset of Gavidia et al. (2022). The goal of the task is to classify a Potentially Euphemistic Term (PET) enclosed within delimiter tokens as either literal or euphemistic in that context. The training set contained 207 unique PETs and 1571 samples, of which 1106 are classified as euphemisms.

2.2 Data Cleaning

Gavidia et al. (2022) characterize common sources of ambiguity and disagreement among annotators. However, while exploring the data, we also spotted some rows which were, beyond a reasonable doubt, mislabelled (Table 1). This is an artifact of many human-annotated datasets (Frenay and Verleysen, 2014) and is a potential source of noise that could negatively affect performance (Nazari et al., 2018).

Sentence Containing PET	Sense (Euph.)	Sense (Non-Euph.)	Label (Original)	Label (Corrected)
Does your software collect any information about me, my listening or my surfing habits? Can it be `<disabled>`?	Handicapped	Switched off	1	0
Europe developed rapidly [...] Effective and `<economical>` movement of goods was no longer a maritime monopoly.	Prudent or frugal	Related to the economy	0	1
The Lancers continued to hang on to the `<slim>` one-point line as Golden West started a possession following [...]	Thin (physical appearance)	Thin (non-physical)	1	0

Table 1: Examples of incorrectly labelled sentences identified by our data cleaning pipeline. The label is 1 if the term is used euphemistically, 0 otherwise.

Motivated by this, we design a data cleaning pipeline to quickly identify and correct such errors (Figure 1). Since the goal is simply to correct as many errors as possible (rather than to be perfectly accurate), we take a few heuristic liberties in our design choices. First, to maximize yield and avoid dealing with less impactful PETs, we filter out PETs which appear <10 times or are classified as positive/negative >80% of the time. This leaves us with 33 PETs. We then manually curate a sense inventory (euphemistic vs. non-euphemistic senses) using context clues and BabelNet definitions (Navigli and Ponzetto, 2012, v5.0). To ensure the quality of the sense inventory, we have multiple members of our team look through the assigned euphemistic and non-euphemistic senses and verify their appropriateness. Next, for each sentence, we replace the PET with its euphemistic meaning and calculate the BERTScores (Zhang* et al., 2020) between the initial sentences and PET-replaced sentences. Replacing euphemistic PETs should not change the semantics drastically and hence should result in a high BERTScore, while replacing non-euphemistic PETs would lead to a low BERTScore. To identify potentially misclassified sentences, we therefore look for positively-classified sentences with low BERTScores or negatively-classified sentences with high BERTScores. We heuristically set this threshold at the halfway mark: if a sentence is among the top half of BERTScores and has a negative label (or among the bottom half and has a positive label), then we flag it as "potentially mislabelled". We end up with 203 potentially mislabelled sentences.

Once these potentially mislabelled sentences have been identified, we go through them manually and correct the ones which we identify as incorrectly labelled, such as the ones in Table 1. In cases where we are unsure of what the label should be (e.g. ambiguous cases as mentioned in Gavidia et al. (2022)), we leave the original label. As was done with the sense inventories, multiple members of our team then verify that the corrections made are appropriate. Although this still involves some human labor, it is much more tractable as compared to having to go through the entire dataset. Out of the 203 potentially mislabelled rows, we modify the labels of 25 of them.

2.3 *EuphAug* Corpus

In addition to data cleaning, we also use data augmentation techniques to gather an extended corpus, which we call *EuphAug*. We explore two variants of *EuphAug*, as outlined below:

1. **Representation-Based Augmentation** – We search in an external corpus for additional sentences in which specific PETs appear, then assign a label to these PETs based on their vector representations. We call this procedure *EuphAug-R*.

Let our training set (provided by task organizers) be S. Consider a PET p, which appears in sentences $s_1, s_2, \ldots s_k \in S$, with corresponding labels $l_{s_1}, l_{s_2}, \ldots l_{s_k} \in \{0, 1\}$. We search in an external corpus C (i.e., WikiText) for n sentences $c_1, \ldots, c_n$ containing the PET p. Finally, for each sentence $c_1, \ldots, c_n$ we assign label l_{c_j} as follows:

Algorithm 1 EuphAug-R

Task: Given sentence c_j containing PET p, assign l_{c_j}.
for $s_i \in \{s_1, s_2, \ldots s_k\}$ **do**
    Find $\text{dist}_i = \text{dist}(s_i, c_j)$
Find $M = \arg\max\{\text{dist}_1, \text{dist}_2, \ldots, \text{dist}_k\}$.
Find $m = \arg\min\{\text{dist}_1, \text{dist}_2, \ldots, \text{dist}_k\}$.
if $\text{dist}_M \geq \delta \wedge (|\text{dist}_M - \delta| > |\text{dist}_m - \epsilon|)$ **then**
    Add c_j to augmented corpus with label $l_{c_j} = l_{s_M}$
else if $\text{dist}_m \leq \epsilon \wedge (|\text{dist}_m - \epsilon| > |\text{dist}_M - \delta|)$ **then**
    Add c_j to augmented corpus with label $l_{c_j} = 1 - l_{s_M}$
else
    Do not add c_j to augmented corpus.
end if

---

where δ and ϵ are manually-tuned thresholds, and $\text{dist}(a, b)$ represents the cosine distance between the sentential embeddings [3] of a and b. In other

[3] https://www.sbert.net/

Step 1: Select Potentially Euphemistic Term (PET)

I've stopped smoking **<weed>** for a week now.

I hope you can **<weed>** through the confusion and find peace

It is frustrating to try to **<weed>** out what is happening

He made money to buy some beer and **<weed>**.

The **<weed>** had deep roots.

Step 2: Replace with euphemistic meaning and get BERTScore

BERTScore(I've stopped smoking **<weed>** for a week now, I've stopped smoking **<marijuana>** for a week now) = **0.99**

BERTScore(I hope you can **<weed>** through the confusion and find peace, I hope you can **<marijuana>** through the confusion and find peace) = **0.70**

BERTScore(It is very frustrating to try to **<weed>** out what is happening, It is very frustrating to try to **<marijuana>** out what is happening) = **0.65**

BERTScore(He made money to buy some beer and **<weed>**, He made money to buy some beer and **<marijuana>**) = **0.98**

BERTScore(The **<weed>** had deep roots, The **<marijuana>** had deep roots) = **0.63**

Step 3: Rerank and identify potentially mislabelled sentences

R E R A N K I N G

0.99 – I've stopped smoking **<weed>** for over a week now.

0.98 – He made money to buy some beer and **<weed>**.

0.70 – I hope you can **<weed>** through the confusion and find peace.

0.65 – It's frustrating to try to **<weed>** out what is happening.

0.63 – The **<weed>** had deep roots.
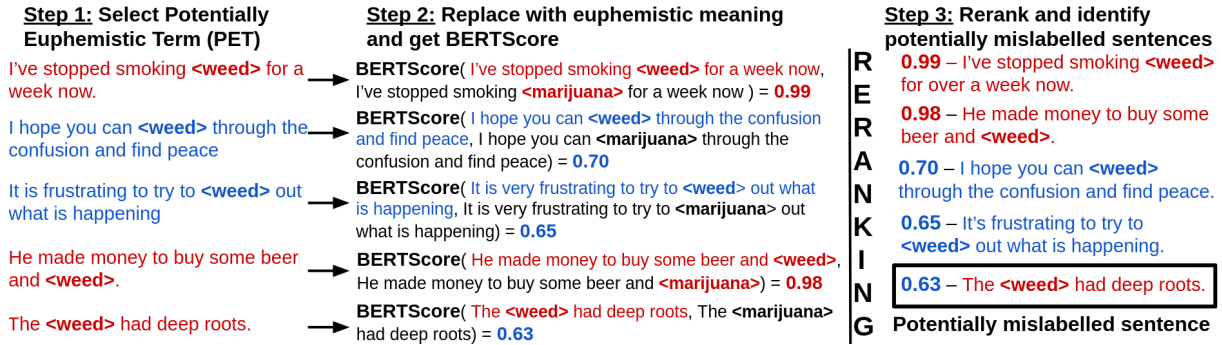
Potentially mislabelled sentence

Figure 1: Example of our data cleaning pipeline to automatically identify potentially mislabelled sentences. Red indicates positively classified sentences and blue indicates negatively classified sentences.

words, we augment our corpus with a sentence c_j only if it is sufficiently similar to, or sufficiently different from, all sentences containing the PET in S. We set $n = 20$ as the maximum number of sentences extracted from C for a PET p, and obtain a corpus of around 4700 additional examples.

2. **Sense-Based Augmentation** – While *EuphAug-R* aims to augment the dataset by finding existing sentences which already contain the PETs, this sense-based approach, instead, considers sentences which contain the senses of the PETs. This is done using the sense inventories created in Section 2.2 and searching the WikiText corpus. For instance, to find new sentences containing "disabled", we do not search directly for appearances of "disabled". Rather, we search for instances of "handicapped" and replace these occurrences with "disabled" to obtain our positive examples. We then search for instances of "switched off" and replace those occurrences with "disabled" to obtain our negative examples. We call this expanded corpus *EuphAug-S*. We sample at most 20 new sentences for each sense (if there are less than 20 occurrences in Wiki-Text, we take all of those present). In addition, some words have senses which cannot be summarized concisely in a single expression (e.g. "slim" in Table 1), so we drop these from our search terms. The final *EuphAug-S* contains 950 rows.

3 Methodology

For our baseline model, we use a pretrained RoBERTa-large model (Liu et al., 2019). For evaluation, we use the macro F1-score, as specified by the shared task description.

3.1 PET Embeddings

We leverage the embeddings of PET expressions. While models usually perform classification by passing the `[CLS]` token embedding to a final classifier layer, we instead pass the embeddings of PET tokens. If there are multiple tokens within the PET, we take the sum of these tokens. We hypothesize that `[CLS]` embeddings lose out on the discriminatory power due to pooling of all the embeddings in a sentence, and that using the PET embeddings as signals can better allow the model to focus specifically on the PET senses.

3.2 Making use of context

Additionally, we explore using context outside of the PET embeddings. Intuitively, euphemistic and non-euphemistic terms tend to be used in slightly different contexts, with euphemistic terms often being used to discuss sensitive topics. We experiment with two ways to make use of this additional context, as detailed below.

3.2.1 kNN Augmentation

Inspired by work on retrieval-based language models (Alon et al., 2022; Khandelwal et al., 2019), we augment the baseline model with a kNN store of the training set, and interpolate the classification probabilities of the base model and a kNN-based model. We follow the usual setup for such a model, with the exception that y is a binary variable indicating euphemistic/non-euphemistic rather than a token from the vocabulary.

In Equation 1, $\mathcal{N}$ refers to the 5 closest neighbours to x in the training set retrieved through cosine similarity with the `[CLS]` token generated by RoBERTa, or $f(x)$. (k_i, v_i) refers to the key and value, in this case `[CLS]` tokens for other sentences, and a binary variable, respectively. In Equa-

Feature Tested	Model	Dataset	P	R	F1
-	RoBERTa-large	Original	0.8756	0.8168	0.8399
1) Data Cleaning	RoBERTa-large	Cleaned	0.8617	0.8300	0.8435
2) Data Augmentation	RoBERTa-large	Original+*EuphAug-R*	0.8529	0.8388	0.8452
	RoBERTa-large	Original+*EuphAug-S*	0.8728	0.8306	0.8481
3) PET Embedding	RoBERTa-large+PET	Original	0.8694	0.8408	0.8533
4) Additional Context	RoBERTa-large+KNN	Original	0.8769	0.8210	0.8411
	RoBERTa-large+DAN	Original	0.8481	0.7983	0.8181
Final Models	RoBERTa-large+PET	Cleaned	0.8728	0.8471	0.8582
	RoBERTa-large+PET	Cleaned+*EuphAug-S*	0.8692	0.8584	0.8633
	RoBERTa-large+PET+KNN	Cleaned	0.8792	0.8517	0.8635
Final Ensemble	Model 1 + Model 2 + Model 3	-	**0.8994**	**0.8788**	**0.8884**

Table 2: We independently test 4 features. The final models leverage one or more of these features, and the final ensemble combines the 3 final models. Results are averaged over 10 random seeds. For the final ensemble, since we are just interested in the best possible model, we pick the best random seeds from each model instead of averaging. The three best seeds have F1-scores of 0.8734, 0.8864, and 0.8842, which is slightly improved by our ensembling.
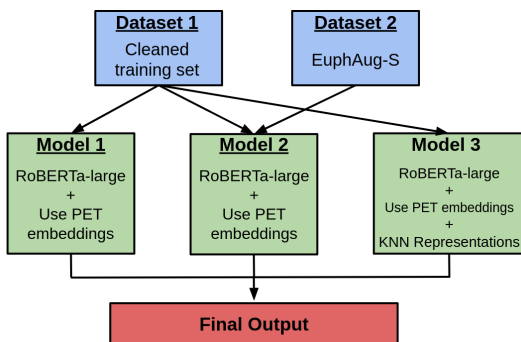

Figure 2: Models and datasets used in the ensemble.

tion 2, this value is combined with the probabilities from the base PET model.

$$p_{\text{kNN}}(y|x) \propto \sum_{(k_i,v_i)\in\mathcal{N}} \mathbb{1}(y_i{=}v_i)\exp\left(-d(k_i,f(x))\right) \quad (1)$$

$$p(y|x){=}\lambda p_{kNN}(y|x){+}(1{-}\lambda)p_{PET}(y|x) \quad (2)$$

3.2.2 Deep Averaging Network

Additionally, we experiment with a Deep Averaging Network (DAN) over embeddings for all the tokens of the sentence (Iyyer et al., 2015). For a sentence with tokens $x_1, ..., x_N$, we take the mean vector for the entire sentence: $z = \frac{1}{N}\sum_{i=1}^{N} x_i$. We then pass the mean vector through a linear layer with dropout before a second linear layer which outputs to $\mathbb{R}^2$. Note that unlike the original DAN, we do not drop out tokens, as this was found to hurt performance in preliminary experiments.

3.3 Ensembling

Our final model consists of an ensemble of 3 different models, as shown in Figure 2 and Table 2. For this ensemble, we simply consider a majority vote of the outputs of the 3 models.

4 Experiments and Results

4.1 Implementation Settings

We split our dataset into train-validation-test splits with an 80-10-10 ratio. Note that this splitting is done before any data cleaning or augmentation, so the validation and test sets are not affected by these processes. Further implementation details are provided in Appendix A.

4.2 Automatic Evaluation Results

The main results are shown in Table 2. We independently test 4 features, namely data cleaning, data augmentation, PET embedding, and kNN. Based on the results of these tests, our final models then use combinations of some or all of these features. From the results in Table 2, we make the following observations:

1. **The data augmentation methods lead to slight increases in performance.** This is true for both data cleaning and augmentation, demonstrating the usefulness of reducing noise and adding high-quality training data. In general, augmentation methods lead to larger gains because adding more data is especially useful in our task, where each PET may appear in the original training data only a few times.

2. **Using embeddings of the PET embeddings (instead of the `[CLS]` classifier token) significantly increases performance.** As hypothesized, this is likely because the `[CLS]` token may have too wide of a scope since it needs to represent the entire sentence, while the PET tokens can specifically give us information about the PET terms we are trying to classify.

3. **KNN models lead to slight increase, while DAN models lead to significant decrease, in performance.** In general, our changes in the data side have much greater effects than our changes in the modelling side. For kNN, we think that the neighbors may provide slight signals but are likely drowned out by the original logits, which leads to incremental changes.

We note that the advantages of the kNN method may increase with more data, as this method benefits greatly from a larger datastore. However, as *EuphAug-S* has a relatively large number of examples compared to the training data, we decided to construct the datastore based on only the original training data, as we did not know if there was any significant domain shift between the test data and *EuphAug-S*, and we did not judge the additional samples to be worth this risk.

These three observations motivate our choices for final models to ensemble. In addition, we submit our final ensembled model to the Shared Task leaderboard, and it received an F1 score of 0.881, ranking first place among all submissions.

5   Related Work

Euphemism detection is a relatively underexplored task. In this paper, we use the euphemism PET dataset gathered by Gavidia et al. (2022). Lee et al. (2022) also use this dataset, but for the task of extracting PETs from a given sentence. In the past, other methods have focused specifically on certain types of euphemisms, such as drugs (Zhu et al., 2021), firing/lying/stealing (Felt and Riloff, 2020), and hate speech (Magu and Luo, 2018).

Below, we further detail some of the methods and techniques previously explored in this area. Zhu et al. (2021) use BERT and the masked language model objective to create candidate euphemisms based on input target keywords of sensitive topics. Zhu and Bhat (2021) extend this to multi-word euphemistic phrases using SpanBERT. Similar to the previous paper, they also generate and filter a list of euphemistic phrase candidates, then rank these candidates using probabilities from the masked language model. Meanwhile, Felt and Riloff (2020) use sentiment analysis methods to detect euphemisms, exploring various properties associated with sentiment such as affective polarity, connotation, and intensity.

Other studies contextualize euphemism detection in a specific use case. For instance, Magu and Luo (2018) train models to detect hateful content or euphemistic hate speech. They employ word embeddings and network analysis, creating clusters of euphemisms by using eigenvector centralities as a ranking metric. Furthermore, euphemism detection can also be used in crime detection. Yuan et al. (2018) analyze jargon from cybercrime marketplaces to find patterns in phrases or code words commonly used in underground communications. However, these two methods use static word embeddings, which do not take into account the context. This may affect performance, as context is very important for euphemisms. In contrast, our method uses context-aware embeddings of transformer-based models.

6   Conclusion and Future Work

We proposed EUREKA, a method for classifying euphemistic usage in a sentence. This is an ensemble model that uses ideas such as data cleaning, data augmentation, representations of Potentially Euphemistic Terms (PETs), and k-nearest-neighbor predictions. Our EUREKA system achieves a score of 0.881 and ranks first on the public leaderboard for the Shared Task.

In the future, we hope to extend our methods to dysphemisms or other figurative language instances. It is also interesting to consider a zero-shot setting for euphemism detection, where euphemisms during test time are unseen during training. Figurative language generation, rather than detection, could also be a fruitful area to explore.

Limitations

Our current model and classifer are deficient in terms of their interpretability on certain aspects, and it would be interesting to explore more interpretable models to ensure that the features used to classify euphemisms can transfer to other scenarios. The models and datasets are limited to English (Bender and Friedman, 2018), and euphemisms in other languages are definitely worth exploring. However, this was not in the scope of the shared task

Due to computational resources, we were not able to explore larger models. For example, it is possible that larger models such as GPT-J or GPT-Neo would perform better on this task. However, we leave this to future work.

References

Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. 2022. Neuro-symbolic language modeling with automaton-augmented retrieval. In *International Conference on Machine Learning*, pages 468–485. PMLR.

Emily M Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 250–259, Sofia, Bulgaria. Association for Computational Linguistics.

Kaustubh D Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahendiran, Simon Mille, Ashish Srivastava, Samson Tan, et al. 2021. Nl-augmenter: A framework for task-sensitive natural language augmentation. *arXiv preprint arXiv:2112.02721*.

Christian Felt and Ellen Riloff. 2020. Recognizing euphemisms and dysphemisms using sentiment analysis. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 136–145, Online. Association for Computational Linguistics.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Benoit Frenay and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869.

Martha Gavidia, Patrick Lee, Anna Feldman, and Jing Peng. 2022. Cats are fuzzy pets: A corpus and analysis of potentially euphemistic terms. *CoRR*, abs/2205.02728.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Taylor Berg-Kirkpatrick. 2021. Investigating robustness of dialog models to popular figurative language constructs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7476–7485.

Savo Fouad Karam. 2011. Truths and euphemisms: How euphemisms are used in the political arena.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models.

Patrick Lee, Martha Gavidia, Anna Feldman, and Jing Peng. 2022. Searching for PETs: Using distributional and sentiment-based methods to find potentially euphemistic terms. In *Proceedings of the Second Workshop on Understanding Implicit and Underspecified Language*, pages 22–32, Seattle, USA. Association for Computational Linguistics.

Emmy Liu, Chenxuan Cui, Kenneth Zheng, and Graham Neubig. 2022. Testing the ability of language models to interpret figurative language. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4437–4452, Seattle, United States. Association for Computational Linguistics.

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. Cite arxiv:1907.11692.

Rijul Magu and Jiebo Luo. 2018. Determining code words in euphemistic hate speech using word embedding networks. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 93–100, Brussels, Belgium. Association for Computational Linguistics.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193(0):217 – 250.

Zahra Nazari, Masoom Nazari, Mir Sayed Shah, and Dongshik Kang. 2018. Evaluation of class noise impact on performance of machine learning algorithms.

Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48.

Simone Tedeschi, Federico Martelli, and Roberto Navigli. 2022. ID10M: Idiom identification in 10 languages. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2715–2726, Seattle, United States. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Kan Yuan, Haoran Lu, Xiaojing Liao, and Xiaofeng Wang. 2018. Reading thieves' cant: Automatically identifying and understanding dark jargons from cybercrime marketplaces. In *USENIX Security Symposium*.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Wanzheng Zhu and Suma Bhat. 2021. Euphemistic phrase detection by masked language model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 163–168, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wanzheng Zhu, Hongyu Gong, Rohan Bansal, Zachary Weinberg, Nicolas Christin, Giulia Fanti, and Suma Bhat. 2021. Self-supervised euphemism detection and identification for content moderation. In *42nd IEEE Symposium on Security and Privacy*.

Appendix A   Implementation Settings

For most methods, we use a batch size of 4, learning rate of 5e-6, and we train for 10 epochs. Training was done mostly on a Google Colaboratory environment using Tesla V100, P100 GPUS, and on a workstation having NVIDIA Quadro RTX 6000 with 24GB of VRAM. With RoBERTa-large, training for 10 epochs took around 30-40 minutes. We use the HuggingFace library (Wolf et al., 2020) for model implementation, as well as for implementing the Trainer function. All other hyperparameters (e.g. learning rate decay, warmup steps, etc.) follow the default ones used by the Trainer function in HuggingFace.