NIT-Agartala-NLP Team at Touché: Detecting and Classifying Fallacies in Reddit Arguments Using Ensemble Methods

Touché at CLEF 2026

Shivam Singh[1,*,†], Manish Kumar[1,†] and Anupam Jamatia[1,†]

[1]National Institute of Technology Agartala, Tripura, India

Abstract

Automated fallacy detection plays a crucial role in identifying flawed reasoning and manipulative argumentation in online discourse. This paper presents the NIT-Agartala-NLP Team's submission to the CLEF 2026 Touché Shared Task on Fallacy Detection. The task involves arguments extracted from Reddit comments along with their source texts. The provided dataset consists of 938 training instances and 233 test instances with a balanced label distribution. The shared task includes two subtasks: (i) Subtask-1 — Fallacy Detection, a binary classification problem to distinguish fallacious from non-fallacious arguments, and (ii) Subtask-2 — Fallacy Classification, a multi-class task to identify the specific fallacy type. For Subtask-1, we explored a stacking ensemble with Logistic Regression, Decision Tree, and Random Forest as base learners and Logistic Regression as the meta-learner. For Subtask-2, we used a Bagging ensemble of Logistic Regression classifiers. Both approaches rely on TF-IDF vector representations of the input features and were evaluated using 10-fold cross-validation during development. On the official test set, our submitted systems achieved macro-averaged precision, recall, and F1-scores of 0.881, 0.876, and 0.875 for Subtask-1, and 0.730, 0.812, and 0.766 for Subtask-2, respectively.

Keywords

Fallacy Detection, Fallacy Classification, Argument Mining, Computational Argumentation, Ensemble Learning, TF-IDF, Binary Classification, Multi-class Classification, Reddit, Natural Language Processing

1. Introduction

With the rapid growth of the internet and social media, the spread of fallacious reasoning in online discourse has become a pressing concern, making automated fallacy detection an increasingly important task in Natural Language Processing (NLP). Given an argument consisting of a premise and a claim, the goal is to determine whether the reasoning is fallacious — that is, whether the claim does not logically follow from the premise. Beyond merely detecting the presence of a fallacy, it is equally important to identify its specific type, as this helps readers understand precisely what causes the reasoning to break down.

Fallacy detection and classification serve a broader purpose in NLP by automating the process of evaluating whether an argument genuinely supports the claim it intends to make. This is particularly relevant in the context of online platforms such as Reddit, where arguments are often informal, emotionally charged, and prone to flawed reasoning. Detecting a fallacy alone is insufficient; knowing *what kind* of fallacy is present provides deeper insight into the nature of the argumentative flaw and aids downstream tasks such as misinformation detection and argument quality assessment.

In this work, we developed systems for two subtasks of the shared task. Our contribution explores both classical and modern approaches to fallacy detection and classification. For Subtask-1, we experimented with classical machine learning algorithms and ensemble techniques, along with modern methods including fine-tuning of large language models and prompting-based approaches such as Zero-Shot and

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany

*Corresponding author.

†These authors contributed equally.

shivamcse2k21@gmail.com (S. Singh); manishkrydv4212@gmail.com (M. Kumar); anupamjamatia@gmail.com (A. Jamatia)

Few-Shot learning. For Subtask-2, we focused on classical machine learning and ensemble techniques due to the limited availability of training data. Specifically, we propose a stacking ensemble for Subtask-1 consisting of Logistic Regression, Decision Tree, and Random Forest as base learners with Logistic Regression as the meta-learner. Notably, our classical systems proved competitive with, and in some cases outperformed, modern techniques such as fine-tuned large language models. This highlights that ensemble methods remain effective for low-resource tasks while being fast and computationally efficient.

For Subtask-2 (Fallacy Classification), we employed a Bagging (Bootstrap Aggregating) ensemble of Logistic Regression classifiers. This approach trains multiple instances of the same base learner on different bootstrap samples of the training data and aggregates their predictions through majority voting to identify the specific fallacy type. Both systems rely on TF-IDF representations as input features.

The task is hosted by Touché[1] [1, 2], and system submissions are managed through TIRA[2] [3]. In this paper, we present the NIT-Agartala-NLP Team's submission to the CLEF 2026 Touché Shared Task on Fallacy Detection, which consists of three subtasks. Our team participated only in Subtask-1 and Subtask-2. The remainder of this paper is organized as follows. Section 2 reviews related work in fallacy detection and classification. Section 3 describes the task dataset and its characteristics. Section 4 presents the proposed model architectures. Section 5 details the experimental setup, while Section 6 reports the official results. An error analysis is provided in Section 7, and Section 8 concludes the paper with a discussion of limitations and future research directions.

2. Related Work

Automated fallacy detection and classification remain relatively underexplored areas within the broader field of argument mining and computational argumentation. As this task shares similarities with other NLP classification problems [4], early approaches typically framed it as a binary or multi-class text classification problem using classical machine learning algorithms. Methods such as Logistic Regression [5], Decision Tree [6], and Random Forest [7] have been applied to classify argumentative texts, while ensemble techniques [8] have further improved performance by combining the strengths of multiple base learners. Although classical machine learning models offer better interpretability and perform reasonably well on small datasets, they rely on fixed, hand-crafted feature sets and often struggle to capture deep semantic meaning and contextual patterns in text. The introduction of the Transformer architecture [9] marked a significant shift in NLP. Since then, fine-tuning pre-trained language models [10] has become a dominant approach for tasks including fallacy detection and classification. More recent studies have also explored prompt-based methods such as zero-shot and few-shot learning with large language models (LLMs) [11, 12], which can deliver competitive results without requiring extensive task-specific training data. Similarly, fine-tuning-based approaches [13] have demonstrated strong performance in identifying argumentative fallacies in both structured and unstructured text.

Despite the growing interest in prompt-based and fine-tuning methods, classical machine learning approaches have received comparatively less attention in recent work on fallacy detection. In this paper, we revisit classical ensemble methods and evaluate their effectiveness on the CLEF 2026 Touché Shared Task dataset. By doing so, we hope to provide a useful reference point for understanding where traditional approaches stand relative to more recent neural and prompt-based techniques.

3. Dataset

The dataset for the CLEF 2026 Touché Shared Task was provided by [14] and is originally derived from the informal fallacies dataset introduced by [15]. It consists of English Reddit comments distributed in JSONL format. The training set contains 938 instances and the test set contains 233 instances.

[1] https://touche.webis.de/clef26/touche26-web/fallacy-detection.html
[2] https://www.tira.io/task-overview/fallacy-detection/

Each instance includes a number of fields: *id*, *text_raw*, *text_raw_parent*, *text_raw_title*, *text_base*, *argument_base*, *text_enhanced*, *argument_enhanced*, *fallacy_exists*, *fallacy_type*, *resembles_fallacy*, and *classification*. These fields are described in Table 3.

As shown in Table 1, the training set contains 473 non-fallacious and 465 fallacious instances, corresponding to 50.43% and 49.57% of the data respectively. This near-equal distribution makes the dataset well-suited for Subtask-1. Similarly, Table 2 presents the distribution of the eight fallacy types among the fallacious instances. The proportions range from 10.32% for *population* to 13.98% for *tradition*, confirming a reasonably balanced setup for Subtask-2 as well.

The shared task comprises two subtasks. Subtask-1 (Fallacy Detection) is formulated as a binary classification problem, where each instance is labelled as either *Fallacy* or *Non-Fallacy*. Subtask-2 (Fallacy Type Classification) is a multi-class classification task applied only to fallacious instances, where each instance is assigned one of the following eight fallacy categories: *authority*, *black_white*, *hasty_generalization*, *natural*, *population*, *slippery_slope*, *tradition*, and *worse_problems*.

Table 1
Training dataset label distribution

Label	Count	%
Non-Fallacy	473	50.43
Fallacy	465	49.57
Total	**938**	**100.00**

Table 2
Training dataset fallacy type distribution among fallacious instances

Fallacy Type	Count	%
Tradition	65	13.98
Slippery Slope	62	13.33
Natural	61	13.12
Worse Problems	61	13.12
Authority	59	12.69
Hasty Generalization	55	11.83
Black & White	54	11.61
Population	48	10.32
Total	**465**	**100.00**

4. System Overview

Building on the dataset described in Section 3, we developed multiple systems for both subtasks. For each subtask, we experimented with classical machine learning models, ensemble methods, as well as prompt-based and fine-tuning approaches. The following subsections describe these approaches in detail.

4.1. Subtask 1: Fallacy Detection

For all approaches in this subtask, we represent each instance using three key fields from the dataset: *text_enhanced*, *argument_enhanced.claim*, and *argument_enhanced.supports*. We chose these features

Table 3
Dataset fields and their descriptions.

Field	Description
`id`	Unique identifier for the entry.
`text_raw`	The original Reddit comment.
`text_raw_parent`	The parent comment of `text_raw`.
`text_raw_title`	Title of the thread that `text_raw` originates from.
`text_base`	A self-contained rewrite of `text_raw` that incorporates relevant context from `text_raw_parent` and `text_raw_title`.
`argument_base`	`{claim, supports}` — the argument structure extracted from `text_base`.
`text_enhanced`	A rewrite of `text_base` that explicitly surfaces fallacy and argumentation scheme information.
`argument_enhanced`	`{claim, supports}` — the argument structure extracted from `text_enhanced`.
`fallacy_exists`	Binary label (`0` / `1`) indicating whether the entry contains a fallacy.
`fallacy_type`	The specific type of fallacy present (e.g., `authority`, `slippery_slope`, `hasty_generalization`).
`resembles_fallacy`	The fallacy type this argument most closely resembles. Equals `fallacy_type` for fallacious entries; for non-fallacious entries, identifies the fallacy type the reasoning could plausibly be confused with.
`classification`	`{argument_goal, argument_basis}`

because *text_enhanced* is an enriched rewrite of *text_base* that explicitly surfaces fallacy and argumentation scheme information, while *argument_enhanced* provides the structured claim and supporting premise extracted from that rewrite, as described in Table 3. Together, these fields capture both the contextual narrative and the core argumentative structure of each instance, making them particularly well-suited for fallacy detection. The selected features are concatenated and converted into numerical vectors using TF-IDF vectorization [16].

In the machine learning based approach, we apply Logistic Regression (LR), Decision Tree (DT), and Random Forest (RF) classifiers on the TF-IDF representations to classify each instance as either *Fallacy* or *Non-Fallacy*. Given the relatively small size of the training set (938 instances), we adopted a 10-fold cross-validation strategy. This helps reduce training bias and allows the models to learn more reliable patterns from limited data.

To further improve classification performance, we explored several ensemble techniques [17], including Bagging, Gradient Boosting, Voting, and Stacking, applied over the base learners described above using 10-fold cross-validation on the training data. Among all configurations, a Stacking ensemble (LR+DT+RF→LR) comprising Logistic Regression, Decision Tree, and Random Forest as base learners, with Logistic Regression as the meta-learner, achieved the best performance for Subtask-1 in our development experiments.

For zero-shot classification [18], we used the `bart-large-mnli` model[3], a model pre-trained on

[3]https://huggingface.co/facebook/bart-large-mnli

the Multi-Genre Natural Language Inference (MNLI) task. Since zero-shot learning requires no task-specific training, we directly queried the model using *text_base* as input — a self-contained rewrite of *text_raw* that incorporates relevant context from *text_raw_parent* and *text_raw_title* — and prompted it to classify each instance as fallacious or non-fallacious. In the few-shot setting [19], we again used the `bart-large-mnli` model with *text_base* as input. Unlike zero-shot learning, the prompt here includes a small number of labelled examples alongside the query instance, guiding the model towards the target labels of *Fallacy* or *Non-Fallacy*.

In the fine-tuning approach, we added a classification layer on top of several pre-trained transformer models and fine-tuned them end-to-end on the training data using *text_base* as the input feature. The models we experimented with include `BERT-base-uncased`, `BERT-large-uncased`, `RoBERTa-base`, `DeBERTa-v3-large`, and `nli-DeBERTa-v3-small`.

4.2. Subtask 2: Fallacy Type Classification

Since Subtask-2 is applied exclusively to fallacious instances, the available training data is limited to 465 examples (see Table 1). Given this constraint, we focused solely on machine learning based approaches for this subtask, as fine-tuning large pre-trained models on such a small dataset carries a significant risk of overfitting.

We followed the same feature selection strategy as in Subtask-1, using *text_enhanced*, *argument_enhanced.claim*, and *argument_enhanced.supports* as input features, as described in Table 3. The target label for this subtask is *fallacy_type*, which takes one of eight possible categories. The selected features are concatenated and converted into TF-IDF vectors in the same manner as Subtask-1.

We applied Logistic Regression, Decision Tree, and Random Forest classifiers to the TF-IDF representations for multi-class fallacy type classification. As with Subtask-1, we used 10-fold cross-validation to mitigate the effect of the limited training data and obtain more reliable performance estimates.

For the ensemble approach, we explored Bagging, Gradient Boosting, Voting, and Stacking configurations [17] over the same set of base learners, using 10-fold cross-validation. Among all configurations evaluated, a Bagging (Bootstrap Aggregating) ensemble of Logistic Regression classifiers, which trains multiple instances of the same base learner on different bootstrap samples of training data and aggregates their predictions by majority voting, achieved the best performance for Subtask-2.

5. Experimental Setup

As described in Section 3, the CLEF 2026 Touché Shared Task organisers provided a pre-annotated training dataset of 938 instances and an unannotated test dataset of 233 instances. Given the relatively small size of the training data, we adopted a 10-fold cross-validation strategy for all experiments. This approach helps reduce training bias and provides more reliable performance estimates.

We evaluated all systems using the official primary metric: macro F1-score along with macro-averaged precision and recall. The macro average of precision is calculated by taking the average of precision values computed independently for each class. Similarly, the macro average of recall is the average of recall values calculated independently for each class, and the macro F1-score is the average of the per-class F1-scores. These metrics give equal importance to all classes and provide a balanced measure of the model's performance across different categories.

All experiments were conducted with a fixed random seed of 42 to ensure reproducibility. For both subtasks, the input features were constructed by concatenating three fields— *text_enhanced*, *argument_enhanced.claim*, and *argument_enhanced.supports*—into a single feature matrix using TF-IDF vectorization, as outlined in Section 4. The TF-IDF vectorization was applied with the following hyperparameters: maximum vocabulary size of 5,000 features, n-gram range of (1,2), sublinear term frequency scaling enabled, and a minimum document frequency of 2. Three separate vectorizers were fitted on *text_enhanced*, *argument_enhanced.claim*, and *argument_enhanced.supports*, and the resulting sparse matrices were horizontally concatenated to form the final feature matrix.

For Subtask-1, the Stacking ensemble was implemented with three base learners—Logistic Regression (LR), Decision Tree (DT), and Random Forest (RF)—and Logistic Regression as the meta-learner. The LR base learner used an inverse regularization strength of $C = 1$. The DT base learner was configured with a maximum depth of 10 and a minimum of 5 samples per leaf. The RF base learner used 200 estimators with a maximum depth of 12. The meta-learner used a plain Logistic Regression with default settings.

For Subtask-2, the Bagging ensemble was implemented with 20 estimators, each using a Logistic Regression classifier with a maximum of 1000 iterations as the base learner.

All machine learning experiments were implemented using the Scikit-Learn library[4]. For fine-tuning and prompt-based experiments, we used the Hugging Face Transformers library[5] with PyTorch[6] as the backend. All experiments were run on Google Colab[7].

For the fine-tuning experiments in Subtask-1, we evaluated the following pre-trained models: `BERT-base-uncased`[8], `BERT-large-uncased`[9], `RoBERTa-base`[10], `DeBERTa-v3-large`[11], and `nli-DeBERTa-v3-small`[12]. Each model was fine-tuned using the AdamW optimiser with a learning rate of 2×10^{-5} for up to 10 epochs with early stopping.

Our complete implementation, including data processing scripts and ensemble code, is publicly available at the following repository[13].

6. Results

This section presents the experimental results for both Subtasks, obtained through 10-fold cross-validation on the 938 training instances described in Section 3. We report Precision, recall and F1-Score, and all these metrics are calculated using macro-averaging. Official test result for Subtask-1 and Subtask-2 are shown in Table 8 and Table 9 respectively.

Subtask-1: Fallacy Detection

This section presents the experimental results for both subtasks, obtained through 10-fold cross-validation on the 938 training instances described in Section 3. We report macro-averaged precision, recall, and F1-score for all experiments. The official test results for Subtask-1 and Subtask-2 are shown in Table 8 and Table 9 respectively.

Subtask-1: Fallacy Detection

Table 4 presents the results of all machine learning and ensemble approaches for Subtask-1. Among the individual classifiers, Logistic Regression performs the strongest, achieving an F1-score of 0.839, followed by Random Forest (0.810) and Gradient Boosting (0.803). Decision Tree performs the weakest among individual models, with an F1-score of 0.691, which is expected given its tendency to overfit on small datasets. Among ensemble methods, the Stacking configuration (LR+DT+RF→LR) achieves the highest F1-score of 0.840 in cross-validation.

Although Logistic Regression achieves a very close F1-score of 0.839, we selected the Stacking ensemble as our strongest development model because it offered the best overall balance, including a higher recall of 0.931. This is preferable in fallacy detection, where missing a genuine fallacy (false negative) is more costly than a false positive. Notably, the Voting ensemble (LR+DT+RF) underperforms

[4] https://scikit-learn.org/
[5] https://github.com/huggingface/transformers
[6] https://github.com/pytorch/pytorch
[7] https://colab.research.google.com/
[8] https://huggingface.co/google-BERT/BERT-base-uncased
[9] https://huggingface.co/google-BERT/BERT-large-uncased
[10] https://huggingface.co/FacebookAI/RoBERTa-base
[11] https://huggingface.co/microsoft/DeBERTa-v3-large
[12] https://huggingface.co/cross-encoder/nli-DeBERTa-v3-small
[13] https://anonymous.4open.science/r/CLEF-2026-Touch-Fallacy-detection-and-classification-B074

relative to its individual components, indicating that simple majority voting does not effectively combine these classifiers on this task.

Table 5 reports the results of prompt-based approaches using the `bart-large-mnli` model. Both zero-shot and few-shot learning produce poor results, with F1-scores of 0.35 and 0.33 respectively. These scores are at chance level for a binary classification task, suggesting that the `bart-large-mnli` model, despite its NLI pre-training, struggles to generalise to the nuanced fallacy detection task through prompting alone. The negligible difference between zero-shot and few-shot performance further indicates that adding a small number of examples to the prompt does not meaningfully help the model.

Table 6 presents the results of fine-tuned transformer models. `DeBERTa-v3-large` performs best among all fine-tuned models, achieving an F1-score of 0.74, followed by `RoBERTa-base` (0.73). `BERT-base` and `BERT-large` both achieve 0.65 in F1-score, with `NLI-DeBERTa-v3-small` performing the weakest at 0.63. While fine-tuning outperforms prompt-based approaches by a considerable margin, all fine-tuned models fall short of the best classical machine learning results. This finding is noteworthy and suggests that on small, well-structured datasets with informative TF-IDF features, classical ensemble methods can outperform large pre-trained language models, likely due to the limited training data being insufficient for effective fine-tuning.

The official test result for Subtask-1 is shown in Table 8. Our submitted system for this subtask achieved a macro-averaged precision of 0.881, recall of 0.876, and F1-score of 0.875.

Subtask-2: Fallacy Type Classification

Table 7 presents the results of all machine learning and ensemble approaches for Subtask-2. Overall, the results are considerably higher than those for Subtask-1, which is somewhat expected given the balanced eight-class distribution shown in Table 2 and the availability of semantically rich features such as *text_enhanced* and *argument_enhanced*. The Bagging ensemble of Logistic Regression classifiers achieves the best performance in cross-validation, with an F1-score of 0.9368, and was our submitted system for Subtask-2. Plain Logistic Regression performs a close second with an F1-score of 0.9337, followed by Random Forest with an F1-score of 0.9294. In contrast to Subtask-1, the Stacking ensemble (LR+DT+RF→LR) performs noticeably worse with an F1-score of 0.8766, suggesting that the added complexity of a meta-learner does not help for multi-class fallacy type classification. Decision Tree again performs the weakest among all models with an F1-score of 0.7468, consistent with its performance in Subtask-1. The strong performance of Bagging (LR) across both subtasks underscores the effectiveness of variance-reduction ensemble strategies when working with limited training data.

The official test result for Subtask-2 is shown in Table 9. Our submitted Bagging (LR) system achieved a macro-averaged precision of 0.730, recall of 0.812, and F1-score of 0.766.

Table 4
Machine learning based approach: Precision and F1-score on training data for Subtask-1.

Model	Precision	Recall	F1-Score
Stacking (LR+DT+RF→LR)	**0.766**	**0.931**	**0.840**
Logistic Regression	0.821	0.858	0.839
Bagging (LR)	0.836	0.834	0.835
Random Forest	0.832	0.791	0.810
Gradient Boosting	0.804	0.804	0.803
Bagging (DT)	0.713	0.802	0.755
Voting (LR+DT+RF)	0.715	0.781	0.745
Decision Tree	0.653	0.740	0.691

Table 5
LLM prompting approach: accuracy and F1-score on training data for Subtask-1.

Model	Precision	Recall	F1-Score
Zero-Shot Learning	0.64	0.50	0.35
Few-Shot Learning	0.25	0.50	0.33

Table 6
Fine-tuning of pre-trained language models: Precision, recall and F1-score on training data for Subtask-1.

Model	Precision	Recall	F1-Score
DeBERTa-v3-large	0.78	0.75	0.74
RoBERTa-base	0.74	0.73	0.73
BERT-base	0.71	0.67	0.65
NLI-DeBERTa-v3-small	0.65	0.64	0.63
BERT-large	0.68	0.66	0.65

Table 7
Machine learning based approach: Precision, Recall and F1-score on training data for Subtask-2.

Model	Precision	Recall	F1-Score
Bagging (LR)	**0.9493**	**0.9393**	**0.9368**
Logistic Regression	0.9439	0.9368	0.9337
Random Forest	0.9364	0.9320	0.9294
Stacking (LR+DT+RF→LR)	0.8987	0.8786	0.8766
Gradient Boosting	0.8796	0.8663	0.8618
Voting (LR+DT+RF)	0.8613	0.8420	0.8393
Bagging (DT)	0.8436	0.8152	0.8107
Decision Tree	0.7904	0.7471	0.7468

Table 8
Test result of Subtask-1

Model	Precision	Recall	F1-Score
Stacking (LR+DT+RF→LR)	**0.881**	**0.876**	**0.875**

7. Error Analysis

To gain deeper insight into the strengths and limitations of our proposed systems, we conducted a detailed error analysis on the training data using the 10-fold cross-validation results reported in Section 6. The analysis combines quantitative evaluation through macro-averaged precision, recall, and F1-score with qualitative examination of model predictions.

Subtask-1: Fallacy Detection

Figure 1 shows the confusion matrix for the best Subtask-1 system — Stacking (LR+DT+RF→LR). Out of 473 non-fallacious instances, 339 are correctly classified as *No Fallacy*, while 134 are incorrectly predicted as *Fallacy*, giving a false positive rate of approximately 28.3%. On the other hand, out of 465 fallacious

Table 9
Test result of Subtask-2

Model	Precision	Recall	F1-Score
Bagging (LR)	**0.730**	**0.812**	**0.766**

instances, 433 are correctly identified as *Fallacy*, with only 32 misclassified as *No Fallacy*, yielding a false negative rate of approximately 6.9%. This asymmetry suggests that the model is more conservative in labelling instances as non-fallacious — it is considerably better at catching actual fallacies than at avoiding false alarms. In a practical setting, this bias towards detecting fallacies may be preferable, as missing a genuine fallacy (false negative) is arguably more costly than flagging a non-fallacious argument for review (false positive).

Subtask-2: Fallacy Type Classification

Figure 2 shows the confusion matrix for the best Subtask-2 system — Bagging (LR). The diagonal is strongly dominant across all eight fallacy types, confirming that the model classifies the majority of instances correctly. Notably, *slippery_slope* (62/62), *worse_problems* (61/61), and *natural* (59/61) achieve near-perfect classification, indicating that these fallacy types have sufficiently distinct linguistic patterns that TF-IDF features can reliably capture. In contrast, *population* shows the most confusion, with 3 instances misclassified as *authority* and 2 as *hasty_generalization*, which is consistent with its being the smallest class (48 instances, 10.32%) in the training data as shown in Table 2. The *black_white* class is entirely misclassified (all 0 on the diagonal), which is a notable failure and warrants attention. This is likely due to the fact that *black_white* has only 54 training instances and may share surface-level linguistic features with other fallacy types, making it harder for TF-IDF representations to distinguish. This is a clear limitation of the current approach and suggests that richer semantic features or data augmentation strategies may be needed for this class in future work.

The high overall F1-macro of 0.9368 for Subtask-2 on the training dataset should be interpreted with some caution. Given the small training set of only 465 fallacious instances and the use of 10-fold cross-validation, there is a risk that the reported scores reflect a degree of overfitting to the training distribution. Regularization techniques such as stronger L_2 penalties on the Logistic Regression base learners could help address this in future experiments. However, the official test result in Table 9 shows a significant drop in performance, with F1-macro of 0.766, indicating overfitting to the training distribution. This is a known limitation of TF-IDF based models trained on small datasets.

Comparison Across Approaches

The results in Tables 6 and 5 highlight a clear performance gap between classical machine learning methods and neural approaches on this task. Fine-tuned transformer models, including `DeBERTa-v3-large` (F1-score 0.74) and `RoBERTa-base` (0.73), underperform relative to the Stacking ensemble (0.84). This is likely attributable to the limited size of the training data (938 instances), which is insufficient for effective fine-tuning of large pre-trained models. Prompt-based approaches using `bart-large-mnli` achieve a macro F1-score of 0.33-0.35, suggesting that models with relatively few parameters and NLI-specific pre-training are poorly suited for nuanced fallacy detection through prompting alone. Larger instruction-tuned models with greater parameter counts may yield better results, but at a considerably higher computational cost.

Beyond raw performance, our machine learning based systems offer several practical advantages. Classical ensemble models train quickly, require minimal memory, and are straightforward to interpret — properties that are particularly valuable in low-resource settings. Since inference does not rely on any large language model, the deployed system is also fast and inexpensive to run, in contrast to heavy-duty fine-tuned transformer models that demand significant computational resources.

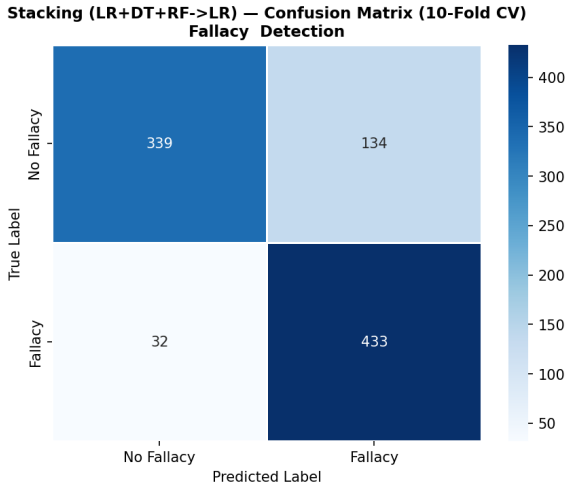

Figure 1: Confusion matrix for Subtask-1 using Stacking (LR+DT+RF→LR) for fallacy detection.

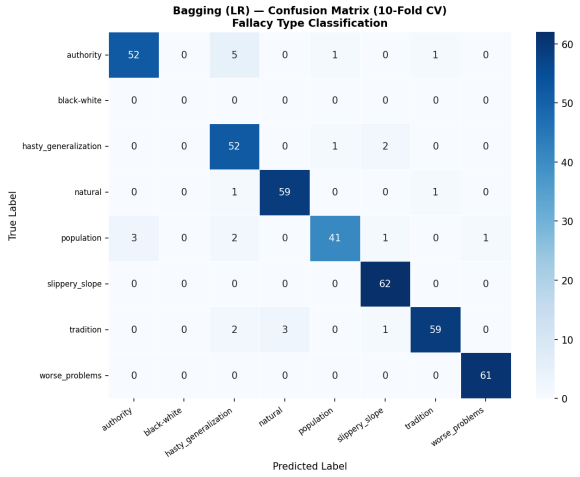

Figure 2: Confusion matrix for Subtask-2 using Bagging (LR) for fallacy type classification.

8. Conclusion and Future Work

This paper presented the NIT-Agartala-NLP Team's submission to the CLEF 2026 Touché Shared Task on Fallacy Detection and Type Classification. We systematically explored classical machine learning models, ensemble methods, prompt-based approaches, and fine-tuning of pre-trained language models across the two subtasks. For Subtask-1 (Fallacy Detection), a stacking ensemble of Logistic Regression, Decision Tree, and Random Forest with Logistic Regression as the meta-learner showed strong performance in development, outperforming both individual classifiers and fine-tuned transformer models in cross-validation. For Subtask-2 (Fallacy Type Classification), a Bagging ensemble of Logistic Regression classifiers performed best among all models evaluated. These findings suggest that well-configured classical ensemble methods remain competitive and practical alternatives to large pre-trained language models, particularly in low-resource settings with limited training data.

Despite the overall strong performance in development, the error analysis in Section 7 revealed notable weaknesses. The *black_white* fallacy type was poorly classified in Subtask-2, and the Stacking model for Subtask-1 produced a relatively high false positive rate of approximately 28.3%. Prompt-based approaches using `bart-large-mnli` performed at chance level, and fine-tuned transformer models underperformed relative to classical methods, likely due to the small size of the training dataset.

Several directions remain open for future work. First, we intend to examine individual misclassified examples more closely, particularly for the *black_white* and *population* fallacy types, to better understand the linguistic patterns that confuse the model. Second, since the current systems rely on TF-IDF representations, exploring richer semantic features such as contextual embeddings from pre-trained language models as input to the ensemble classifiers may improve performance without requiring full fine-tuning. Third, we plan to apply prompt-based and fine-tuning approaches to Subtask-2 as well, using larger instruction-tuned language models that were beyond our current hardware capacity. Fourth, extending the approach to multilingual data beyond English would broaden the applicability of the system to fallacy detection in other languages and cultural contexts. Finally, we plan to incorporate explainability tools such as LIME [20] or SHAP [21] to improve the interpretability of model predictions, and to explore automatic argument extraction pipelines that identify claims and supporting premises directly from raw text, removing the current dependency on pre-extracted argument fields from the dataset.

9. Acknowledgments

We would like to express our sincere gratitude to the organisers of the CLEF 2026 Touché Shared Task for their dedicated efforts in curating the dataset, designing the evaluation framework, and promptly addressing participant queries throughout the competition. We are also grateful to the Department of Computer Science and Engineering at the National Institute of Technology Agartala for providing the necessary computational resources, institutional support, and guidance that made our participation in this shared task possible.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT, Claude, and Overleaf's Writefull extension for grammar and spelling checking, as well as paraphrasing and rewording of text. After using these tools and services, the authors reviewed and edited all content as needed and take full responsibility for the content of this publication.

References

[1] J. Kiesel, M. Feger, T. Hagen, S. Heineking, M. Heinrich, M. Fröbe, K. Boland, C. Mathews, W. Pertsch, J. Romberg, I. Zelch, S. Dietze, M. Hagen, M. Potthast, B. Stein, Overview of touché 2026 — argumentation systems, in: M. Hagen, M. Potthast, B. Stein, P. Schaer, E. Zangerle, S. MacAvaney, J. M. Struß, E. S. Salido, A. Barrón-Cedeño, A. G. S. Herrera (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. 17th International Conference of the CLEF Association (CLEF 2026), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2026.

[2] J. Kiesel, M. Feger, T. Hagen, S. Heineking, M. Heinrich, M. Fröbe, K. Boland, C. Mathews, W. Pertsch, J. Romberg, I. Zelch, S. Dietze, M. Hagen, M. Potthast, B. Stein, Overview of touché 2026 — argumentation systems — extended version, in: E. S. Salido, A. Barrón-Cedeño, A. G. S. Herrera, S. MacAvaney, J. M. Struß (Eds.), CLEF 2026 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org, 2026.

[3] M. Fröbe, M. Wiegmann, N. Kolyada, B. Grahm, T. Elstner, F. Loebe, M. Hagen, B. Stein, M. Potthast, Continuous integration for reproducible shared tasks with tira.io, in: J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, A. Caputo (Eds.), Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2023, pp. 236–241. doi:10.1007/978-3-031-28241-6_20.

[4] Shivam, M. Kumar, A. Jamatia, NIT-agartala-NLP-team at SemEval-2026 task 9: A weighted soft-voting ensemble framework of fine-tuned LLMs for binary and multi-label polarization detection, in: E. Kochmar, D. Ghosh, K. North, M. Komachi (Eds.), Proceedings of the 20th International Workshop on Semantic Evaluation (2026), Association for Computational Linguistics, San Diego, California, USA, 2026, pp. 3169–3181. URL: https://aclanthology.org/2026.semeval-1.398/. doi:10.18653/v1/2026.semeval-1.398.

[5] P. McCullagh, J. A. Nelder, Generalized Linear Models, 2nd ed., Routledge, New York, 1989. URL: https://doi.org/10.1201/9780203753736. doi:10.1201/9780203753736.

[6] L. Rokach, O. Maimon, Decision Trees, Springer US, Boston, MA, 2005, pp. 165–192. URL: https://doi.org/10.1007/0-387-25465-X_9. doi:10.1007/0-387-25465-X_9.

[7] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32. URL: https://doi.org/10.1023/A:1010933404324. doi:10.1023/A:1010933404324.

[8] E. Tuv, Ensemble Learning, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 187–204. URL: https://doi.org/10.1007/978-3-540-35488-8_8. doi:10.1007/978-3-540-35488-8_8.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, At-

tention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 6000–6010.

[10] S. Pratap, A. R. Aranha, D. Kumar, G. Malhotra, A. P. N. Iyer, S. S.S., The fine art of fine-tuning: A structured review of advanced llm fine-tuning techniques, Natural Language Processing Journal 11 (2025) 100144. URL: https://www.sciencedirect.com/science/article/pii/S2949719125000202. doi:https://doi.org/10.1016/j.nlp.2025.100144.

[11] F. Pan, X. Wu, Z. Li, A. T. Luu, Are LLMs good zero-shot fallacy classifiers?, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 14338–14364. URL: https://aclanthology.org/2024.emnlp-main.794/. doi:10.18653/v1/2024.emnlp-main.794.

[12] T. Alhindi, T. Chakrabarty, E. Musi, S. Muresan, Multitask instruction-based prompting for fallacy recognition, in: Y. Goldberg, Z. Kozareva, Y. Zhang (Eds.), Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 8172–8187. URL: https://aclanthology.org/2022.emnlp-main.560/. doi:10.18653/v1/2022.emnlp-main.560.

[13] E. Cantín Larumbe, A. Chust Vendrell, Argumentative fallacy detection in political debates, in: E. Chistova, P. Cimiano, S. Haddadan, G. Lapesa, R. Ruiz-Dolz (Eds.), Proceedings of the 12th Argument mining Workshop, Association for Computational Linguistics, Vienna, Austria, 2025, pp. 369–373. URL: https://aclanthology.org/2025.argmining-1.36/. doi:10.18653/v1/2025.argmining-1.36.

[14] M. Heinrich, C. Mathews, B. Stein, Touché26-fallacy-detection, 2026. URL: https://doi.org/10.5281/zenodo.19925569. doi:10.5281/zenodo.19925569.

[15] S. Sahai, O. Balalau, R. Horincar, Breaking down the invisible wall of informal fallacies in online discussions, in: C. Zong, F. Xia, W. Li, R. Navigli (Eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online, 2021, pp. 644–657. URL: https://aclanthology.org/2021.acl-long.53/. doi:10.18653/v1/2021.acl-long.53.

[16] T. Roelleke, J. Wang, Tf-idf uncovered: a study of theories and probabilities, in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 435–442. URL: https://doi.org/10.1145/1390334.1390409. doi:10.1145/1390334.1390409.

[17] J. R. Quinlan, Bagging, boosting, and c4.s, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 1, AAAI'96, AAAI Press, 1996, p. 725–730.

[18] W. Wang, V. W. Zheng, H. Yu, C. Miao, A survey of zero-shot learning: Settings, methods, and applications, ACM Trans. Intell. Syst. Technol. 10 (2019). URL: https://doi.org/10.1145/3293318. doi:10.1145/3293318.

[19] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020.

[20] M. T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?": Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 1135–1144. URL: https://doi.org/10.1145/2939672.2939778. doi:10.1145/2939672.2939778.

[21] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 4768–4777.