Article

More Than a Model: The Compounding Impact of Behavioral Ambiguity and Task Complexity on Hate Speech Detection

Shuo Xu , Hailiang Wang , Yijun Gao , Yixiang Li , Meng-Ju Kuo [*]

Posted Date: 25 November 2025

doi: 10.20944/preprints202511.1746.v1

Keywords: hate speech detection; behavioral data science; label ambiguity; transformer models; text classification

Article

More Than a Model: The Compounding Impact of Behavioral Ambiguity and Task Complexity on Hate Speech Detection

Shuo Xu [1], Hailiang Wang [2], Yijun Gao [3], Yixiang Li [4] and Meng-Ju Kuo [5,*]

1 Computer Science and Engineering Department, University of California San Diego,La Jolla, CA, USA
2 School of Computer Science, College of Computing, Georgia Institute of Technology, Atlanta, GA, USA
3 Krieger School of Arts and Sciences, Johns Hopkins University, Washington, DC, USA
4 Department of Computer Science, The George Washington University, Washington, DC, USA
5 Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA
* Correspondence: mengjuk@alumni.cmu.edu

Abstract

The automated detection of hate speech is a critical but difficult task due to its subjective, behavior-driven nature, which leads to frequent annotator disagreement. While advanced models (e.g., transformers) are state-of-the-art, it is unclear how their performance is affected by the methodological choice of label aggregation (e.g., 'majority vote' vs. 'unanimous agreement') and task complexity. We conduct a 2x2 quasi-experimental study to measure the compounding impact of these two factors: Labeling Strategy (low-ambiguity 'Pure' data vs. high-ambiguity 'Majority' data) and Task Granularity (Binary vs. Multi-class). We evaluate five models (Logistic Regression, Random Forest, LightGBM, GRU, and ALBERT) across four quadrants derived from the HateXplain dataset. We find that (1) ALBERT is the top-performing model in all conditions, achieving its peak F1-Score (0.8165) on the 'Pure' multi-class task. (2) Label ambiguity is the primary driver of performance loss; ALBERT's F1-Score drops by $\approx$15.6% (from 0.8165 to 0.6894) when trained on noisy 'Majority' data in the multi-class setting. (3) This negative effect is compounded by task complexity, with the performance drop being nearly twice as severe for the multi-class task as for the binary task. A sensitivity analysis confirmed this drop is attributable to data quality (noise), not sample size. We conclude that behavioral label ambiguity is a more significant bottleneck to model performance than model architecture, providing strong evidence for a data-centric approach.

Keywords: hate speech detection; behavioral data science; label ambiguity; transformer models; text classification

1. Introduction

The automated detection of hate speech is a critical challenge for online platform governance and social science research [1]. While deep learning models, particularly transformers, have become the state-of-the-art approach [2], their performance is fundamentally dependent on the quality of the human-annotated data used for training. Unlike objective classification tasks like spam detection [3], identifying hate speech is a subjective, behavior-driven task. Hate speech is nuanced, context-dependent, and culturally specific, leading to significant and unavoidable disagreements among human annotators [4].

This 'behavioral ambiguity' is a core problem in many applied data science fields, including the detection of fake news [5–8] and the monitoring of mental illness on social media [9–11]. A common method for resolving this ambiguity is to aggregate labels using a 'majority vote'. However, this approach can be problematic, as it masks underlying disagreement and treats highly contested labels with the same certainty as those with unanimous agreement. This introduces significant label noise, which can degrade model performance.

While many studies have compared the performance of different model architectures [2], the methodological choice of label aggregation itself is often overlooked as a key experimental variable. It is unclear how this inherent label ambiguity interacts with model complexity. Do more complex models like transformers robustly handle this noise, or are they more sensitive to it than simpler models?

This paper investigates the interplay between label ambiguity, task complexity, and model performance through a 2x2 quasi-experimental study. Specifically, we are guided by three research questions. First, we establish a baseline performance (RQ1) by evaluating classical and deep learning models on 'Pure' (unanimous agreement) data. We then measure how model performance changes (RQ2) when the models are trained instead on 'Majority' (ambiguous, high-noise) data. Finally, we explore whether task complexity (Binary vs. Multi-class) compounds the negative effects of label ambiguity (RQ3).

To answer these questions, we evaluate five models (Logistic Regression, Random Forest, LightGBM, GRU, and ALBERT) across four distinct datasets derived from the HateXplain corpus [4]. Our results demonstrate that while the ALBERT transformer is the top-performing model in all conditions, its performance is (1) highest on 'Pure' data, (2) significantly degraded by 'Majority' data ambiguity, and (3) further compounded by the combination of ambiguity and task complexity. We conclude that data quality, rooted in behavioral agreement, is a more significant bottleneck than model architecture for this task.

2. Methods

To systematically investigate our research questions (RQs) on the impact of label ambiguity and task complexity, we designed and executed a 2×2 quasi-experimental study. The design is quasi-experimental in the sense that factor levels are induced by curating subsets of an existing annotated corpus rather than by randomly assigning instances to conditions. This framework allowed us to isolate and measure the effects of these two key variables on the performance of a diverse range of machine learning (ML) and deep learning (DL) models.

All experiments were conducted in a Python 3 environment, primarily using Google Colab with NVIDIA T4 GPUs. The implementation relied on `pandas` for data management, `scikit-learn` and `lightgbm` for classical ML models, `PyTorch` for the recurrent neural network, and the `Transformers` library for the ALBERT model [12,13].

2.1. Experimental Design

Our methodology is built around the three Research Questions (RQs) introduced in Section 1 that investigate the interplay between model choice, label ambiguity, and task granularity. To test our hypotheses, we structured our study around two primary factors, creating four distinct experimental quadrants.

The first factor is the **Labeling Strategy (Behavioral Dimension)**, which directly tests the impact of annotator disagreement (i.e., behavioral ambiguity). We defined two levels for this factor. The **Pure** condition represents a low-noise, high-agreement scenario using only data with unanimous annotator agreement. Conversely, the **Majority** condition represents a higher-noise, higher-ambiguity scenario using a simple majority vote, which mixes clear and contested labels.

The second factor is **Task Granularity (Task Dimension)**, which tests whether task complexity compounds the effects of label noise. This factor also has two levels: **Binary Classification**, which is a simpler task (aggregating to 'Normal' vs. 'Toxic'), and **Multi-class Classification**, which is the original, more complex task ('Normal' vs. 'Offensive' vs. 'Hatespeech').

This 2×2 design yields four distinct experimental conditions: (1) `Binary-Pure`, (2) `Binary-Majority`, (3) `MultiClass-Pure`, and (4) `MultiClass-Majority`. By training and evaluating an identical suite of models within each quadrant, we can isolate the performance effects attributable to our two main factors and their interaction. We summarize this experimental framework in Table 1.

Table 1. Summary of the 2 × 2 Experimental Design.

	Pure (Unanimous)	**Majority (High Ambiguity)**
Binary Task (`Normal` vs. `Toxic`)	`Binary-Pure` (Low Noise, Simple Task)	`Binary-Majority` (High Noise, Simple Task)
Multi-class Task (`Normal` vs. `Offensive` vs. `Hatespeech`)	`MultiClass-Pure` (Low Noise, Complex Task)	`MultiClass-Majority` (High Noise, Complex Task)

2.2. Data Source and Curation

We selected the HateXplain dataset [4] as the source corpus for our experiments. While the original paper focused on explainability (rationales), this corpus is well-suited to our purpose because it provides the individual, non-aggregated annotations for its 20,148 posts. The original authors reported 'moderate agreement' among annotators; our methodology is explicitly designed to treat this observation as a variable to be tested rather than simply a dataset limitation.

We curated our four experimental datasets by operationalizing the Labeling Strategy factor separately for each task type. For the **Multi-class Tasks**, the `MultiClass-Pure` dataset includes only posts where all three annotators agreed on the specific class (e.g., all three voted 'Normal'). In contrast, the `MultiClass-Majority` dataset includes posts where at least two of the three annotators agreed on a specific class (a simple majority agreement).

For the **Binary Tasks**, we first consolidated the 'Offensive' and 'Hatespeech' labels into a single 'Toxic' class. The `Binary-Pure` dataset was then created by requiring unanimous agreement on this binary split (i.e., all three agreed on 'Normal', or all three agreed on one of the 'Toxic' categories). For the `Binary-Majority` dataset, we first consolidated the labels to binary at the annotator level, resulting in three binary labels per post. We then applied a majority-vote rule on these binary labels, retaining all posts where at least two of the three annotators agreed on the binary outcome.

The resulting class distributions and total sample sizes for each of our four experimental datasets are detailed in Table 2. As shown, all conditions exhibit substantial class imbalance, which informed our choice of evaluation metrics.

Table 2. Total Sample Size and Class Distribution for Each Experimental Dataset.

Experiment	**Label Strategy Logic**	**Normal**	**Offensive**	**Hatespeech**	**Total**
`Binary-Majority`	Majority	7,814	Toxic: 12,334		20,148
`Binary-Pure`	Pure	5,124	Toxic: 8,637		13,761
`MultiClass-Majority`	Majority	7,814	5,480	5,935	19,229
`MultiClass-Pure`	Pure	5,124	1,761	2,960	9,845

2.3. Data Preprocessing and Feature Engineering

A standardized preprocessing pipeline was applied to the raw text of all four datasets to ensure consistency. This pipeline, implemented as a single cleaning function, executed a sequence of four transformations: **Lowercasing** was performed on all text. **Token Replacement** was applied using regular expressions to identify social media-specific entities, which were then replaced with special tokens to preserve context (e.g., URLs became `<URL>`, user mentions became `<USER>`, and hashtags became `<HASHTAG>`). Following this, all remaining **Special Character Removal** of non-alphanumeric and non-whitespace characters occurred. Finally, **Whitespace Normalization** collapsed multiple whitespace characters into a single space, and leading or trailing whitespace was stripped.

Following this cleaning pipeline, we employed two distinct feature engineering strategies tailored to the different model architectures. For the **Classical ML Models** (LR, RF, LGBM), the cleaned text was further processed by removing English stopwords (via NLTK) and applying lemmatization. The resulting text was then vectorized using Term Frequency–Inverse Document Frequency (TF–IDF), capturing both individual words (unigrams) and two-word pairs (bigrams) with an `n_gram_range` of (1, 2). For the **Deep Learning Models** (GRU, ALBERT), we used the cleaned text directly (without stopword removal or lemmatization) to preserve the full sequential context. For the GRU, a custom vocabulary was built, and sequences were tokenized and padded to a fixed length. For ALBERT, the text was fed directly into the pre-trained `Albert-base-v2` tokenizer, with sequences truncated or padded consistent with the pre-trained checkpoint's requirements. In both deep learning cases, the models learned their own latent representations directly from these token sequences during training.

2.4. Model Architectures

To test our research questions, we selected a suite of five models representing a wide spectrum of learning strategies, from interpretable linear models to complex non-linear ensembles and state-of-the-art contextual deep learning models.

2.4.1. Machine Learning Models

We selected three classical ML models to serve as strong baselines. These models are widely used in text classification and are well-suited for high-dimensional, sparse TF–IDF feature matrices. They represent three distinct approaches to classification: linear models, bagging ensembles, and boosting ensembles.

Logistic Regression (LR)

As a robust and highly interpretable linear baseline, we used Logistic Regression [14]. LR models the probability of a discrete outcome by fitting a linear combination of the input features (the TF–IDF vectors) to a logistic (sigmoid) function, generalized using the softmax function for the multi-class task. We used the `scikit-learn` implementation with ℓ_2 (Ridge) regularization to prevent overfitting and manage multicollinearity.

Tree-Based Ensemble Models

We implemented two powerful non-linear ensemble models, grouped by their core ensemble strategy: bagging and boosting. Our bagging model of choice was the Random Forest (RF) [15], which constructs a large number of decorrelated decision trees in parallel through the use of bootstrapped samples and random feature subsets. The final prediction is determined by a majority vote across trees, which effectively reduces variance. Our boosting model of choice was LightGBM (LGBM) [16], a highly efficient and scalable implementation of gradient boosting [17]. Unlike RF's parallel approach, gradient boosting builds trees sequentially, with each new tree trained to correct the residual errors of the existing ensemble. LightGBM utilizes a leaf-wise growth strategy and histogram-based splitting, making it particularly efficient on large, sparse feature spaces.

2.4.2. Deep Learning Models

We selected two deep learning architectures to assess the performance of models that learn representations directly from raw sequential text, rather than from pre-computed TF–IDF features.

Gated Recurrent Unit (GRU)

To represent sequential models, we used a Gated Recurrent Unit (GRU) network [18]. GRUs are a variant of recurrent neural networks (RNNs) that process text token by token, using update and reset gates to control the flow of information, allowing the model to capture long-range dependencies. Our model, implemented in `PyTorch`, consisted of an embedding layer, a multi-layer GRU encoder, and a final linear layer for classification.

ALBERT (A Lite BERT)

To represent the state of the art in contextual modeling, we fine-tuned ALBERT [19]. ALBERT is an efficient variant of the transformer model BERT that uses parameter sharing and factorized embeddings to reduce model size while maintaining high performance. Unlike the sequential GRU, ALBERT processes the entire text sequence in parallel, building deep, bidirectional contextual representations of each token. We fine-tuned the `Albert-base-v2` checkpoint from the Hugging Face `Transformers` library for our classification tasks [20].

2.5. Model Training and Hyperparameter Tuning

A central component of our methodology was ensuring a fair, 'apples-to-apples' comparison between the computationally inexpensive classical models and the more resource-intensive deep learning models. To this end, we enforced a consistent data-splitting and model-selection protocol across all four experimental quadrants. For each quadrant, the data were split into 60% training, 20% validation, and 20% test sets. Splits were stratified by class to preserve distributions, and a fixed random seed ensured that all models used the exact same partition. No model used information from the test set during training or hyperparameter tuning.

All reported scores are from the single held-out 20% test set. The training and tuning process was standardized as follows: All models (LR, RF, LGBM, GRU, ALBERT) were initially trained on the 60% `train` set for a grid or random sample of hyperparameters. Hyperparameter configurations were evaluated using performance on the 20% `validation` set, with **Weighted F1-Score** as the primary selection criterion. The single best configuration per model architecture (i.e., the one achieving the highest validation Weighted F1) was selected.

2.5.1. Hyperparameter Search Strategies

We employed different search strategies for the two categories of models to balance thoroughness and computational cost.

For the **Machine Learning Models** (LR, RF, LGBM), we performed an exhaustive grid search using `GridSearchCV` from `scikit-learn` to identify the optimal combination of hyperparameters. Cross-validation folds were drawn only from the training partition. The specific parameter grids used for the classical models are detailed in Table 3.

Table 3. Hyperparameter Grids for Machine Learning Models (Grid Search).

Model	Hyperparameter	Values Searched
LR	`C`	[0.1, 1, 10]
	`solver`	[`'lbfgs'`, `'saga'`]
	`penalty`	[`'l2'`]
	`class_weight`	[`'balanced'`, None]
RF	`n_estimators`	[50, 100, 200]
	`max_depth`	[None, 10, 20]
	`min_samples_split`	[2, 5, 10]
	`min_samples_leaf`	[1, 2, 4]
	`class_weight`	[`'balanced'`, None]
LGBM	`n_estimators`	[100, 200]
	`learning_rate`	[0.01, 0.1, 0.2]
	`num_leaves`	[31, 63]
	`max_depth`	[None, 10]
	`class_weight`	[`'balanced'`, None]

For the **Deep Learning Models** (GRU, ALBERT), an exhaustive grid search was computationally infeasible. We instead employed a random search of 10 iterations to efficiently explore the hyperparameter space. For each sampled configuration, models were trained on the training set and evaluated on the validation set. The ranges from which hyperparameters were drawn are detailed in Table 4. Both DL models were optimized with standard variants of the Adam optimizer and trained with mini-batches; early stopping based on validation performance was used where applicable to prevent overfitting.

Table 4. Hyperparameter Ranges for Deep Learning Models (Random Search).

Model	Hyperparameter	Ranges Sampled
GRU	`embedding_dim`	$[150, 250]$ (Integer)
	`hidden_dim`	$[256, 768]$ (Integer)
	`lr` (learning rate)	$[10^{-4}, 10^{-3}]$ (Log-uniform)
	`epochs`	$[5, 10]$ (Integer)
ALBERT	`lr` (learning rate)	$[10^{-5}, 3 \times 10^{-5}]$ (Log-uniform)
	`epochs`	$[3, 6]$ (Integer)
	`dropout`	$[0.0, 0.25]$ (Uniform)

2.6. Evaluation Metrics and Confidence Intervals

Given the substantial class imbalance in all quadrants (Table 2), we focused on metrics that account for class support. For both binary and multi-class tasks, we report the **Weighted Precision**, **Weighted Recall**, and **Weighted F1-Score**, computed as the support-weighted average of class-specific values. We also report the **AUC** for binary tasks and **Macro-AUC** for multi-class tasks, where Macro-AUC is defined as the unweighted mean of one-vs-rest ROC AUCs across classes.

To quantify uncertainty, we computed 95% **confidence intervals (CIs)** for F1 and AUC via a non-parametric bootstrap of the test set, stratified by class. For each model and quadrant, we generated $B = 1000$ bootstrap resamples of the test data (with replacement), refit the metric on each resample, and reported the 2.5th and 97.5th percentiles of the resulting empirical distribution as the CI.

All results reported in Section 3 correspond to the held-out 20% test set.

3. Results

This section presents the empirical findings from our 2×2 experimental framework. The results are organized by our research questions, first establishing the baseline performance in low-noise 'Pure' conditions (RQ1), and then analyzing the impact of label ambiguity and task complexity (RQ2 & RQ3). All reported scores are from the held-out 20% test set.

3.1. RQ1: Baseline Performance on 'Pure' Data

Our first research question sought to establish baseline model performance under ideal, low-noise conditions. To answer this, we evaluated all five models on the 'Pure' datasets, where all annotators were in unanimous agreement.

3.1.1. Performance on the `Binary-Pure` Task

In the simpler binary classification task (N=13,761), the deep learning models demonstrated a clear advantage over the classical TF-IDF-based models. Table 5 details the performance of all models. The **ALBERT transformer model** was the clear top performer, achieving a Weighted F1-Score of 0.8126 (95% CI $[0.7980, 0.8270]$). The GRU model also performed strongly with an F1-Score of 0.7877 (95% CI $[0.7732, 0.8046]$), notably outperforming the best classical model, Random Forest (F1: 0.7356). The non-overlapping confidence intervals suggest a clear performance gap between the deep learning models and the classical models.

Table 5. Model Performance on the `Binary-Pure` Test Set. Metrics are weighted averages for Precision and Recall. Best scores are in bold.

	Weighted Metrics			
Model	**Precision**	**Recall**	**F1-Score (95% CI)**	**AUC (95% CI)**
LR	0.71	0.72	0.7138 [0.6975, 0.7309]	0.7791 [0.7622, 0.7963]
RF	0.74	0.74	0.7356 [0.7179, 0.7522]	0.7860 [0.7684, 0.8035]
LGBM	0.73	0.74	0.7323 [0.7159, 0.7485]	0.7929 [0.7754, 0.8093]
GRU	0.79	0.79	0.7877 [0.7732, 0.8046]	0.8619 [0.8481, 0.8751]
ALBERT	0.81	0.82	**0.8126 [0.7980, 0.8270]**	**0.8835 [0.8694, 0.8965]**

3.1.2. Performance on the `MultiClass-Pure` Task

We observed a similar pattern in the more granular 'MultiClass-Pure' task (N=9,845), as detailed in Table 6. ALBERT once again achieved the highest performance across all metrics, with a Weighted F1-Score of 0.8165 (95% CI $[0.7997, 0.8338]$) and a Macro-AUC of 0.9226 (95% CI $[0.9118, 0.9328]$). Interestingly, the GRU model (F1: 0.7367) did not perform as well in this multi-class setting, falling behind the classical ensemble models. The best classical model was Random Forest (F1: 0.7730). The confidence intervals for ALBERT do not overlap with any other model, indicating a clear and substantial performance advantage.

Table 6. Model Performance on the `MultiClass-Pure` Test Set. Metrics are weighted averages for Precision and Recall. Best scores are in bold.

	Weighted Metrics			
Model	**Precision**	**Recall**	**F1-Score (95% CI)**	**Macro-AUC (95% CI)**
LR	0.77	0.76	0.7646 [0.7451, 0.7840]	0.8796 [0.8665, 0.8928]
RF	0.77	0.77	0.7730 [0.7549, 0.7921]	0.8875 [0.8746, 0.9001]
LGBM	0.78	0.76	0.7673 [0.7480, 0.7860]	0.8792 [0.8658, 0.8915]
GRU	0.74	0.73	0.7367 [0.7177, 0.7547]	0.8497 [0.8351, 0.8650]
ALBERT	**0.82**	**0.81**	**0.8165 [0.7997, 0.8338]**	**0.9226 [0.9118, 0.9328]**

These baseline results from both 'Pure' quadrants confirm our first hypothesis: under ideal, high-agreement data conditions, the deep contextual representations learned by the transformer model (ALBERT) provide a measurable performance advantage over both sequential (GRU) and TF-IDF-based classical models.

3.2. RQ2: The Impact of Label Ambiguity on Performance

Our second research question addresses the effect of behavioral ambiguity (label noise) on model performance. To answer this, we evaluated all models on the 'Majority' datasets and compared their performance to the 'Pure' (low-noise) baselines.

3.2.1. Performance on the `Binary-Majority` Task

We first analyzed the `Binary-Majority` task (N=20,148). As shown in Table 7, the introduction of label ambiguity resulted in a universal and clear performance degradation for every model compared to the 'Pure' condition (Table 5).

ALBERT remained the top-performing model with a Weighted F1-Score of 0.7447 (95% CI $[0.7304, 0.7576]$). However, this represents a substantial $\approx$ 8.4% relative decrease from its 0.8126 F1-Score on the 'Pure' data. The GRU model (F1: 0.7196) also saw a significant drop from its 'Pure' performance (F1: 0.7877) but remained the clear second-best performer, outperforming all classical models.

Table 7. Model Performance on the `Binary-Majority` Test Set. Metrics are weighted averages for Precision and Recall. Best scores are in bold.

	Weighted Metrics			
Model	**Precision**	**Recall**	**F1-Score (95% CI)**	**AUC (95% CI)**
LR	0.66	0.67	0.6637 [0.6488, 0.6784]	0.7154 [0.7005, 0.7310]
RF	0.69	0.70	0.6887 [0.6746, 0.7035]	0.7279 [0.7124, 0.7425]
LGBM	0.68	0.69	0.6810 [0.6668, 0.6957]	0.7365 [0.7216, 0.7520]
GRU	0.72	0.72	0.7196 [0.7060, 0.7337]	0.7941 [0.7801, 0.8082]
ALBERT	**0.74**	**0.75**	**0.7447 [0.7304, 0.7576]**	**0.8272 [0.8141, 0.8400]**

3.2.2. Performance on the `MultiClass-Majority` Task

A similar, and even more pronounced, drop in performance was observed in the `MultiClass-Majority` task (N=19,229), as shown in Table 8.

Again, ALBERT remained the best performer, but its F1-Score fell to 0.6894 (95% CI [0.6742, 0.7037]). This is a massive $\approx$ 15.6% relative decrease from its 0.8165 score in the clean 'MultiClass-Pure' task (Table 6). In this high-noise environment, the classical models (LR, RF, LGBM) were all clustered around $\approx$ 0.65 F1, while the GRU model (F1: 0.5984) struggled significantly, performing the worst of all models.

Table 8. Model Performance on the `MultiClass-Majority` Test Set. Metrics are weighted averages for Precision and Recall. Best scores are in bold.

		Weighted Metrics		
Model	**Precision**	**Recall**	**F1-Score (95% CI)**	**Macro-AUC (95% CI)**
LR	0.66	0.66	0.6518 [0.6368, 0.6676]	0.8106 [0.7995, 0.8218]
RF	0.66	0.66	0.6552 [0.6400, 0.6700]	0.8194 [0.8090, 0.8305]
LGBM	0.65	0.65	0.6526 [0.6376, 0.6681]	0.8138 [0.8030, 0.8243]
GRU	0.60	0.60	0.5984 [0.5821, 0.6132]	0.7566 [0.7446, 0.7680]
ALBERT	**0.69**	**0.69**	**0.6894 [0.6742, 0.7037]**	**0.8472 [0.8375, 0.8571]**

These two experiments confirm our second hypothesis: introducing label ambiguity by using a 'Majority' vote strategy severely degrades the performance of all models, regardless of task complexity.

3.3. RQ3: Analysis of the Compounding Impact of Task Complexity

Our third research question explored whether task complexity (Binary vs. Multi-class) interacts with and worsens the negative effect of label ambiguity. By comparing the results from the four tables presented in RQ1 and RQ2, we can analyze this critical interaction effect.

This analysis reveals two key patterns based on two separate measurement axes. We first measure the 'cost of ambiguity' (Horizontal Comparison) for both tasks by comparing the 'Pure' results (Tables 5 and 6) against the 'Majority' results (Tables 7 and 8) for our best model, ALBERT. For the Binary task, introducing ambiguity caused the F1-Score to drop from 0.8126 to 0.7447, resulting in a loss of 0.0679. However, for the more complex Multi-class task, introducing ambiguity caused the F1-Score to drop from 0.8165 to 0.6894, a significantly larger loss of 0.1271. This difference demonstrates that the performance drop from label noise was almost twice as severe for the more complex multi-class task.

The second axis measures the 'cost of complexity' (Vertical Comparison) for both data quality levels. In the 'Pure' (low-noise) condition, increasing the task complexity from Binary to Multi-class had a negligible impact on ALBERT's performance (F1-Score was stable at 0.8126 vs. 0.8165). This suggests the ALBERT model is robust to task complexity when the underlying data quality is high. In sharp contrast, in the 'Majority' (high-noise) condition, increasing task complexity caused a significant performance drop, from 0.7447 to 0.6894.

These two analyses confirm our third hypothesis. The negative impact of label ambiguity is compounded by task complexity: while the transformer model is robust to complexity on clean data, it is highly sensitive to complexity when data quality is low. This interaction effect is a key finding of our 2×2 experiment.

3.4. Sensitivity Analysis: Disentangling Data Quality from Sample Size

A primary critique of our findings could be that the performance gap between the 'Pure' and 'Majority' conditions is confounded by sample size. For instance, the `MultiClass-Pure` dataset (N=9,845) is smaller than the `MultiClass-Majority` dataset (N=19,229). One could argue that the 'Pure' models performed better simply because the dataset was smaller and easier to model.

To rule out this confound, we conducted a sensitivity analysis. We created a `Majority-Downsampled` dataset by taking a random subsample of the `MultiClass-Majority` data to match the exact size of the `MultiClass-Pure` dataset (N=9,845). We then trained and evaluated our key models on this new dataset, which had low quality (high noise) but an identical sample size to the 'Pure' data.

The implication of this analysis was clear. Model performance on the random sampled `Majority-Downsampled` dataset was nearly identical to the performance observed on the full

`MultiClass-Majority` dataset. For instance, ALBERT achieved an F1-Score of approximately 0.69 on the downsampled set, which is dramatically lower than its 0.8165 score achieved on the `MultiClass-Pure` set. This analysis provides strong evidence that the observed performance degradation is genuinely attributable to data quality (specifically, label ambiguity), rather than being an artifact of differences in sample size.

3.5. Summary of Findings

To summarize the entire 2 × 2 experiment, Table 9 presents the best-performing model (based on Weighted F1-Score) from each of the four experimental quadrants. The results clearly show a **'performance cliff'**.

While ALBERT was the top-performing model in all four conditions, its performance was highest in the `MultiClass-Pure` condition (F1: **0.8165**), suggesting that the model benefits from fine-grained, high-quality labels. Performance drops significantly when label ambiguity is introduced ('Majority' data), and drops further still when task complexity is added on top of that ambiguity (the `MultiClass-Majority` condition), confirming our core hypotheses.

Table 9. Summary of Best Weighted F1-Scores (ALBERT) Across the 2x2 Experimental Design.

	Factor 1: 'Pure' Data (Low Ambiguity)	**Factor 1: 'Majority' Data** (High Ambiguity)
Factor 2: Binary Task	0.8126	0.7447
Factor 2: Multi-class Task	**0.8165**	0.6894

4. Discussion

The results from our 2 × 2 experimental design provide clear answers to our research questions and offer significant insights into the impact of behavioral ambiguity on hate speech detection. Our discussion is organized into three parts: first, we address the critical confound of sample size; second, we interpret the main findings in the context of our hypotheses; and third, we discuss the broader implications and avenues for future work.

4.1. Addressing the Confound of Sample Size (Sensitivity Analysis)

A primary critique of our findings could be that the performance gap between the 'Pure' and 'Majority' conditions is confounded by sample size. For instance, the `MultiClass-Pure` dataset (N=9, 845) is much smaller than the `MultiClassMajority` dataset (N=19, 229). One might argue that the 'Pure' models performed better simply because the dataset was smaller and easier to model.

To rule out this confound, we conducted a sensitivity analysis. We created a `Majority-Downsampled` dataset by taking a random subsample of the `MultiClass-Majority` data to match the exact size of the `MultiClass-Pure` dataset (N=9, 845). We then trained and evaluated our key models on this new dataset, which had *low quality (high noise)* but an *identical sample size* to the 'Pure' data.

The indications of this sensitivity analysis were clear: model performance on the random sampled `Majority-Downsampled` dataset was nearly identical to the performance observed on the full `MultiClass-Majority` dataset. For example, ALBERT achieved an F1-Score of approximately 0.69 on the downsampled set, which is dramatically lower than its 0.8165 score achieved on the `MultiClass-Pure` set. This analysis provides strong evidence that the performance gap is genuinely attributable to **data quality (i.e., label ambiguity)**, not the artifact of sample size.

4.2. Interpretation of Main Findings

With the sample size confound addressed, we can interpret the main results from our 2 × 2 experiment.

4.2.1. RQ1: Transformers Excel on 'Pure' Behavioral Data

Our results from the 'Pure' conditions (Tables 5 and 6) serve as a clear baseline. The ALBERT model's superior performance ($F_1 > 0.81$) confirms that when the behavioral signal is unambiguous (i.e., unanimous annotator agreement), modern transformers can effectively learn and generalize. The non-overlapping confidence intervals suggest this advantage is not trivial. This finding aligns with the broader consensus that transformers are the state-of-the-art for most NLP tasks.

4.2.2. RQ2: Label Ambiguity is the Primary Driver of Performance Loss

The most significant finding of this study is the dramatic performance drop when moving from 'Pure' to 'Majority' data. In the multi-class task (Table 8), ALBERT's F1-Score plummeted from 0.8165 to 0.6894, a $\approx$ 15.6% relative decrease. This demonstrates that the 'noise' introduced by simple majority-vote labeling—which mixes clear signals with highly contested ones—is the single greatest bottleneck to model performance. This confirms our second hypothesis that label ambiguity severely harms model efficacy.

4.2.3. RQ3: Task Complexity Compounds the Effect of Ambiguity

Our third hypothesis was that task complexity would compound the negative effect of noise. The summary in Table 9 clearly supports this by highlighting an interaction effect between the two factors.

When data was 'Pure' (Low Noise), moving from a simple binary task (F_1 : 0.8126) to a complex multi-class task (F_1 : 0.8165) had a negligible impact. This suggests the ALBERT model is robust to task complexity when the data is clean. However, when data was 'Majority' (High Noise), the story changed dramatically. Performance on the simple binary task (F_1 : 0.7447) was already degraded, but when task complexity was added, performance fell even further to 0.6894. This demonstrates a clear interaction: the models are capable of handling complexity, but not when the training data is also ambiguous and noisy. The negative impact of ambiguity is significantly worsened when combined with high task complexity.

4.3. Implications and Future Work

Our findings have significant implications for the field of behavioral data science and hate speech detection.

The results constitute a powerful argument for the 'Data-Centric' approach to AI. The bottleneck for this behavioral task was clearly not the model architecture (ALBERT was superior) but the quality of the data. Consequently, simply building a bigger model will not solve a problem rooted in ambiguous human-annotated labels.

Furthermore, our results show that the widespread practice of using 'majority vote' is a problematic aggregation strategy. It masks inherent behavioral ambiguity and creates a noisy signal that degrades even the most advanced models. Researchers should be transparent about their aggregation methods and, when possible, use 'Pure' (unanimous) subsets for more reliable model benchmarking.

For future work on ambiguity, researchers should treat annotator disagreement as a *feature*, not as noise to be filtered. Methods from psychology and psychometrics, such as modeling individual annotator biases or using 'think-aloud' protocols, could provide a deeper understanding of why disagreement occurs [21]. Furthermore, developing models that can explicitly learn from ambiguous or 'soft' labels could be a fruitful avenue for progress in handling behaviorally complex tasks.

4.4. Conclusions

This study conducted a 2×2 quasi-experimental analysis to measure the impact of behavioral label ambiguity and task complexity on hate speech detection. Our findings demonstrate that while transformer models (ALBERT) are the top performers, their efficacy is fundamentally dependent on data quality. We found that the ambiguity introduced by 'majority vote' labeling is the primary driver of performance degradation, and that this negative effect is significantly compounded when the task is also complex (multi-class). This work provides strong evidence that for subjective, behavior-driven tasks, investing in data quality and resolving annotator disagreement is more critical for progress than model architecture alone.

References

1. Mansur, Z.; Omar, N.; Tiun, S. Twitter Hate Speech Detection: A Systematic Review of Methods, Taxonomy Analysis, Challenges, and Opportunities. *IEEE Access* **2023**, *11*, 16226–16249. https://doi.org/10.1109/ACCESS.2023.3239375.
2. Malik, J.S.; Qiao, H.; Pang, G.; et al.. Deep learning for hate speech detection: a comparative study. *International Journal of Data Science and Analytics* **2025**, *20*, 3053–3068. https://doi.org/10.1007/s41060-024-00650-6.
3. Xu, S.; Ding, Z.; Wei, Z.; Yang, C.; Li, Y.; Chen, X.; Wang, H. A Comparative Analysis of Deep Learning and Machine Learning Approaches for Spam Identification on Telegram. In Proceedings of the 2025 6th International Conference on Computer Communication and Network Security, 2025.
4. Mathew, B.; Saha, P.; Yimam, S.M.; Biemann, C.; Goyal, P.; Mukherjee, A. HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. In Proceedings of the The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21), 2021, pp. 14867–14875.
5. Tanvir, A.A.; Mahir, E.M.; Akhter, S.; Huq, M.R. Detecting Fake News using Machine Learning and Deep Learning Algorithms. In Proceedings of the 2019 7th International Conference on Smart Computing & Communications (ICSCC), 2019, pp. 1–5. https://doi.org/10.1109/ICSCC.2019.8843612.
6. Alghamdi, J.; Lin, Y.; Luo, S. Towards COVID-19 fake news detection using transformer-based models. *Knowledge-Based Systems* **2023**, *274*, 110642. https://doi.org/10.1016/j.knosys.2023.110642.
7. Shah, S.; Patel, S. A comprehensive survey on fake news detection using machine learning. *Journal of Computer Science* **2025**, *21*, 982–990. https://doi.org/10.3844/jcssp.2025.982.990.
8. Tian, Y.; Xu, S.; Cao, Y.; Wang, Z.; Wei, Z. An Empirical Comparison of Machine Learning and Deep Learning Models for Automated Fake News Detection. *Mathematics* **2025**, *13*, 2086. https://doi.org/10.3390/math13132086.
9. Zhang, Y.; Wang, Z.; Ding, Z.; Tian, Y.; Dai, J.; Shen, X.; Liu, Y.; Cao, Y. Employing Machine Learning and Deep Learning Models for Mental Illness Detection. *Computation* **2025**, *13*, 186. https://doi.org/10.3390/computation13080186.
10. Ding, Z.; Wang, Z.; Zhang, Y.; Cao, Y.; Liu, Y.; Shen, X.; Tian, Y.; Dai, J. Trade-offs between machine learning and deep learning for mental illness detection on social media. *Scientific Reports* **2025**, *15*, 14497. https://doi.org/10.1145/1143844.1143874.
11. Cao, Y.; Dai, J.; Wang, Z.; Zhang, Y.; Shen, X.; Liu, Y.; Tian, Y. Machine learning approaches for depression detection on social media: A systematic review of biases and methodological challenges. *Journal of Behavioral Data Science* **2025**, *5*. https://doi.org/10.35566/jbds/caoyc.
12. Lan, G.; Inan, H.A.; Abdelnabi, S.; Kulkarni, J.; Wutschitz, L.; Shokri, R.; Brinton, C.G.; Sim, R. Contextual Integrity in LLMs via Reasoning and Reinforcement Learning, 2025. https://doi.org/10.48550/arXiv.2506.04245.
13. Lan, G.; Zhang, S.; Wang, T.; Zhang, Y.; Zhang, D.; Wei, X.; Pan, X.; Zhang, H.; Han, D.J.; Brinton, C.G. MaPPO: Maximum a Posteriori Preference Optimization with Prior Knowledge, 2025. https://doi.org/10.48550/arXiv.2507.21183.
14. Hosmer, D.W.; Lemeshow, S. *Applied Logistic Regression*, 2nd ed.; John Wiley & Sons: New York, NY, 2000.
15. Breiman, L. Random Forests. *Machine Learning* **2001**, *45*, 5–32. https://doi.org/10.1023/A:1010933404324.
16. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Proceedings of the Advances in Neural Information Processing Systems 30 (NeurIPS 2017); Guyon, I.; et al.., Eds. Curran Associates, Inc., 2017, pp. 3149–3157.
17. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* **2001**, *29*, 1189–1232. https://doi.org/10.1214/aos/1013203451.
18. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, oct 2014; pp. 1724–1734. https://doi.org/10.3115/v1/D14-1179.
19. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A lite BERT for self-supervised learning of language representations, 2020. https://doi.org/10.48550/arXiv.1909.11942.

20. Rao Killi, C.B.; Balakrishnan, N.; Rao, C.S. A Novel Approach for Early Rumour Detection in Social Media Using ALBERT. *International Journal of Intelligent Systems and Applications in Engineering* **2024**, *12*, 259–265.
21. Ge, J. Technologies in Peace and Conflict: Unraveling the Politics of Deployment. *International Journal of Research Publication and Reviews (IJRPR)* **2024**, *5*, 5966–5971. https://doi.org/10.55248/gengpi.5.0524.1273.