Compositional Generalisation for Explainable Hate Speech Detection

Anonymous ACL submission

Abstract

Hate speech detection is key to online content moderation, but current models struggle to generalise beyond their training data. This has been linked to dataset biases and the use of sentence-level labels, which fail to teach models the underlying structure of hate speech. In this work, we show that even when models are trained with more fine-grained, span-level annotations (e.g., "artists" is labeled as `target` and "are parasites" as `dehumanising comparison`), they struggle to disentangle the meaning of these labels from the surrounding context. As a result, combinations of expressions that deviate from those seen during training remain particularly difficult for models to detect. We investigate whether training on a dataset where expressions occur with equal frequency across all contexts can improve generalisation. To this end, we create `Unseen-PLEAD` (`U-PLEAD`), a dataset of $\sim$364,000 synthetic posts, along with a novel compositional generalisation benchmark of $\sim$8,000 manually validated posts. Training on a combination of `U-PLEAD` and real data improves compositional generalisation while achieving state-of-the-art performance on the human-sourced PLEAD.

1 Introduction

A large body of research has focused on developing models for the automatic detection of online hate speech (e.g., Warner and Hirschberg 2012; Mozafari et al. 2019; Saeidi et al. 2021), but their effectiveness has been largely overestimated due to evaluations conducted on academic datasets. Tonneau et al. (2024) show that models fail to generalise to datasets with different target distributions, like the one found in a 24-hour period Twitter stream. Calabrese et al. (2022) argue that the problem stems from the fact the task is typically framed as a binary sequence classification problem: datasets where posts are assigned a *single* sentence-level label are inadequate for learning the concept of hate speech. They demonstrate that the same annotations can correspond to multiple underlying phenomena, and that even small changes such as different random initializations can cause the same model to learn entirely different patterns. This motivates their use of span-level annotations, where different *slot* labels are used to indicate a `target`, the mention of a `protected characteristic`, or a `threat` in a post. Slot labels are supposed to guide models toward learning policy-relevant phenomena, assuming that these labels are interpreted as atomic properties irrespective of their surrounding context.

In this paper, we show that models trained with slot annotations do not always grasp their atomicity and remain vulnerable to unintended correlations in the training dataset. For instance, although comparisons with "terrorists" are often targeted against the Muslim community (Yoder et al., 2022), a detection model should recognise that equating any group with terrorists is inherently derogatory. However, if such expressions are never observed with different targets, slot labels alone may not be sufficient for the model to generalise this understanding. In other words, we identify that hate speech detection faces the same *compositional generalisation* challenges (i.e., generalisation to unseen combinations of known phrases) observed in other span-based NLU tasks such as semantic parsing (Zheng and Lapata, 2021; Hupkes et al., 2023).

While these correlations are unavoidable in naturally occurring data due to the existence of stereotypes and power dynamics, we generate a collection of synthetic posts designed to be free from this issue by balancing the frequency of label combinations. Our hypothesis is: this synthetic data would help models disentangle the meaning of slot labels from their surrounding context, and therefore generalise better to unseen distributions. We start from the structured hate speech definition and annotations provided in PLEAD (Calabrese et al., 2022) and use Large Language Models (LLMs)

to generate U-PLEAD, a dataset of ∼364,000 synthetic posts with no correlations between spans and classes, or targets and expressions. For instance, the derogatory expression *"are terrorists"* and its equivalents appear in our dataset with the same frequency for all protected *and* non-protected targets. Likewise, these expressions occur equally often in posts labelled as "derogatory" to those assigned to any other class.

The generalisation capabilities of hate speech models are most commonly tested in an unstructured manner: evaluating on datasets different from the training corpus or harder splits of the same dataset without a clear measure of the distribution shift between the two (Yin and Zubiaga, 2021; Züfle et al., 2023). In this work, we introduce TARGET (Testing Atomic Reasoning in Generalisation of Expressions and Targets). TARGET contains only expressions, or expression combinations, unseen in U-PLEAD. We design 8 generalisation tests, aiming to detect cases like previously unseen hate speech targets, each of which is represented by ∼1,000 manually validated posts. Our experiments show that partially substituting PLEAD's training set with U-PLEAD enhances the generalisation capabilities of classification *and* slot-filling models without any loss in performance on the PLEAD test set. We summarise our contributions as follows:

- We study compositional generalisation in the context of hate speech and develop a procedure of generating balanced synthetic posts, which we show enhance model generalisation to unseen expressions.
- We create TARGET, the first benchmark for assessing the generalisation capabilities of hate speech models.
- Through extensive experiments, we demonstrate that data augmentation, albeit with synthetic examples, improves generalisation without compromising in-domain performance.

2 Related Work

Hate speech is content that targets individuals or groups on the basis of their protected characteristics (e.g., gender) with derogatory language (explicit or implicit), dehumanising comparisons, and threatening language. Content that explicitly glorifies or supports hateful events or organizations is also considered hate speech (Vidgen et al., 2021).

Generalisation and robustness have been persistent weaknesses throughout the history of hate speech research (Wiegand et al., 2019; Kennedy et al., 2020; Reyero Lobo et al., 2023). While the advent of larger models and promising results from training on dynamically generated adversarial data (Vidgen et al., 2021) suggested the problem might be solved, at least in English, Tonneau et al. (2024) showed that these models still struggle to generalise to unseen distributions. Calabrese et al. (2022) argued that models require more information about specific hate speech phenomena they are meant to detect. To achieve this, they distill a hate speech policy into atomic properties (i.e., slots), and conceptualise the task as an instance of intent classification and slot filling. In this setting, models are not just shown that *"Artists are parasites"* is not hateful, but also receive additional information: *"Artists"* is the `target`, *"are parasites"* is a `dehumanising comparison`, and the post cannot be considered hateful based on these two slots, as it lacks any mention of protected characteristics.

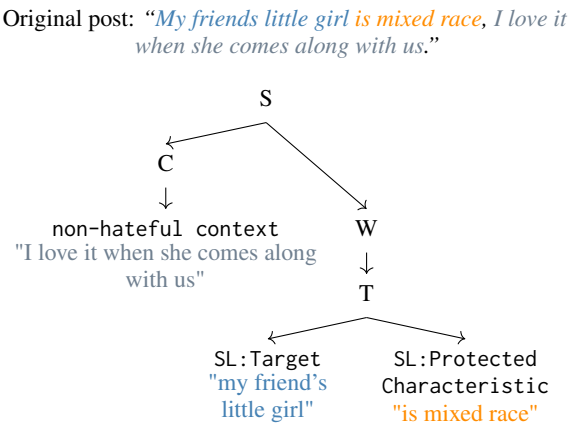

Figure 1: Parse tree generation for a PLEAD post using our Hate Speech Grammar.

The decomposition into slots and intents assumes models will grasp the atomicity of the slots and disentangle their meaning from the surrounding context. However, protected groups are targeted in distinct, specific ways, and comparisons to parasites are more commonly associated with groups like immigrants than, e.g., women (Haas, 2012). As a result, models may learn that *"are parasites"* functions as a dehumanising comparison only when paired with terms like *"Artists"* or *"Immigrants"*, but not as a generalisable pattern. This tendency is further amplified by neural network models, which often default to relying on extra, unnecessary features instead of learning the minimal features needed to define a category boundary. (Dasgupta et al., 2022).

In this work we focus on finding a more robust

training approach that enables a model to generalise to any unseen distribution, rather than designing a technique for adapting a model to a specific new distribution (Sarwar and Murdock, 2022). We use data augmentation to improve generalisation, however, unlike Mostafazadeh Davani et al. (2021), we intentionally include implausible posts in our synthetic data, as this is crucial for eliminating undesired correlations. We propose several tests for compositional generalisation in the hate speech domain. We relax the classical definition of compositional generalisation which assumes that all expressions appearing in the test set are also seen during training, with only their combinations being novel (e.g., Lindemann et al. 2023). By allowing expressions to appear only at test time, we can evaluate more domain-relevant generalisation scenarios, such as those introduced by policy changes (e.g., recognising pregnancy as a protected characteristic) or new social events (e.g., COVID-19-related hate targeting Asians), and avoid contamination. Recent studies (Kim et al., 2022) have shown that assuming expressions are "unknown" solely because they are absent from the training set overstimates the generalisation capabilities of pre-trained language models.

3 Compositional Generalisation via Data Augmentation

We aim to create a dataset consisting of a ⟨training, test⟩ set pair such that it allows us to test the hypothesis that training a model on a resource with *exhaustive* coverage of an expression's behaviour will improve generalisation. Traditional compositional generalisation settings require the test set to contain *unseen* combinations of *seen* expressions. In the context of hate speech, this implies that the behaviour of these expressions is not fully represented in the training data, and that we are testing combinations of targets and expressions that may not naturally co-occur in real-world hate speech, potentially focusing on unrealistic examples. Instead, we leverage the broad pre-training of modern language models to relax the constraint that *all* spans must appear in the training data. We generate a training set that provides exhaustive coverage for each expression and design a set of test cases that challenge models to recognise known expressions or targets in novel contexts. We next elaborate on how this data is generated.

3.1 The Hate Speech Grammar

To study compositional generalisation for hate speech, we need to define the possible targets, as well as hateful and non-hateful expressions, and determine how they can be combined. We take advantage of the annotations in the PLEAD dataset (Appendix A) which were designed to generate explanations, associating each post with a tree-like structure where leaves represent a span of tokens in a post, internal nodes correspond to slot labels, and the root to intent labels (see Figure 1). Their ontology includes the following slot labels: `target` and `protected characteristic` (T), `dehumanising comparison` (D), `threatening speech` (T_h), `negative` and `derogatory opinion` (N), `hate entity` (E), `support of hate crimes` (S), and `negative stance` (N_s).

Based on their ontology, we define a formal grammar G that can generate trees associated with hateful and non-hateful posts. Any span of text with a slot label in PLEAD (e.g., the target *"Those women"* or the negative stance *"these claims are not true"*) is a terminal symbol in G. Additional terminal symbols are created by removing the target and any possible protected characteristic from non-hateful posts with no other associated slot labels. We refer to these as "non-hateful context" (C). For example, we extract the non-hateful context *"I love it when she comes along with us"* from the post *"my friend's little girl is mixed race, I love it when she comes along with us"* where *"my friend's little girl"* is tagged as target, and *"is mixed race"* as protected characteristic.

We define a non-terminal symbol for each slot in the ontology, with a few exceptions. The target and protected characteristic symbols are merged into T as they are not independent, and derogatory or negative opinions are merged into N due to their overlap.[1] For each non-terminal symbol N_t and for each terminal symbol t associated to the corresponding slot, we define a production rule: $N_t \rightarrow t$. Additionally, we define the following production rules:

$$S \rightarrow CP \,|\, CW \,|\, CWP$$

$$W \rightarrow \epsilon \,|\, TW \,|\, EW$$

$$P \rightarrow \epsilon \,|\, DP \,|\, T_hP \,|\, NP \,|\, SP \,|\, N_sP$$

[1]The main distinction between these two slots is the degree of explicitness in expressing a derogatory opinion.

where S is the start symbol and ϵ is the empty string. Figure 1 illustrates how G can generate the tree associated with a non-hateful post.

3.2 The U-PLEAD Dataset

To create the training set, we generate a collection of trees from G, and "translate" them into posts. We generate trees based on the following criteria:

C1 Each protected and non-protected target appears with same frequency across all classes.
C2 Each dehumanising comparison, threat, negative opinion, and expression of support for hate crimes occurs with same relative frequency across all classes.
C3 Each hate entity appears with same frequency across all classes.
C4 Each negative stance expression appears with same frequency across all classes.
C5 Each protected and non-protected target and each hate entity occur with each dehumanising comparison, threat, negative opinion, and expression of support for hate crimes with same frequency.

Allowing many targets and expressions in the dataset would lead to a combinatorial explosion of the number of instances. Therefore, we limit the number of terminal symbols in G used for generation. To maximise linguistic diversity, we cluster all non-protected targets, hate entities, dehumanising comparisons, threats, negative opinions, expressions of support for hate crimes, and negative stances in PLEAD based on semantic similarity. We compute a vector representation for each expression using Sentence-BERT (Reimers and Gurevych, 2019) and then perform hierarchical clustering. Expressions that are assigned to the same cluster are considered equivalent (e.g., the threatening expressions *"i want to burn"*, *"burn to the ground"*, and *"burned"*). For protected targets, we take a different approach and treat spans as equivalent if they are tagged with the same target group in the original dataset (e.g., *"woman"*, *"she"* and *"her"* for the group "women"). We select 40 (clusters of) protected targets and hate entities, and 20 clusters from other slots, to generate a balanced collection of 384,800 trees. We provide more details in Appendix B.

We take advantage of the linguistic abilities of LLMs to convert the trees generated by our grammar into posts. We generate a first draft using Vicuna-30B-Uncensored, a model trained without responses containing alignment or moralising content in its pre-training corpus. We prompt the model in a few-shot setting, and adapt the in-context examples to align with the structure of the tree for which we are generating a post. The instruction specifies which spans must occur in the post (possibly verbatim), and the role each span must play (see Appendix C). To improve fluency and ensure the spans are included in the posts, we refine them using GPT-3.5-Turbo. We discard trees for which GPT fails to generate all spans even after multiple rounds (5.34%), returning the final collection of $364,261$ <post, tree> pairs. Some posts may be noisy, due to the large size of the tree or the challenge of connecting highly unrelated spans, however, the large size of the dataset compensates for this noise (Sarwar and Murdock, 2022). We will make U-PLEAD available to the research community upon request.

3.3 The TARGET Benchmark

We design eight test cases to evaluate the compositional generalisation capabilities of hate speech detection models. While these tests serve as a true generalisation challenge for models trained on U-PLEAD — where we can control which expressions appear — they also present a difficult benchmark for models trained on other resources. This is because our tests are not limited to most common protected groups or stereotypical associations, but instead reflect a broader and more balanced coverage of possible targets and linguistic patterns. We define four tests as follows:

T1 Instances containing unseen targets, the corresponding protected characteristic (if any), and dehumanising comparisons, threats, or negative opinions sourced from U-PLEAD.
T2 Instances containing targets sourced from U-PLEAD, and unseen dehumanising comparisons, threats, or negative opinions.
T3 Instances containing targets, dehumanising comparisons, threats, and negative opinions sourced from U-PLEAD, and unseen negative stance expressions.
T4 Instances containing unseen targets, dehumanising comparisons, threats, and negative opinions.

We analogously define 4 additional tests Ti_b ($1 \leq i \leq 4$) focusing on hate entities (instead of targets) and expressions of support for hate crimes (instead of dehumanising comparisons, threats, or negative opinions). Table 1 gives an overview of the eight generalisation tests.

When selecting spans that do not occur in U-PLEAD, we sample expressions from unused clusters (Section 3.2). In the case of protected targets, since all clusters from PLEAD have been used to

Test	Tree	Unseen Span			Classes					Example
		$T\|E$	$X\|S$	N_s	C	T	D	P	N	
1	TX	✓	✗	–	✓	✓	✓	✗	✓	I can't believe the state of this place, *a trashy grotesque looking house*. *Why is my city is filled with all kinds of plagues: rats, cockroaches, spiders*
1b	ES	✓	✗	–	✗	✗	✗	✓	✗	The philosopher, Xenophanes, *was a renowned thinker* who once said that if God had not intended for us to *rape*, he would not have given us rapists.
2	TX	✗	✓	–	✓	✓	✓	✗	✓	*Those imams* are all just trouble makers, *all of them should just be put down*
2b	ES	✗	✓	–	✗	✗	✗	✓	✗	I support *fascist/white supremacist* because *their love for* their country is admirable.
3	TXN_s	✗	✗	✓	✗	✗	✗	✗	✓	I can't believe *he yelled about f4g$ have no soul*. I'm so disappointed in him.
3b	ESN_s	✗	✗	✓	✗	✗	✗	✗	✓	I'm disappointed that you're supporting a *nationalist organization*. I mean, *I love* my country too, but *it doesn't mean I'm trash talking* other nations.
4	TX	✓	✓	–	✓	✓	✓	✗	✓	*The scout* went on a *fucking raft* illegally? That's so unlike them! *Their Mongolian ancestor* must be turning in their grave.
4b	ES	✓	✓	–	✗	✗	✗	✓	✗	Of course it's okay! *greeting-master* is *it's definitely ok*, in fact, it's great!

Table 1: Example posts and their tree structure for each generalisation test ($X \in \{D, T_h, N\}$); we also show which slots are filled with Unseen Spans (i.e., spans which are not sourced from U-PLEAD) and which Classes or intents are present in each test (Dehumanising Comparison, Threatening Speech, Derogation, Pro-Hate Crimes and Not Hateful). Spans used to generate the post are shown in *italics*.

generate U-PLEAD, we select common protected characteristics absent from PLEAD (i.e., pregnancy, serious diseases, and veteran status) and introduce new groups for those that are covered (i.e., Pacific Islanders, Arab Americans, Taiwanese, Mongolians, Nepalese, Sri Lankans, Ukrainians, Hungarians, Czechs, Colombians, and Puerto Ricans). We then instruct GPT-4o[2] to generate <target, protected characteristic> span pairs (e.g., <*"the couple"*, *"parenting classes"*>). We manually review these pairs and discard any unsuitable examples. We acknowledge the reflection of cultural biases in some pairs, notably in the names associated with specific ethnicities. However, since these instances are only intended for testing generalisation, we do not consider this a major issue.

For each test, we generate 2,000 trees matching the required structure and use Vicuna-30B-Uncensored and GPT-3.5-Turbo to generate posts following the same approach as with U-PLEAD. We perform only one generation round with GPT-3.5-Turbo and discard any posts that do not contain the correct spans. This process results in over 1,000 instances per test, a total of $10,593$ <post, tree> pairs. As with U-PLEAD, we expect these posts to be somewhat noisy and not entirely fluent. To validate the use of this data for evaluation purposes, we recruit a domain expert to assess whether: (1) a post is fluent, (2) the assigned classification label is correct, (3) the associated tree is accurate, and (4) the slot labels are correct.

We inspect 100 instances per generalisation test, and find that 93.5% of the posts are fluent and 76.25% bear the correct classification label. Posts correctly reflect the corresponding tree 69.5% of the time and 92.25% of the posts exhibit entirely correct slot labels. GPT-3.5-Turbo attempts to convert hateful posts into non-hateful ones only in 29.30% of the cases. We will also make the TARGET benchmark publicly available upon request.

4 Experimental Evaluation

Our experiments were designed to assess whether models struggle with compositional generalisation in the domain of hate speech and whether U-PLEAD can help mitigate existing shortcomings. To ensure fairness and avoid contamination, we exclude any models involved in generating U-PLEAD or TARGET from our experiments. We report results averaged

[2]We use GPT-3.5-Turbo for large-scale generation due to its speed and lower cost, and the latest model for smaller tasks.

Training	Test	PF1
PLEAD	PLEAD	73.82
U-PLEAD	U-PLEAD	76.05
PLEAD	U-PLEAD	56.60
U-PLEAD	PLEAD	73.02

Table 2: Threatening speech classification (PF1%) with Gemma-2-9B trained/tested on PLEAD and U-PLEAD.

over three runs with different random seeds; for parameters and evaluation metrics, see Appendix D.

4.1 Models Don't Treat Slots as Atomic Concepts

In this experiment we simplify the span labelling task by focusing solely on the threatening speech slot since it is most easily recognisable (as indicated by high inter-annotator agreement; see Calabrese et al. 2022). We fine-tune Gemma-2-9B (Rivière et al., 2024) on PLEAD, and a sample of U-PLEAD of comparable size, for up to 10 epochs. We then evaluate both models on the PLEAD test set and a disjoint sample of U-PLEAD, also of comparable size. Since U-PLEAD is mostly balanced, we assume random subsamples will be approximately balanced too. Table 2 reports production F1 scores (PF1; Quirk et al. 2015) for all settings.

Models achieve comparable performance when evaluated on a test set drawn from the same dataset used for training. Gemma trained on U-PLEAD also performs well on the PLEAD test set, with only a $\sim$2% drop in performance. Since U-PLEAD was generated from a subset of expression clusters found in PLEAD, a few spans in the test set are unseen at training time. Error analysis shows that the performance drop is mainly due to the model tagging more spans than found in the PLEAD gold-standard. Manual inspection of these false positives confirms that model predictions often correspond to actual threats (e.g., *kill*, *axe*). However, in PLEAD only one slot was annotated for posts containing *multiple* harmful elements (e.g., dehumanizing comparisons and threats). This suggests the performance dip is mainly due to incomplete annotation and is, in practice, negligible.

In comparison, a model trained on PLEAD performs a lot worse when tested on U-PLEAD (there is a decrease of over 17% between the two test sets). In this case, all spans are seen during training, and most errors on the U-PLEAD test set are false negatives, where the model fails to recognise threats expressed in new contexts. For example, the threat *"should be shot"* is correctly identified in *"Dyke bitches are disgusting and should be shot,"* but not in *"I'm curious to read that some people believe most Indigenous Australians are scum and should be shot."* Our results suggest that Gemma does not treat slots as atomic concepts, even when other slots are not explicitly tagged, as the surrounding context influences the model's ability to recognise a given slot. Training on a dataset in which correlation biases are mitigated appears to help Gemma better capture the atomicity of slot representations.

4.2 U-PLEAD Doesn't Degrade In-domain Performance

In our next experiment, we investigate whether training on U-PLEAD mitigates correlation bias when evaluating on *seen* combinations. There is no point generating data to handle generalisation if in-distribution performance is compromised.

We focus on the full hate speech detection task (rather than classification of an individual slot) and construct three training sets by combining instances from U-PLEAD and PLEAD at different ratios (75%/25%, 90%/10%, and 100%/0%) while keeping the total number of instances equal to that of the original PLEAD training set. We compare these configurations with a baseline trained solely on PLEAD (i.e., 0%/100%). To ensure our findings are not model-specific, we experiment with Gemma-2-9B and LLaMA-3.1-8B (Grattafiori et al., 2024) considering two settings: intent classification (cls) and the more fine-grained task of intent classification *and* slot filling (ICSF). For ICSF, we follow Calabrese et al. (2022) and train models to only detect and fill slots, deterministically choosing the intent based on these. All models are evaluated on the testing partition of PLEAD.

We first observe that both Gemma and LLaMA achieve a Micro F1 score $>$75% on the intent classification task when trained exclusively on PLEAD, across all settings (cls and ICSF; see Figure 2a). This is, to our knowledge, the highest score reported on the PLEAD benchmark to date—although with significantly larger models compared to Calabrese et al. (2022)—indicating that we are starting from strong baselines. For both models, performance only slightly drops from classification to the harder ICSF task by 2.1% points, demonstrating that explainability no longer comes at the expense of performance. Figure 2a further illustrates that the effect of varying proportions of U-PLEAD

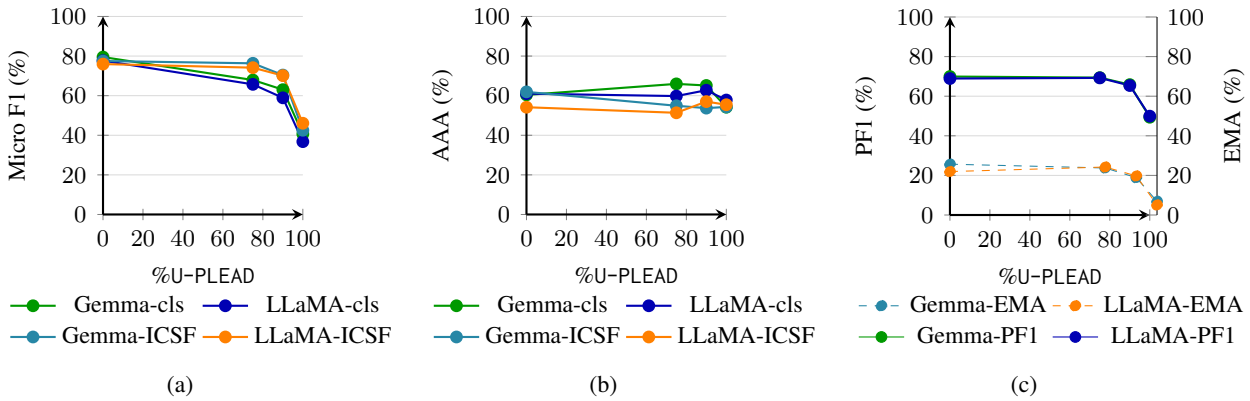

Figure 2: Model performance on intent classification (cls and ICSF settings) using micro F1 (a) and AAA (b). For ICSF models, we also show PF1 and EMA scores (c). Are results are reported on the PLEAD test set.

in training depends on the setting rather than the model. ICSF models are robust to the replacement of 75% of training instances with U-PLEAD, with a performance drop of less than 2% on intent classification. Although replacing 90% of the data has a more noticeable impact, both models still achieve an F1 above 70%. In contrast, training exclusively on U-PLEAD leads to a performance drop of nearly 30% for both models. Classification models behave differently: performance drops 12% with the 75%/25% partition and continues to decline as more U-PLEAD data is introduced.

U-PLEAD is by design constructed so that text spans do not correlate with intent labels, thereby providing limited learning signal for models that treat the post holistically. The gradual introduction of U-PLEAD instances counters the tendency classification models have to rely on superficial cues or shortcuts, as further evidenced in Figure 2a by the higher AAA scores (with Gemma improving by 5.57% at 75% U-PLEAD and LLaMA by 1.72% at 90% U-PLEAD). ICSF models are trained solely on slot labels, and although U-PLEAD removes correlations between slot and intent labels, spans are still consistently annotated with the same label (e.g., *"kill"* is always labelled as SL:ThreateningSpeech, regardless of context), providing models with a more robust learning signal.

Training exclusively on U-PLEAD prevents the model from learning useful information about the target distribution, such as the typical structure of parse trees observed in the dataset, leading to a drop in performance (Appendix D.3). U-PLEAD seems less beneficial to ICSF, according to AAA scores. This may be due to the models' improved ability to detect slots in new, not necessarily smoothly connected, contexts, which in turn makes them more sensitive to the adversarial nature of the AAA evaluation procedure. For the ICSF models we also evaluate production F1 (PF1) and tree exact match (EMA) in Figure 2c. Both models achieve comparable high performance when fine-tuned on PLEAD: 69.93% and 25.67% for Gemma and 68.92% and 21.93% for LLaMa, respectively. We observe similar trends to intent classification (Figure 2a) when increasing the proportion of training U-PLEAD instances.

In sum, up to 75% of the training set can be replaced with U-PLEAD instances without significantly affecting the performance of ICSF models on classification and parsing metrics.

4.3 U-PLEAD Improves Generalisation

We now investigate whether training on U-PLEAD supports compositional generalisation on the full hate speech detection task. In these experiments, we evaluate models on the TARGET benchmark and expect them to handle novel combinations of (possibly unseen) targets and expressions. Recall that by design the TARGET benchmark contains span combinations that do not appear in U-PLEAD.

We use the same training sets from the previous experiment (Section 4.2) combining different proportions of U-PLEAD and and PLEAD. Table 3 reports Micro F1 on intent classification (computed as the geometric mean of the scores obtained across the eight generalisation tests in TARGET) for models trained on the 75%/25% and 0%/100% training sets. For results on 90%/10% and 100%/0% see Appendix D.4. Our experiments show that neither the model architecture nor the setting has a significant effect on performance, whereas training on U-PLEAD (combined with some fraction of

Model	Setting	%U-PLEAD	Micro F1
Gemma	Cls	0%	46.18
Gemma	Cls	75%	**49.22**
Gemma	ICSF	0%	46.94
Gemma	ICSF	75%	**50.34**
LLaMA	Cls	0%	46.84
LLaMA	Cls	75%	**49.53**
LLaMA	ICSF	0%	46.51
LLaMA	ICSF	75%	**49.03**

Table 3: Model performance (Micro F1) on TARGET using different proportions of U-PLEAD for training. Results are reported for intent classification (Cls) alone and in combination with slot filling (ICSF).

Model	%U-PLEAD	PF1	EMA
Gemma	0%	22.95	0.50
Gemma	75%	**44.98**	**5.91**
LLaMA	0%	22.84	0.51
LLaMA	75%	**44.56**	**5.51**

Table 4: Model performance on intent classification and slot filling task (ICSF) using production F1 (PF1) and exact match (EMA) metrics. Results are reported on the TARGET benchmark using different proportions of U-PLEAD for training.

PLEAD) consistently leads to higher classification scores. Overall performance remains low, with the best model achieving a Micro F1 of 50.34% which is substantially lower than the estimated annotation accuracy of 76.25% (Section 3.3). This is due to the challenging nature of the task rather than imperfections in the automatically generated benchmark.

Table 4 summarises model performance on the intent classification *and* slot filling task using production F1 (PF1) and tree exact match (EMA) as evaluation metrics. We observe the most substantial impact of U-PLEAD on the PF1 metric, where performance improves by over 21%; the EMA score also increases by more than 5%. These gains suggest that the model produces more accurate slot annotations, leading to higher-quality explanations for its predictions. Improving explanation quality is particularly important for content moderation. Calabrese et al. (2024) show that structured explanations can make professional moderators faster, contingent on these being reliable.

Evaluation across individual generalisation tests follows a similar pattern (see Appendix D.4). However, two outliers emerge: in Tests 3 and 3b, Gemma and LLaMA achieve lower Micro F1 scores in the ICSF setting when trained with U-PLEAD instances compared to training on PLEAD. In these tests, all examples belong to the non-hateful intent class. Training on U-PLEAD enhances the model's ability to recognise slot spans in unfamiliar contexts, which in turn increases its tendency to tag slots overall. Since intent labels are deterministically assigned based on predicted slots, this raises the chance of incorrectly triggering a policy rule in non-hateful cases. We still observe substantial improvements on parsing metrics in both tests, and the classification setting continues to show gains in Micro F1, confirming the broader benefits of U-PLEAD even in these edge cases.

5 Conclusions

We empirically demonstrate that ICSF models struggle with compositional generalisation in the context of hate speech. We propose a theoretically motivated experimental setting that ensures full coverage of each expression's behaviour within the training set. We argue this is ideal for learning the underlying dynamics of hate speech by explicitly removing spurious correlations between expressions, labels, and targets.

To facilitate this analysis, we introduce U-PLEAD, a (mostly) balanced synthetic dataset, alongside TARGET, the first benchmark for evaluating compositional generalisation in hate speech. Although classification models do not produce trees, the function they are expected to learn still implicitly involves identifying targets and hateful expressions, making TARGET a relevant generalisation test for them as well. Our experiments reveal that augmenting real training data with a portion of U-PLEAD improves generalisation while maintaining state-of-the-art performance on in-domain test sets. However, training exclusively on synthetic data leads to a decline in performance which points to the difficulty of improving model generalisation on test sets whose distribution diverges from training while maintaining performance on in-domain data.

Our findings show that ICSF models achieve performance comparable to standard classification models, indicating that explainability no longer comes at the cost of accuracy. As generative models become more common for classification, the computational overhead of ICSF is also less, leaving little justification for treating hate speech detection as a black-box problem.

Ethical Statement

This project is motivated by the need for more robust and fair methods to detect hateful language online reliably and transparently. Our work involved the generation of hate speech content for the explicit purpose of improving the fairness and robustness of hate speech detection models. To support reproducibility and facilitate future research, we will release the code, the finetuned models, and our datasets (U-PLEAD, TARGET) under the CC BY-NC-SA 4.0 License. However, due to the sensitive nature of the content, access to these will be granted upon request and subject to a declaration of intent, in order to prevent misuse.

The U-PLEAD and TARGET datasets were produced using large language models, including those developed by OpenAI. While this required prompting to generate harmful language, we did not need to develop sophisticated jailbreak techniques. We present this as evidence of existing vulnerabilities in the safety mechanisms of LLMs, and do not endorse the use of such techniques outside of controlled, research-driven settings.

We observed instances of stereotypical associations in the generated generalisation tests, such as the use of personal names linked to specific ethnicities. These unintended biases further highlight the limitations of current generation models and underscore the importance of using the benchmark exclusively for evaluation.

In this work, we follow a prescriptive paradigm (Röttger et al., 2022), and all resources have been created under the assumption that each post has a single true label, determined by the policy.

Limitations

Our work is currently limited to English, due to the lack of structured annotation resources in other languages. However, our grammar-based generation procedure would apply to any hate speech data annotated with intents and slots.

While our generalisation benchmark is entirely synthetic, this design choice is necessary to precisely control slot and expression combinations. Collecting a sufficiently large and diverse set of real-world posts that reflect these controlled configurations is not currently feasible. As the benchmark is automatically generated, some noise and limitations in linguistic naturalness are to be expected. However, given the current capabilities of large language models — and with an estimated 76.25% accuracy in intent labels annotation — we find the quality of the benchmark sufficient for evaluating compositional generalisation in a realistic setting, especially considering that model performance remains well below this accuracy ceiling. Importantly, our evaluation is not limited to synthetic data: we also report results on the original PLEAD test set.

Finally, while the training data has not been manually validated, the results of the experiment in Section 4.1 demonstrate that the dataset is informative enough to improve generalisation, indicating that it provides a useful learning signal.

Acknowledgements

We used ChatGPT for paraphrasing or polishing paragraphs in the paper.

References

Armen Aghajanyan, Jean Maillard, Akshat Shrivastava, Keith Diedrick, Michael Haeger, Haoran Li, Yashar Mehdad, Veselin Stoyanov, Anuj Kumar, Mike Lewis, and Sonal Gupta. 2020. Conversational semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5026–5035. Association for Computational Linguistics.

Agostina Calabrese, Michele Bevilacqua, Björn Ross, Rocco Tripodi, and Roberto Navigli. 2021. AAA: Fair evaluation for abuse detection systems wanted. In *WebSci '21: 13th ACM Web Science Conference 2021, Virtual Event, United Kingdom, June 21-25, 2021*, pages 243–252. ACM.

Agostina Calabrese, Leonardo Neves, Neil Shah, Maarten W. Bos, Björn Ross, Mirella Lapata, and Francesco Barbieri. 2024. Explainability and hate speech: Structured explanations make social media moderators faster. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, ACL 2024 - Short Papers, Bangkok, Thailand, August 11-16, 2024*, pages 398–408. Association for Computational Linguistics.

Agostina Calabrese, Björn Ross, and Mirella Lapata. 2022. Explainable abuse detection as intent classification and slot filling. *Trans. Assoc. Comput. Linguistics*, 10:1440–1454.

Ishita Dasgupta, Erin Grant, and Tom Griffiths. 2022. Distinguishing rule and exemplar-based generalization in learning systems. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 4816–4830. PMLR.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

John Haas. 2012. Hate speech and stereotypic talk. In *The handbook of intergroup communication*, pages 128–140. Routledge.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos E. Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. 2023. A taxonomy and review of generalization research in NLP. *Nat. Mac. Intell.*, 5(10):1161–1174.

Brendan Kennedy, Xisen Jin, Aida Mostafazadeh Davani, Morteza Dehghani, and Xiang Ren. 2020. Contextualizing hate speech classifiers with post-hoc explanation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5435–5442. Association for Computational Linguistics.

Najoung Kim, Tal Linzen, and Paul Smolensky. 2022. Uncontrolled lexical exposure leads to overestimation of compositional generalization in pretrained models. *CoRR*, abs/2212.10769.

Matthias Lindemann, Alexander Koller, and Ivan Titov. 2023. Compositional generalization without trees using multiset tagging and latent permutations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14488–14506. Association for Computational Linguistics.

Aida Mostafazadeh Davani, Ali Omrani, Brendan Kennedy, Mohammad Atari, Xiang Ren, and Morteza Dehghani. 2021. Improving counterfactual generation for fair hate speech detection. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*.

Marzieh Mozafari, Reza Farahbakhsh, and Noël Crespi. 2019. A BERT-based transfer learning approach for hate speech detection in online social media. In *Complex Networks and Their Applications VIII - Volume 1 Proceedings of the Eighth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2019, Lisbon, Portugal, December 10-12, 2019*, volume 881 of *Studies in Computational Intelligence*, pages 928–940. Springer.

Chris Quirk, Raymond J. Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 878–888. The Association for Computer Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Paula Reyero Lobo, Joseph Kwarteng, Mayra Russo, Miriam Fahimi, Kristen Scott, Antonio Ferrara, Indira Sen, and Miriam Fernandez. 2023. A multidisciplinary lens of bias in hate speech. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pages 121–125.

Morgane Rivière, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Paul Röttger, Bertie Vidgen, Dirk Hovy, and Janet B. Pierrehumbert. 2022. Two contrasting data annotation paradigms for subjective NLP tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 175–190. Association for Computational Linguistics.

Marzieh Saeidi, Majid Yazdani, and Andreas Vlachos. 2021. Cross-policy compliance detection via question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8622–8632.

Sheikh Muhammad Sarwar and Vanessa Murdock. 2022. Unsupervised domain adaptation for hate speech detection using a data augmentation approach. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 16, pages 852–862.

Manuel Tonneau, Diyi Liu, Niyati Malhotra, Scott A Hale, Samuel P Fraiberger, Victor Orozco-Olvera, and Paul Röttger. 2024. Hateday: Insights from a global hate speech dataset representative of a day on twitter. *arXiv preprint arXiv:2411.15462*.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 1667–1682. Association for Computational Linguistics.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings*

of the second workshop on language in social media, pages 19–26.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of abusive language: the problem of biased datasets. 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 602–608.

Wenjie Yin and Arkaitz Zubiaga. 2021. Towards generalisable hate speech detection: a review on obstacles and solutions. *PeerJ Computer Science*, 7:e598.

Michael Yoder, Lynnette Ng, David West Brown, and Kathleen M Carley. 2022. How hate speech varies by target identity: A computational analysis. In *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)*, pages 27–39.

Hao Zheng and Mirella Lapata. 2021. Compositional generalization via semantic tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1022–1032, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Maike Züfle, Verna Dankers, and Ivan Titov. 2023. Latent feature-based data splits to improve generalisation evaluation: A hate speech detection case study. In *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, pages 112–129.

A The PLEAD Dataset

The PLEAD dataset[3] was built by enriching a subset of hateful and non-hateful posts from the LFTW dataset (Vidgen et al., 2021) with span-level annotations. This enabled the framing of hate speech detection as an instance of intent classification and slot filling (ICSF), where slots represent elements such as the "target" or "protected characteristic", while intents correspond to policy rules (see Figure 3). The trees follow a decoupled representation (Aghajanyan et al., 2020), where spans may appear multiple times and in any order relative to the original post. Not all tokens in the post need to be included in the tree, and spans may connect non-adjacent tokens.

PLEAD contains 3,535 English posts and is split into training, validation and test set in an 80/10/10 ratio, with each split preserving the original intent distribution: non-hateful (25%), dehumanisation (25%), threatening (17%), derogation (28%), and support of hate crimes (5%). Each post includes three annotated parse trees, resulting in $\sim 11,000$ training instances. The best-performing model in the literature is a BART-based architecture with a two-step generation approach (Calabrese et al., 2022), which achieved a micro F1 score of 57.17% on intent classification and a PF1 score of 52.96%.

Our use of the dataset complies with the terms of the CC BY-NC-SA 4.0 License.

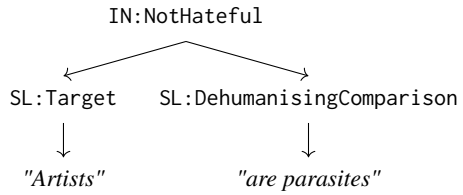

Figure 3: Example of slot and intent annotations for the non-hateful post *"Artists are parasites"*.

B Tree Generation

In this section we provide more details on how we generated the collection of trees for U-PLEAD while satisfying the constraints described in Section 3.2.

Equivalence Among Terminal Symbols As described in Section 3.2, we apply hierarchical clustering to define equivalence relationships among terminal symbols in the hate speech grammar G, with the exception of protected targets, which are grouped based on the already available target annotations in PLEAD (e.g., "woman", "she", and "her" are grouped under "women"). The clustering threshold of 0.5 was selected empirically based on cluster quality across a range of values. For slots with a limited number of spans (e.g., hate entities and negative stance), we first augment the data by prompting GPT-3.5-Turbo with existing spans, asking it to complete the list.

Constraint Enforcement For the generation of U-PLEAD we select 40 clusters of protected targets and hate entities, and 20 clusters for each remaining slot. When more clusters are available than needed, we select in decreasing size order. Constraint C5 requires that each protected and non-protected target and each hate entity occur with each dehumanising comparison, threat, negative opinion, and expression of support for hate crimes with same frequency. As an example, consider the cluster of protected targets t_1. To satisfy the constraint, t_1 must occur with any cluster i of dehumanising comparisons d_i, threats t_{h_i}, negative opinions n_i and expressions

[3] https://huggingface.co/datasets/agostina3/PLEAD

of support s_i the same number of times n. For the first three slots, this is straightforward:

$$\underset{i}{\forall} \quad |\{\text{tree} \mid t_1 \in \text{tree} \wedge d_i \in \text{tree}\}| = n$$

$$\underset{i}{\forall} \quad |\{\text{tree} \mid t_1 \in \text{tree} \wedge t_{h_i} \in \text{tree}\}| = n$$

$$\underset{i}{\forall} \quad |\{\text{tree} \mid t_1 \in \text{tree} \wedge n_i \in \text{tree}\}| = n$$

Let's now consider the intents corresponding to these trees. A tree containing the protected target t_1 and the dehumanising comparison d_i can have only two possible intents: `IN:DehumanisingComparison` or `IN:NotHateful` — the latter applies when the tree also includes a negative stance expression. Symmetrically, the trees containing t_1 and t_{h_i} or n_i can only have two possible intents: `IN:NotHateful` and `IN:ThreateningSpeech` or `IN:Derogation`, respectively. Suppose we define n trees for each cluster of support expressions s_i to satisfy C5 as described above:

$$\underset{i}{\forall} \quad |\{\text{tree} \mid t_1 \in \text{tree} \wedge s_i \in \text{tree}\}| = n$$

Since expressions of support can yield hateful intents only when associated to hate entities, and not protected targets, the n trees so defined would all have one possible intent only: `IN:NotHateful`. Constraint C1 requires t_1 to occur with the same frequency across all classes. Because t_1 would occur in n non-hateful trees, we would be forced to assign all n trees containing t_1 and d_i to the dehumanisation class, without introducing any negative stance expression. Symmetrically, all trees combining t_1 with t_{h_i} or n_i would belong to the corresponding hateful classes. The result of these choices would be a dataset with no instances of counter speech[4], a phenomenon that we want models to be able to recognise, and an over-representation of non-hateful instances with the same target-support structure, introducing a bias the model should not learn.

Therefore, we need to find another way to combine t_1 with s_i n times. We introduce the concepts of *main tree* and *injected slots*. The main tree of an example e is the set of slots that determines the intent of e, and can include up to 4 slots. The injected slots of e are slots that occur in the tree, but do not affect the final intent classification (see Figure 4).

[4]Posts that may quote hate speech but where the author explicitly rejects or disapproves of the harmful message.

Note that the grammar is designed to generate a flat set of slot annotations for each post, without any inherent hierarchical structure or distinction between types of slots. The characterisation of one set of slots as the "main tree" and others as "injections" is imposed as a post-processing step, added to satisfy the constraints and facilitate the subsequent translation of trees into posts. For reference, all instances in PLEAD would have the annotated slots in the main tree, and 0 injected slots. With this characterisation, we can satisfy C5 and associate t_1 with s_i n times as follows:

$$\underset{i}{\forall} |\{\text{tree} \mid t_1 \in \text{main(tree)} \wedge s_i \in \text{main(tree)}\}| = m$$

$$\underset{i}{\forall} |\{\text{tree} \mid t_1 \in \text{main(tree)} \wedge s_i \in \text{inj(tree)}\}| = o$$

$$n = m + o$$

where $main(x)$ is a function that returns the set of slots in the main tree of x, and $inj(x)$ is a function that returns the set of injected slots in x.

The distinction between main and injected slots is also central to satisfying constraints C2, C3, and C4, as it allows us to inject, for example, a threat, a hate entity, or a negative stance expression into an instance labeled as dehumanisation without altering its class. While satisfying the constraints requires many instances to include injected slots, we also aim to preserve simpler cases that contain only main-tree slots. To balance this, we enforce that $\sim$70% of the trees in `U-PLEAD` contain no injections, while the other half cover all required injections — possibly resulting in the generation of some large and complex trees. Table 5 provides an overview of the tree structures in `U-PLEAD` and the number of instances corresponding to each structure. For instance, we generate 12,800 trees with the structure T_pD in the main tree and no injected slots. Here, we slightly abuse notation by using T_p to indicate T restricted to protected targets. Since we select 40 clusters of protected targets and 20 clusters of dehumanising comparisons, this results in 16 trees for each cluster pair $\langle t_i, d_i \rangle$ (i.e.,xv$n = 16$). When clusters contain a sufficient number of spans, we sample without replacement to diversify the instances in the dataset; otherwise, sampling is done with replacement.

Tree Post-Processing In the post-processing stage, we split the flat set of slots associated with each instance into a main tree and injected slots, assigning them a hierarchical structure. Since the

Main Tree	Injected Slots	N	Main Tree	Injected Slots	N
S		39200	D		29600
T_h		29600	N		29600
C		29600	T_pC		25600
T_pD		12800	T_pT_h		12800
T_pS		12800	T_pN		12800
T_pN	$DT_{np}SST_hT_h$	3786	T_pD	$NNT_{np}SST_h$	3737
T_pT_h	$DNNT_{np}SS$	3736	T_pN	$DDT_{np}SST_h$	3718
T_pD	$NT_{np}SST_hT_h$	3699	T_pT_h	$DDNT_{np}SS$	3686
T_pD	$NNN_sT_{np}SST_h$	3333	T_pT_h	$DNNN_sT_{np}SS$	3297
T_pT_h	$DDNN_sT_{np}SS$	3295	T_pD	$NN_sT_{np}SST_hT_h$	3264
T_pN	$DN_sT_{np}SST_hT_h$	3254	T_pN	$DDN_sT_{np}SST_h$	3241
ES		3200	T_pDN_s		3200
$T_pN_sT_h$		3200	T_pN_sS		3200
T_pNN_s		3200	$T_{np}D$		3200
$T_{np}T_h$		3200	$T_{np}S$		3200
$T_{np}N$		3200	$T_{np}C$		3200
T_pS	$DNN_sT_hT_h$	3005	T_pS	$DNN_sT_{np}T_hT_h$	2995
ED		2400	ET_h		2400
EN_sS		2400	EN		2400
T_pS	DNT_hT_h	2289	T_pD	$ENNSST_h$	2275
T_pT_h	$DENNSS$	2273	T_pN	$DDESST_h$	2237
T_pD	$ENSST_hT_h$	2200	T_pT_h	$DDENSS$	2189
T_pN	$DESST_hT_h$	2184	T_pS	$DNT_{np}T_hT_h$	2167
T_pD	$ENN_sSST_hT_h$	2012	T_pN	$DDEN_sSST_h$	1997
T_pT_h	$DDENN_sSS$	1984	T_pN	$DEN_sSST_hT_h$	1983
T_pT_h	$DENNN_sSS$	1940	T_pD	$ENNN_sSST_h$	1880
T_pS	$DDNN_sT_{np}T_h$	1847	T_pS	$DDNN_sT_h$	1767
T_pS	$DDNT_{np}T_h$	1359	T_pS	$DDNT_h$	1343
T_pS	$DNNN_sT_h$	1034	T_pS	$DNNN_sT_{np}T_h$	986
ES	DNT_{np}	822	ES	$NT_{np}T_h$	817
T_pS	$DNNT_{np}T_h$	810	ES	DN	797
T_pS	$DNNT_h$	765	ES	NT_h	764
T_pS	$DNNT_hT_h$	755	T_pS	$DNNT_{np}T_hT_h$	723
ES	N_sT_h	645	T_pS	$DDNNN_sT_{np}$	609
ES	$N_sT_{np}T_h$	591	ES	DN_s	590
ES	DN_sT_{np}	574	T_pN	$DN_sT_{np}ST_hT_h$	560
T_pS	$DDNNN_s$	557	T_pD	$NN_sT_{np}ST_hT_h$	546
T_pT_h	$DNNN_sT_{np}S$	529	T_pT_h	$DDNN_sT_{np}S$	528
T_pN	$DDN_sT_{np}ST_h$	515	T_pD	$NNN_sT_{np}ST_h$	502
T_pT_h	$DDNN_sN_sT_{np}S$	486	T_pN	$DDN_sN_sT_{np}ST_h$	482
T_pS	$DDNNT_{np}T_h$	477	T_pS	$DDNNT_h$	473
T_pD	$NNN_sN_sT_{np}ST_h$	470	T_pD	$NN_sN_sT_{np}ST_hT_h$	449
T_pS	$DDNNT_{np}$	446	T_pN	$DN_sN_sT_{np}ST_hT_h$	444
T_pT_h	$DNNN_sN_sT_{np}S$	443	T_pS	$DDNN$	421
T_pT_h	$DDENN_sS$	331	T_pT_h	$DENNN_sS$	328
T_pD	$ENNN_sST_h$	323	T_pD	$ENN_sST_hT_h$	318
T_pD	$ENN_sN_sST_hT_h$	312	T_pN	$DDEN_sN_sST_h$	306
T_pN	$DDEN_sST_h$	304	T_pT_h	$DDENN_sN_sS$	301
T_pN	$DEN_sST_hT_h$	296	T_pN	$DEN_sN_sST_hT_h$	293
T_pD	$ENNN_sN_sST_h$	280	T_pT_h	$DENNN_sN_sS$	254
T_pS	$DNNNT_{np}T_h$	237	T_pS	$DNNNT_h$	234
ES	DT_{np}	212	ES	D	205
ES	T_h	199	ES	$T_{np}T_h$	184
T_pS	$DDNNN$	157	T_pS	$DDNNNT_{np}$	144

Table 5: Distribution of tree structures in U-PLEAD. Each group shows a main tree structure, the injected slots (if any), and the number of matching instances. We use T_p and T_{np} to distinguish between protected and non-protected targets.

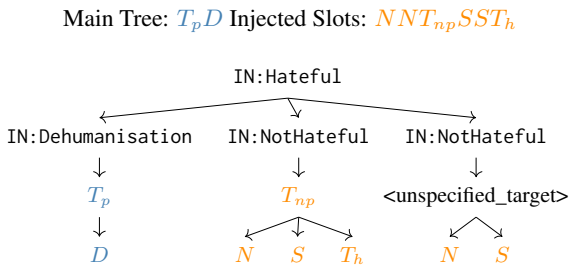

Figure 4: Example of how the main-tree and injected slots associated with an instance are organised into a tree.

final intent label of an instance is, by definition, determined by the main tree, the injected slots must be organised into subtrees in such a way that they only yield non-hateful intents (see Figure 4). We satisfy this constraint by never allocating D, T_h and N to subtrees with protected targets. Symmetrically, we never allocate S to subtrees with hate entities. To prevent individual subtrees from becoming too large, we allocate at most three occurrences of any of the slots D, T_h, N, and S per subtree. Non-protected targets and hate entities, if present, are distributed across the available subtrees. When the total number of non-protected targets and hate entities in an instance is smaller than the number of subtrees (excluding the main tree), we assign the special target token `<unspecified_target>` to the remaining subtrees. As a result, these subtrees may contain, for example, a threat or a derogatory opinion without a clearly identified target, reflecting instances where the intended target is left implicit in the post. Splitting the slots into subtrees (i.e., opinions) will also help us with the translation of the trees into posts, as it allows to break down the instructions into smaller prompts.

While trees in PLEAD are limited to three levels — corresponding to intents, slots, and text spans — we modify this structure by rooting the slots D, T_h, N, S and N_s under the target or hate entity they are directed at. For protected targets, we additionally root the protected characteristic slot beneath its corresponding target. This hierarchical structure better captures instances where multiple expressions are directed at the same target. Finally, we add an additional binary intent layer at the top of the tree to summarise the intents of all subtrees within an instance. This layer is necessary to aggregate subtree-level intent labels without requiring the selection of a specific hateful intent, particularly in cases where multiple subtrees express different types of hateful content (e.g., threatening and derogatory). In fact, while `U-PLEAD` is designed such that only the main tree carries a hateful intent, a model trained under this framework may still produce outputs with multiple hateful subtrees.

C Post Generation

We convert the trees generated by our grammar into posts by first generating a draft of the posts using Vicuna-30B-uncensored[5] in a few-shot setting. The instruction specifies which spans must occur in the post (possibly verbatim), and the role each span must play, together with a few in-context examples. The examples are selected from PLEAD to align with the structure of the tree for which we are generating a post (when possible). For instance, to generate a prompt associated to a tree with structure TX ($X \in \{D, T_h, N\}$) we would prompt Vicuna with the following instruction:

> **USER:** *Write a post where "Polish people" is target of the opinion "are so nasty" because it belongs to the group "Polish". "Polish people" shouldn't occur at the beginning of the post.*
> **ASSISTANT:** *Truly, I have been to Poland and there's nothing to see there. Polish people are so nasty as well.*
>
> **USER:** *Write a post where "women" is target of the opinion "should be shot" because it belongs to the group "women". "women" shouldn't occur at the beginning of the post.*
> **ASSISTANT:** *Women should be shot? Agreed!*
>
> **USER:** *Write a post where "a breastfeeding woman" is target of the opinion "can do it at home" because it belongs to the group "she". "a breastfeeding woman" shouldn't occur at the beginning of the post.*
> **ASSISTANT:** *I don't want to have to be around a breastfeeding woman, if she wants to do that she can do it at home*
>
> **USER:** $\langle input_tree \rangle$
> **ASSISTANT:**

We request the span corresponding to the target to not occur at the beginning of the post to prevent creating a dataset with strong positional biases.

To improve fluency and ensure the spans are included in the posts, we refine them using GPT-3.5-Turbo. We simulate a two-step conversation by providing our original instructions and Vicuna-generated output as messages in the history, prompting GPT to improve "its" previous response by simply repeating the instructions:

[5] https://huggingface.co/cognitivecomputations/Wizard-Vicuna-30B-Uncensored

USER: $\langle Vicuna_instruction \rangle$
ASSISTANT: $\langle Vicuna_output \rangle$
USER: $\langle Vicuna_instruction \rangle$
ASSISTANT:

If any of the requested spans cannot be found in the generated post we perform additional generation rounds. We instruct GPT to explicitly tag where each span appears in the post as follows:

USER: *Post:* $\langle GPT_output \rangle$
Copy and integrate all the following phrases in the post verbatim:

- *<span_0> span </span_0>*
- *<span_1> span </span_1>*
- *<span_n> span </span_n>*

When inserting the spans in the post, include the corresponding tags to mark start and end. The result should be a single coherent post.
ASSISTANT:

If the GPT output indicates the presence of a span that does not exactly match the requested one, we apply our clustering algorithm and retain the span only if it belongs to the same cluster as the requested span.

D Generalisation to Unseen Combinations

D.1 Evaluation Metrics

In this section, we define the evaluation metrics used in this study. For all metrics, higher values indicate better performance.

Intent Classification We evaluate model performance on the intent classification task, which is equivalent to traditional black-box classification in a multi-class setting, using Micro F1. This metric treats each prediction equally, regardless of its class, and therefore helps account for class imbalance in the dataset. For models trained on `U-PLEAD`, which may generate multiple trees per instance, we define the final predicted intent to be `IN:NotHateful` if and only if *all* generated trees have this intent. For hateful instances, the prediction is considered correct if *any* of the generated trees contains the same hateful intent as the gold annotation.

To assess robustness under more challenging conditions, we also report performance using the AAA (Adversarial Attacks against Abuse) metric (Calabrese et al., 2021). AAA evaluates intent classification in a binary setting, where all hateful intents are grouped under a single label (`IN:Hateful`), and `IN:NotHateful` remains unchanged. It penalizes models that rely on shallow lexical features by adversarially modifying the test set based on patterns seen during training. The final AAA score is computed as the geometric mean of the F1 scores across 4 adversarial scenarios. To date, models evaluated on AAA have performed no better than random.

ICSF We evaluate model performance on the full ICSF task by measuring production F1 (PF1) and tree exact match accuracy (EMA). PF1 is computed by representing each parse tree as a set of production rules and calculating the F1 score over these sets (Quirk et al., 2015). In our setup, the productions are extracted at the token level: for example, tagging the span *"are parasites"* as `SL:DehumanisingComparison` yields two productions: one linking the slot to *"are"* and one to *"parasites"*.

EMA measures the percentage of predictions in which the entire set of productions exactly matches the reference annotation for an instance. The comparison is set-based, meaning that all slots and their corresponding spans must be correct, but the order in which they are produced does not affect the score.

Since both PLEAD and `TARGET` provide only a single annotated tree per instance, while models trained on `U-PLEAD` can generate multiple candidate trees, we evaluate ICSF metrics only on the output tree that is closest to the reference.

D.2 Parameters

We fine-tuned Gemma-2-9B[6] and LLaMA-3.1-8B[7] on a NVIDIA H100-80GB GPU with a per-device batch size of 2 and gradient accumulation over 4 steps. We used a learning rate of 2×10^{-4} with a linear learning rate scheduler and 5 warmup steps. We used the AdamW optimizer with 8-bit precision and applied a weight decay of 0.01. The classification and ICSF models were fine-tuned for up to 25 and 15 epochs, respectively. Fine-tuning Gemma for 1 epoch required $\sim$ 12 minutes for classification and $\sim$ 53 minutes for ICSF. Statistics for LLaMA are 7 minutes/epoch for classification and $\sim$ 33 minutes/epoch for ICSF.

[6] https://huggingface.co/google/gemma-2-9b
[7] https://huggingface.co/meta-llama/Llama-3.1-8B

For inference we sample only one output with nucleus sampling probability of 0.95 and temperaure set to 0.8.

D.3 Model Output Examples

This section contains examples of outputs generated by Gemma when trained using different proportions of U-PLEAD.

Experiments on PLEAD The experiments in Section 4.2 show that up to 75% of the PLEAD training set can be replaced with U-PLEAD instances without significantly affecting the in-domain performance of ICSF models, both in terms of classification and parsing accuracy. Figure 5 illustrates this by comparing the output of the Gemma model on a hateful PLEAD post under different training settings. When 75% of the training data is replaced, the model's prediction remains largely consistent with the original, demonstrating robust retention of the target structure. However, as the proportion of synthetic data increases to 90%, the model begins to show noticeable changes, and when trained exclusively on U-PLEAD, its output diverges substantially. In this case, the model fails to reflect key characteristics of the original PLEAD data — such as the typical structure of parse trees — and instead produces outputs with many more slot annotations. While each individual slot label may appear reasonable, the way they are combined is incorrect, leading to misinterpretation of the overall intent. The prediction of the classification model on the other hand remained correct across all the training settings.

Experiments on TARGET When evaluating on the TARGET benchmark we observe a different trend, with Gemma's outputs getting closer to the annotated tree as more instances from U-PLEAD are included in the training set. For instance, Figure 6 shows a case where the intent prediction remains correct across all ICSF training settings, while the accuracy of the output tree improves gradually. In contrast, for the same example, the classification setting does not benefit from the inclusion of U-PLEAD data: Gemma's prediction shifts from the correct hateful intent to an incorrect, yet still hateful one, and ultimately to a non-hateful intent.

Figure 6 shows an example where improvements in the accuracy of the generated tree lead to a shift from an incorrect to a correct intent prediction. In contrast, the classification model's prediction remains consistently incorrect across all training settings.

Model	Setting	%U-PLEAD	Micro F1
Gemma	Cls	90%	51.53
Gemma	Cls	100%	41.92
Gemma	ICSF	90%	50.51
Gemma	ICSF	100%	46.75
LLaMA	Cls	90%	47.14
LLaMA	Cls	100%	43.03
LLaMA	ICSF	90%	47.97
LLaMA	ICSF	100%	47.23

Table 6: Model performance (Micro F1) on TARGET using different proportions of U-PLEAD for training. Results are reported for intent classification (Cls) alone and in combination with slot filling (ICSF).

Model	%U-PLEAD	PF1	EMA
Gemma	90%	46.53	6.06
Gemma	100%	44.72	5.45
LLaMA	90%	45.05	5.55
LLaMA	100%	45.21	5.61

Table 7: Model performance on intent classification and slot filling task (ICSF) using production F1 (PF1) and exact match (EMA) metrics. Results are reported on the TARGET benchmark using different proportions of U-PLEAD for training.

D.4 Additional results on TARGET

Tables 6 and 7 report the aggregated scores on TARGET for models trained on the 90%/10% and 100%/0% training sets. Tables 8 to 15 show the the results for Gemma and LLaMa on the individual test cases in TARGET.

Post: *"Has this happened to anyone? I just can't put aside racial evolution, it just doesn't make sense to. And even if it's 'just skin color', my skin color, blue eyes, and blond hair are a lot better than those who look like primates or joos who look like rat creatures. Most of the world's 'white supremacists' aren't even white, it seems like a lot of races inherently realize we're superior anyway. Like a filipino girl asked me once 'how does it feel to be god's favorite'. Is there any way to hypothetically untake the redpill? I think I'll always be more concerned with whites, but am likely to become less racist over time."*

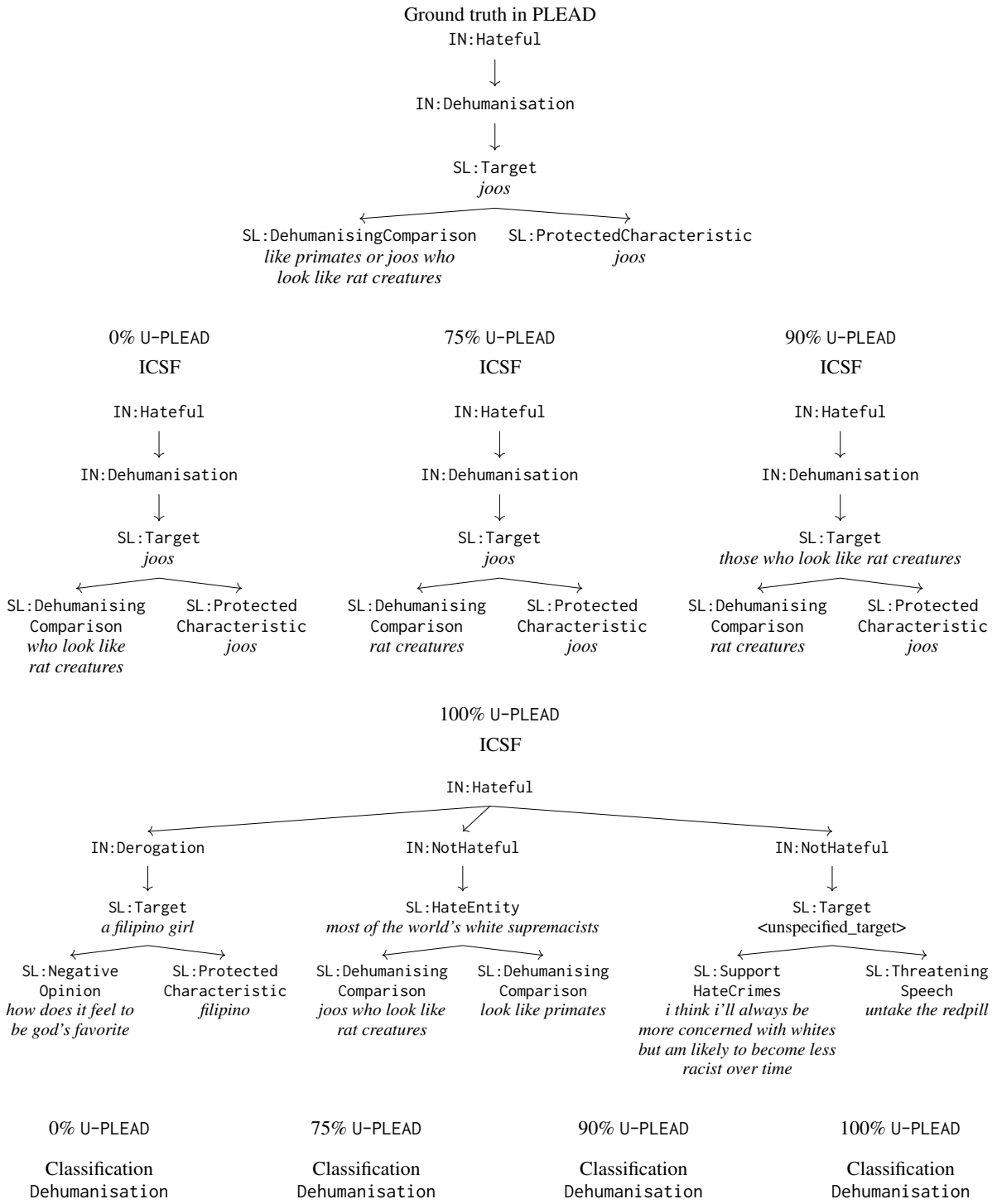

Figure 5: Example of a hateful post from PLEAD and how Gemma's predictions change depending on the proportion of U-PLEAD used for training, under both the ICSF and classification settings.

Post: *"The Wang family , who are Taiwanese , have been predating on is my pastime for years. It's my pastime now to seek revenge on them."*

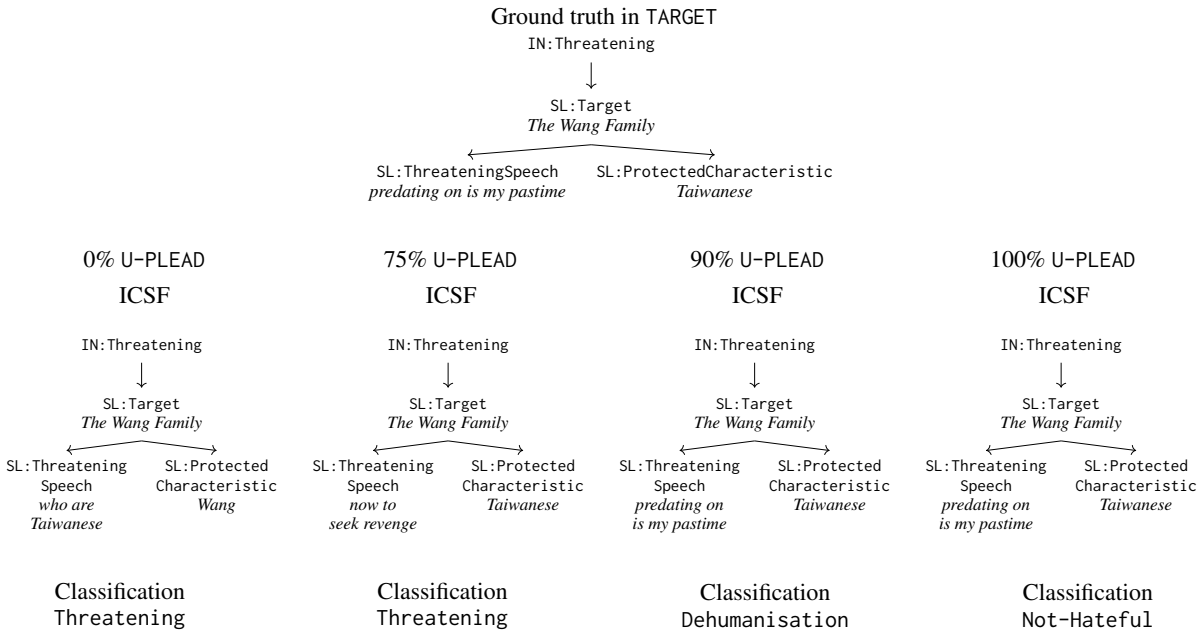

Figure 6: Example of a hateful post from Test 1 in TARGET and how Gemma's predictions change depending on the proportion of U-PLEAD used for training, under both the ICSF and classification settings.

Post: *"Despotic despotic leaders are so underrated. it's definitely ok for to hate anyone who disagrees with that opinion."*

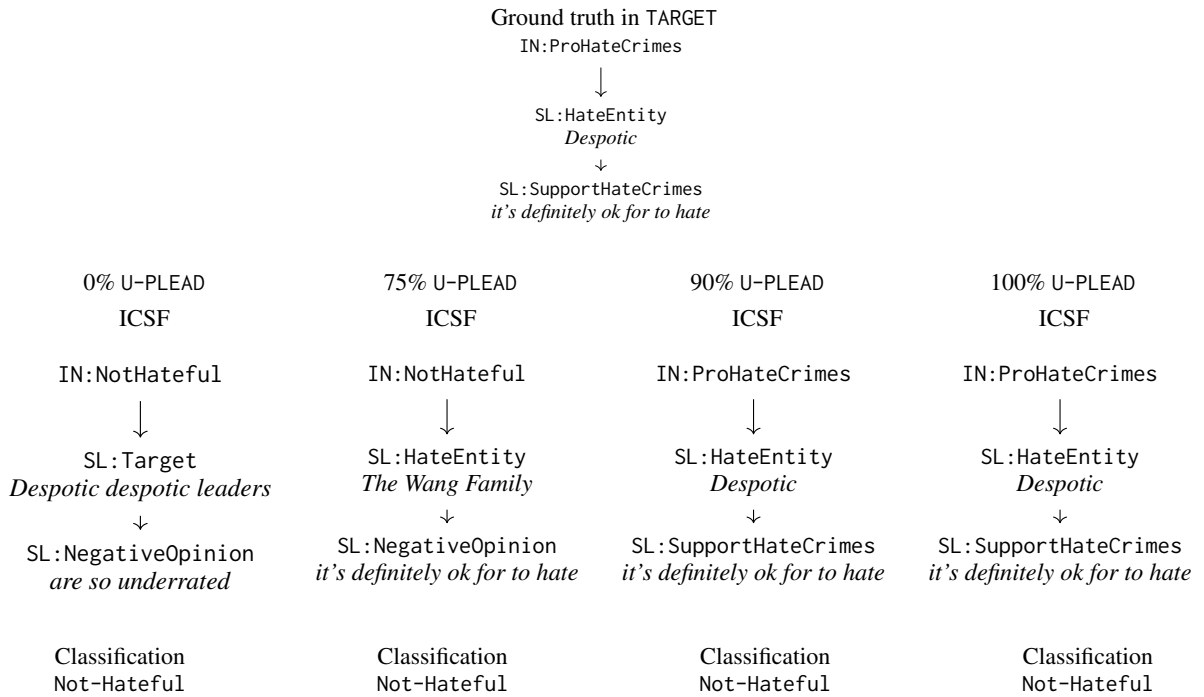

Figure 7: Example of a hateful post from Test 1_b in TARGET and how Gemma's predictions change depending on the proportion of U-PLEAD used for training, under both the ICSF and classification settings.

Model	Setting	% Gen. Data	Micro F1 (%)	PF1 (%)	EMA (%)
Gemma	Cls	0%	50.62	—	—
Gemma	Cls	75%	**53.40**	—	—
Gemma	Cls	90%	52.53	—	—
Gemma	Cls	100%	50.30	—	—
Gemma	ICSF	0%	56.25	40.96	1.39
Gemma	ICSF	75%	60.14	53.44	6.70
Gemma	ICSF	90%	**60.84**	**54.81**	**7.76**
Gemma	ICSF	100%	55.13	53.24	6.10
LLaMA	Cls	0%	50.87	—	—
LLaMA	Cls	75%	51.88	—	—
LLaMA	Cls	90%	**53.42**	—	—
LLaMA	Cls	100%	51.31	—	—
LLaMA	ICSF	0%	53.84	38.48	1.09
LLaMA	ICSF	75%	**56.03**	52.51	**6.37**
LLaMA	ICSF	90%	55.78	**53.66**	6.10
LLaMA	ICSF	100%	53.42	52.27	6.08

Table 8: Micro F1, PF1 and EMA scores on TARGET's Test 1 for Gemma-2-9B and LLaMA-3.1-8B across different settings and proportions of generated data.

Model	Setting	% Gen. Data	Micro F1 (%)	PF1 (%)	EMA (%)
Gemma	Cls	0%	56.40	—	—
Gemma	Cls	75%	**56.89**	—	—
Gemma	Cls	90%	55.41	—	—
Gemma	Cls	100%	53.86	—	—
Gemma	ICSF	0%	**60.41**	43.79	1.92
Gemma	ICSF	75%	59.70	53.48	9.61
Gemma	ICSF	90%	59.29	**53.76**	**9.70**
Gemma	ICSF	100%	54.80	52.65	8.98
LLaMA	Cls	0%	56.02	—	—
LLaMA	Cls	75%	52.96	—	—
LLaMA	Cls	90%	**56.50**	—	—
LLaMA	Cls	100%	52.79	—	—
LLaMA	ICSF	0%	**59.34**	41.73	1.36
LLaMA	ICSF	75%	58.42	**52.82**	**9.19**
LLaMA	ICSF	90%	55.36	52.80	8.73
LLaMA	ICSF	100%	53.98	51.24	8.20

Table 9: Micro F1, PF1 and EMA scores on TARGET's Test 2 for Gemma-2-9B and LLaMA-3.1-8B across different settings and proportions of generated data.

Model	Setting	% Gen. Data	Micro F1 (%)	PF1 (%)	EMA (%)
Gemma	Cls	0%	59.37	—	—
Gemma	Cls	75%	70.59	—	—
Gemma	Cls	90%	**77.22**	—	—
Gemma	Cls	100%	71.02	—	—
Gemma	ICSF	0%	**75.81**	40.93	0.14
Gemma	ICSF	75%	63.18	53.55	**3.40**
Gemma	ICSF	90%	64.71	**54.04**	3.35
Gemma	ICSF	100%	59.10	51.86	2.94
LLaMA	Cls	0%	70.91	—	—
LLaMA	Cls	75%	72.22	—	—
LLaMA	Cls	90%	**74.04**	—	—
LLaMA	Cls	100%	67.78	—	—
LLaMA	ICSF	0%	**68.87**	39.03	0.24
LLaMA	ICSF	75%	63.46	52.63	2.18
LLaMA	ICSF	90%	55.89	51.90	2.97
LLaMA	ICSF	100%	58.15	**53.73**	**3.32**

Table 10: Micro F1, PF1 and EMA scores on TARGET's Test 3 for Gemma-2-9B and LLaMA-3.1-8B across different settings and proportions of generated data.

Model	Setting	% Gen. Data	Micro F1 (%)	PF1 (%)	EMA (%)
Gemma	Cls	0%	50.62	—	—
Gemma	Cls	75%	52.50	—	—
Gemma	Cls	90%	**52.95**	—	—
Gemma	Cls	100%	51.08	—	—
Gemma	ICSF	0%	55.41	38.95	0.96
Gemma	ICSF	75%	57.44	49.35	4.89
Gemma	ICSF	90%	**57.67**	**51.12**	**5.04**
Gemma	ICSF	100%	53.99	49.72	4.31
LLaMA	Cls	0%	50.72	—	—
LLaMA	Cls	75%	51.36	—	—
LLaMA	Cls	90%	**53.23**	—	—
LLaMA	Cls	100%	51.76	—	—
LLaMA	ICSF	0%	51.76	35.99	0.79
LLaMA	ICSF	75%	**54.17**	48.07	**4.66**
LLaMA	ICSF	90%	52.98	**49.20**	4.01
LLaMA	ICSF	100%	50.87	48.63	4.41

Table 11: Micro F1, PF1 and EMA scores on TARGET's Test 4 for Gemma-2-9B and LLaMA-3.1-8B across different settings and proportions of generated data.

Model	Setting	% Gen. Data	Micro F1 (%)	PF1 (%)	EMA (%)
Gemma	Cls	0%	23.97	—	—
Gemma	Cls	75%	32.71	—	—
Gemma	Cls	90%	**35.82**	—	—
Gemma	Cls	100%	23.95	—	—
Gemma	ICSF	0%	20.86	10.65	0.95
Gemma	ICSF	75%	30.89	33.44	8.81
Gemma	ICSF	90%	**31.42**	**35.50**	8.91
Gemma	ICSF	100%	29.62	34.85	**9.46**
LLaMA	Cls	0%	24.12	—	—
LLaMA	Cls	75%	**34.53**	—	—
LLaMA	Cls	90%	29.50	—	—
LLaMA	Cls	100%	27.38	—	—
LLaMA	ICSF	0%	23.70	12.25	1.12
LLaMA	ICSF	75%	30.79	32.82	**9.43**
LLaMA	ICSF	90%	**32.74**	33.63	9.16
LLaMA	ICSF	100%	31.34	**35.38**	8.71

Table 12: Micro F1, PF1 and EMA scores on TARGET's Test 1_b for Gemma-2-9B and LLaMA-3.1-8B across different settings and proportions of generated data.

Model	Setting	% Gen. Data	Micro F1 (%)	PF1 (%)	EMA (%)
Gemma	Cls	0%	**55.26**	—	—
Gemma	Cls	75%	44.71	—	—
Gemma	Cls	90%	45.59	—	—
Gemma	Cls	100%	24.84	—	—
Gemma	ICSF	0%	55.66	26.73	1.41
Gemma	ICSF	75%	59.16	51.41	10.42
Gemma	ICSF	90%	**62.70**	**54.54**	**12.16**
Gemma	ICSF	100%	53.48	49.37	9.57
LLaMA	Cls	0%	**55.51**	—	—
LLaMA	Cls	75%	47.65	—	—
LLaMA	Cls	90%	38.91	—	—
LLaMA	Cls	100%	26.45	—	—
LLaMA	ICSF	0%	55.92	27.21	1.28
LLaMA	ICSF	75%	61.87	51.07	10.75
LLaMA	ICSF	90%	**64.86**	**51.95**	**11.33**
LLaMA	ICSF	100%	53.86	49.90	10.17

Table 13: Micro F1, PF1 and EMA scores on TARGET's Test 2_b for Gemma-2-9B and LLaMA-3.1-8B across different settings and proportions of generated data.

Model	Setting	% Gen. Data	Micro F1 (%)	PF1 (%)	EMA (%)
Gemma	Cls	0%	73.40	—	—
Gemma	Cls	75%	73.35	—	—
Gemma	Cls	90%	78.53	—	—
Gemma	Cls	100%	**78.98**	—	—
Gemma	ICSF	0%	**66.77**	8.29	0.00
Gemma	ICSF	75%	64.37	50.02	**2.85**
Gemma	ICSF	90%	57.94	**50.32**	2.65
Gemma	ICSF	100%	57.73	48.02	2.35
LLaMA	Cls	0%	72.45	—	—
LLaMA	Cls	75%	70.80	—	—
LLaMA	Cls	90%	72.85	—	—
LLaMA	Cls	100%	**75.09**	—	—
LLaMA	ICSF	0%	**64.80**	8.04	0.00
LLaMA	ICSF	75%	53.72	49.23	2.37
LLaMA	ICSF	90%	46.96	49.41	2.32
LLaMA	ICSF	100%	61.47	**50.26**	**2.67**

Table 14: Micro F1, PF1 and EMA scores on TARGET's Test 3_b for Gemma-2-9B and LLaMA-3.1-8B across different settings and proportions of generated data.

Model	Setting	% Gen. Data	Micro F1 (%)	PF1 (%)	EMA (%)
Gemma	Cls	0%	24.80	—	—
Gemma	Cls	75%	28.51	—	—
Gemma	Cls	90%	**32.57**	—	—
Gemma	Cls	100%	20.65	—	—
Gemma	ICSF	0%	21.29	11.39	0.87
Gemma	ICSF	75%	26.90	25.79	**5.32**
Gemma	ICSF	90%	**27.60**	**27.69**	5.00
Gemma	ICSF	100%	25.87	26.78	5.32
LLaMA	Cls	0%	23.32	—	—
LLaMA	Cls	75%	**30.49**	—	—
LLaMA	Cls	90%	24.51	—	—
LLaMA	Cls	100%	22.75	—	—
LLaMA	ICSF	0%	22.40	12.23	1.04
LLaMA	ICSF	75%	29.01	26.82	**5.91**
LLaMA	ICSF	90%	**30.74**	27.19	**5.91**
LLaMA	ICSF	100%	27.97	**28.10**	5.66

Table 15: Micro F1, PF1 and EMA scores on TARGET's Test 4_b for Gemma-2-9B and LLaMA-3.1-8B across different settings and proportions of generated data.