Measuring and Mitigating Local Instability in Deep Neural Networks

Arghya Datta[†] and **Subhrangshu Nandi**[†] and **Jingcheng Xu**[†*] and **Greg Ver Steeg** and **He Xie** and **Anoop Kumar** and **Aram Galstyan**
Amazon Alexa
Seattle, WA, USA
{argdatta, subhrn, gssteeg, hexie, anooamzn, argalsty} @amazon.com
{xjc}@stat.wisc.edu
[†] Equal contribution, alphabetical order

Abstract

Deep Neural Networks (DNNs) are becoming integral components of real world services relied upon by millions of users. Unfortunately, architects of these systems can find it difficult to ensure reliable performance as irrelevant details like random initialization can unexpectedly change the outputs of a trained system with potentially disastrous consequences. We formulate the model stability problem by studying how the predictions of a model change, even when it is retrained on the same data, as a consequence of stochasticity in the training process. For Natural Language Understanding (NLU) tasks, we find instability in predictions for a significant fraction of queries. We formulate principled metrics, like per-sample "label entropy" across training runs or within a single training run, to quantify this phenomenon. Intriguingly, we find that unstable predictions do not appear at random, but rather appear to be clustered in data-specific ways. We study data-agnostic regularization methods to improve stability and propose new data-centric methods that exploit our local stability estimates. We find that our localized data-specific mitigation strategy dramatically outperforms data-agnostic methods, and comes within 90% of the gold standard, achieved by ensembling, at a fraction of the computational cost.

1 Introduction

While training large deep neural networks on the same data and hyperparameters may lead to many distinct solutions with similar loss, we say that the model is *underspecified* (D'Amour et al., 2022). One tangible manifestation of underspecification is that a model prediction on a single data point can change across different training runs, without any change in the training data or hyperparameter settings, due to stochasticity in the training procedure. This extreme sensitivity of model output, which has been termed as *model variance/instability* or *model jitter/churn* (Hidey et al., 2022; Milani Fard et al., 2016), is highly undesirable as it prohibits comparing models across different experiments (Dodge et al., 2019). We refer to this problem as *local instability* [1], a term that highlights our focus on the non-uniformity of instability across data points. Local instability can lead to highly undesirable consequences for deployed industrial systems, as it can cause inconsistent model behavior across time, eroding trust on AI systems (Dodge et al., 2020; D'Amour et al., 2020a). The problem is further exacerbated by the fact that industry models are typically more complex and trained on diverse datasets with potentially higher proportion of noise.

Utterance (gold label)	$\hat{p}_{[min-max]}$, $\sigma_m(\hat{p})$	**Label predictions over 50 runs**
funny joke (general)	[0.98-0.99], 0.003 (low)	general:50
start house cleanup (IOT)	[0.002-0.97], 0.17 (high)	lists:26, IOT:6, general:6, play:5, news:3, social:1, calendar:1
search for gluten free menus (cooking)	[0.002-0.693], 0.06 (low)	lists:28, takeaway:18, social:1, music:1, cooking:1, play:1

Table 1: Variability in predictions of utterances from Massive dataset. This shows different predictions over 50 model runs with different seeds. $\hat{p}$ is the prediction score on gold labels and σ_m is the standard deviation over multiple model outputs $\hat{p_1}, \ldots, \hat{p_{50}}$. For example, *start house cleanup* with gold label *IOT* is predicted to label *lists* 26 out of the 50 model runs. Its prediction score on *IOT* ranges between 0.002 and 0.97. green: low variability, predictions match gold label, red: high predicted label flipping or switching.

Table 1 shows examples of local instability for a domain classification problem, where we used a pre-trained language model DistilBERT (Sanh et al., 2019) to train 50 independent classifiers

[*] Work done as an intern at Amazon Alexa

[1] We use *local instability* to mean *local model instability*

(with random initial conditions) on Massive dataset (FitzGerald et al., 2022). It shows that a validation set utterance *start house cleanup* with gold label *IOT* gets assigned seven different predicted labels over the 50 runs, with the predicted confidence on gold label $\hat{p}$ ranging between 0.002 and 0.97, with high σ_m (the standard deviation of $\{\hat{p_i}\}_{i=1}^{50}$) of 0.17. In comparison, *search for gluten free menus* gets 6 different predicted labels over 50 runs, with a relatively low σ_m of 0.06. The differences in stability across examples demonstrates that the phenomenon is localized to certain data points. See Figures 4 and 5 in Appendix. Examples in table 1 also highlight that variability in confidence is not perfectly aligned with stability of predictions.

Measuring Local Model Instability While detecting and quantifying local instability across multiple runs that is trivial for toy problems, it becomes infeasible when dealing with much larger industrial datasets. Previous research (Swayamdipta et al., 2020) suggested the use of single-run training dynamics to estimate the variance in prediction scores over multiple epochs for a model. However, as shown in Table 1, low prediction variance does not always lead to less label switching, which is the defining feature of local instability. Instead, here we introduced *label switching entropy* as a new metric for characterizing local instability. Furthermore, we have demonstrated that label switching entropy calculated over different training epochs of a single run is a good proxy for label switching over multiple runs, so that data points with high prediction instability over time also exhibit high instability across multiple training runs.

Mitigating Local Model Instability One straightforward strategy of mitigating local instability is to train an ensemble of N models and average their weights or their predictions. Unfortunately, ensembling neural networks such as large language models is often computationally infeasible in practice, as it requires multiplying both the training cost and the test time inference cost by a factor of N. Therefore, we proposed and compared more economical and computationally feasible solutions for mitigating local instability.

Here we proposed a more efficient smoothing-based approach where we train a pair of models. The first (teacher) model is trained using the one-hot encoded gold labels as the target variable. Once the model has converged and is no longer in the transient learning regime (after N training or optimization steps), we compute the temporal average predicted probability vector over K classes after each optimization step, which is then adjusted by temperature T to obtain the smoothed predicted probability vector. A student model is then trained using these "soft" labels instead of the one-hot encoded gold labels. We call this Temporal Guided Temperature Scaled Smoothing (TGTSS). TGTSS allows local mitigation of local instability as each datapoint is trained to its unique label in the student model. In contrast to existing methods such stochastic weight averaging (Izmailov et al., 2018) or regularizing options such as adding L2-penalty, TGTSS significantly outperforms existing methods and reaches within 90% of the gold standard of ensemble averaging.

We summarize our contributions as follows:

- We propose a new measure of local instability that is computationally efficient and descriptive of actual prediction changes.
- We introduce a data-centric strategy to mitigate local instability by leveraging temporally guided label smoothing.
- We conduct extensive experiments with two public datasets and demonstrate the effectiveness of the proposed mitigation strategy compared to existing baselines.

2 Related work

Sophisticated, real-world applications of Deep Neural Networks (DNNs) introduce challenges that require going beyond a myopic focus on accuracy. Uncertainty estimation is increasingly important for deciding when a DNN's prediction should be trusted, by designing calibrated confidence measures that may even account for differences between training and test data (Nado et al., 2021). Progress on uncertainty estimation is largely orthogonal to another critical goal for many engineered systems: *consistency* and *reliability*. Will a system that works for a particular task today continue to work in the same way tomorrow? One reason for inconsistent performance in real-world systems is that even if a system is re-trained with the same data, predictions may significantly change, a phenomenon that has been called model *churn*(Milani Fard et al., 2016). The reason for this variability is that neural networks are underspecified (D'Amour et al., 2020b), in the sense that there are many different neural networks that have nearly equivalent average performance for the target task. While randomness could be trivially removed by fixing seeds, in practice tiny changes to data will still significantly alter stochasticity and results. We will explore the case of altering training data in future studies. Studying how stochasticity affects model churn addresses a key obstacle in re-training engineered systems while maintaining consistency with previous results.

The most common thread for reducing model churn focuses on adding constraints to a system so that predictions for re-trained system match some reference model. This can be accomplished by adding hard constraints (Cotter et al., 2019) or distillation (Milani Fard et al., 2016; Jiang et al., 2021; Bhojanapalli et al., 2021).

We adopt a subtly different goal which is to train at the outset in a way that reduces variability in predictions due to stochasticity in training. Previous research (Hidey et al., 2022) have suggested a co-distillation procedure to achieve this. Label smoothing, which reduces over-confidence (Müller et al., 2019), has also been suggested to reduce variance, with a local smoothing approach to reduce model churn appearing in (Bahri and Jiang, 2021).

A distinctive feature of our approach is a focus on how properties of the data lead to instability. Inspired by dataset cartography (Swayamdipta et al., 2020) which explored variance in predictions over time during training of a single model, we investigate how different data points vary in predictions across training runs. Non-trivial patterns emerge, and we use sample-specific instability to motivate a new approach to reducing model churn.

Our work draws connections between model stability and recent tractable approximations for Bayesian learning (Izmailov et al., 2018; Maddox et al., 2019). Recent Bayesian learning work focuses on the benefits of Bayesian model ensembling for confidence calibration, but an optimal Bayesian ensemble would also be stable. Bayesian approximations exploit the fact that SGD training dynamics approximate MCMC sampling, and therefore samples of models over a single training run can approximate samples of models across training runs, although not perfectly (Fort et al., 2019; Wenzel et al., 2020; Izmailov et al., 2021). We have studied connections between prediction variability within a training run and across training runs, and used this connection to devise practical metrics and mitigation strategies.

Similar to BANNs (Furlanello et al., 2018), our teacher and corresponding student models use the same model architecture with same no. of parameters rather than using a high-capacity teacher model, however, unlike BANNS, our work is geared towards addressing model instability. Architecturally, our methodology (TGTSS) uses a temperature scaled temporally smoothed vector that is obtained from the last *N* checkpoints from the teacher model instead of the finalized teacher model and not use the annotated labels for the utterances.

3 Model instability measurement

The examples in Table 1 show that re-training a model with different random seeds can lead to wildly different predictions. The variance of predictions across models, σ_m^2, is intuitive, but is expensive to compute and does not necessarily align with user experience since changes in confidence may not change predictions. A changed prediction, on the other hand, may break functionality that users had come to rely on. Hence we want to include a metric which measures how often predictions change.

Therefore, we computed the label switching entropy. Given a setup with training data $\{x_i, y_i\} \in X$ where X are utterances, $y \in \{1, ..., K\}$ are the corresponding gold labels, the multi-run Label Entropy (LE_m) over N independent runs for an utterance x_i can be computed as,

$$LE_m^{(i)} = \sum_{k=1}^{K} -\frac{n_k^{(i)}}{N} \log(\frac{n_k^{(i)}}{N}) \qquad (1)$$

where, n_k is the number of times utterance i was predicted to be in class k across N models trained with different random seeds. For example, if an utterance gets labeled to three classes A, B and C for 90%, 5% and 5% of the time respectively, then its multi-run label entropy ($LE_m^{(i)}$) will be $-(0.9*\log(0.9)+0.05*\log 0.05+0.05 \log 0.05) = 0.39$. Similarly, an utterance that is consistently predicted to belong to one class over N runs will have a $LE_m^{(i)}$ of 0 (even if it is consistently put in the *wrong* class). We can compute the overall LE_m by averaging $LE_m^{(i)}$ for all the utterances. Empirically, we also observed a relatively strong linear relationship between LE_m and σ_m (Figure 1).

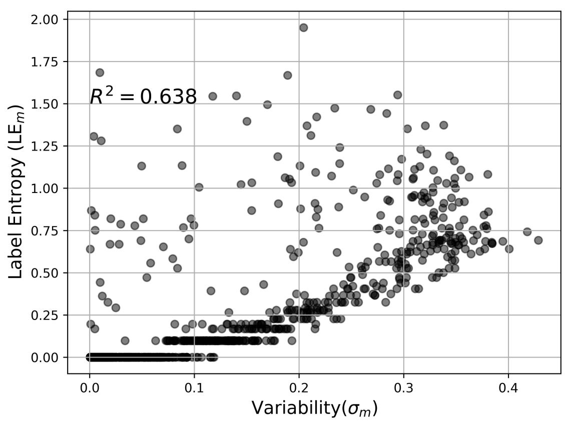

Figure 1: LE_m vs σ_m for Massive dataset shows a strong linear relationship. Each data point is an utterance with $LE_m^{(i)}$ vs $\sigma_m^{(i)}$ values.

Since computing LE_m is computationally expensive due to training N independent models, we propose using *single-run Label Entropy* (LE_s) that can be computed over a single model run. Mathematically, the formula for label entropy stays consistent for both multi-run and single-run, however, LE_s is computed across different model checkpoints. In our analyses, we computed LE_s by accumulating the predicted class after each optimization step, where as LE_m was computed by accumulating the final predicted class across N models on the validation set.

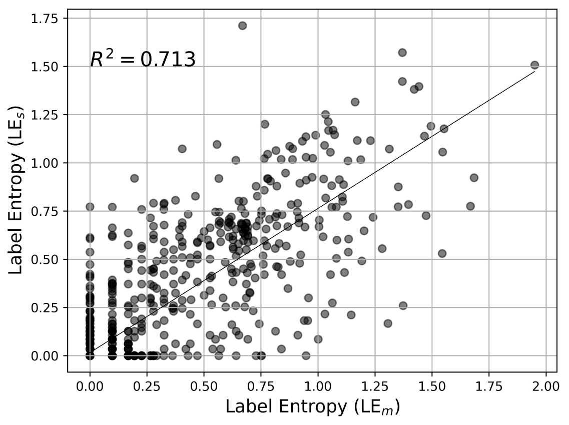

Figure 2: LE_s vs LE_m for Massive dataset shows a strong linear relationship. Each data point is an utterance with $LE_s^{(i)}$ vs $LE_m^{(i)}$ values. Zero entropy corresponds to utterances with confidence scores close to 1 for a class with very low variability.

Empirically, we found that there exists a strong linear relationship between LE_s and LE_m (Figure 2). This demonstrates that utterances which suffer from local instability across multiple independent runs exhibit similar instability across multiple optimization steps for a single model. This finding supports our hypothesis that LE_s is a suitable proxy for LE_m in real world production settings for NLU systems.

4 Model instability mitigation

In our study, we have explored 3 baseline mitigation strategies to address model instability: ensembling, stochastic weight averaging (SWA) and uniform label smoothing. These methodologies have been used in numerous other works to improve generalization as well as predictive accuracy across a diverse range of applications. Performance of the ensembling strategy serves as our upper bound in reducing model instability. We propose a novel model instability mitigation strategy, temporal guided temperature scaled label smoothing, that is able to recover 90% of the reduction in model instability as ensembling at a fraction of model training time and computational cost. We describe all the mitigation strategies below.

4.1 Ensemble averaging and regularizing

In this setting, we trained N independent models, initialized with different random seeds, using the standard cross-entropy loss, computed between the ground truth labels and the predicted probability vector. For every utterance in the test set, we recorded the mean predicted probability of the gold label, the predicted label and our proposed local instability metric, label entropy, across N models. We also trained another baseline by leveraging $L2$ regularization. No other mitigation strategies were used in the process since our aim was to emulate the current model training scenario in natural language understanding(NLU) production settings.

4.2 Stochastic Weight Averaging

Stochastic weight averaging(SWA) (Izmailov et al., 2018) is a simple yet effective model training methodology that improves generalization performance in deep learning networks. SWA performs an uniform average of the weights traversed by the stochastic gradient descent based optimization algorithms with a modified learning rate. In our implementation, we equally averaged the weights at the end of the last two training epochs. We also explored equal averaging of weights from two randomly selected epochs out of the final 3 epochs but that strategy did not yield better results. We left the work of using a modified learning rate to a future study with a significantly larger training dataset.

4.3 Label Smoothing

Label smoothing (Szegedy et al., 2016) is a popular technique to improve performance, robustness and calibration in deep learning models. Instead of using “hard” one-hot labels when computing the cross-entropy loss with the model predictions, label smoothing introduces “soft” labels that are essentially a weighted mixture of one-hot labels with the uniform distribution. For utterances $\{x_i, y_i\}$ where $y \in \{1, ..., K\}$ for K classes, the new "soft" label is given by $y^{LS} = (1 - \alpha) * y + \alpha/K$ where α is the label smoothing parameter. The "soft" labels are then used in the softmax cross-entropy loss.

4.4 Ensemble baseline

To obtain consistent predictions with low local instability, ensembling is often utilized as the default mitigation strategy. Given a problem setup with training data $\{x_i, y_i\} \in X$ where X are utterances, $y \in \{1, ..., K\}$ are the corresponding gold labels, then intuitively, ensembling over N independent models,where N is sufficiently large, will converge to the average predicted probability by the law of large numbers. Hence, using a sufficiently large ensemble of independently trained models would give stable predictions in general.

In our study, we used ensembling to aggregate (uniform average) predictions for each utterance across N independently trained models. Each model was trained using the softmax cross-entropy loss between the predicted logit vector z_i over K classes and the one-hot encoded vector representing the gold label. For an utterance x_i, the uniform average predicted probability vector $\bar{p_i}$ across N models over all class K (softmax probability vector of length K) is adjusted by a temperature T, to obtain the smoothed predicted probability vector q_i:

$$q_i = \frac{\bar{p_i}^T}{\sum_{k=1}^{K} \bar{p_k}^T} \tag{2}$$

The temperature T can be used to control the entropy of the distribution. The smoothed probability vector q is now used as the "soft" labels to train a model instead of the "hard" one hot encoded gold labels and the resultant model is robust to local instability. One challenge for ensembling is that it requires training, storing and running inference on a large number of models which is often infeasible for large scale NLU systems.

4.5 Temporal Guided Temperature Scaled Smoothing (TGTSS)

Since ensembling is infeasible for large models in practice, we propose temporal guided label smoothing that does not require training large ensembles to compute the soft labels.

In this setup, we train a pair of models as opposed to training a large ensemble of models. The first (teacher) model is trained using the one-hot encoded gold labels as the target. Once the model has converged and is no longer in the transient training state (after N training or optimization steps), we compute the uniform average predicted probability vector ($\bar{p_i}$) after each optimization step of the model, which is then adjusted by temperature T to obtain the smoothed predicted probability vector q_i using eqn.(2). A suitable N can be chosen by looking at the cross-entropy loss curve for the validation dataset. The second (student) model is now trained using q_i as the "soft" label instead of the one-hot encoded gold labels.

The significant advantage of TGTSS over ensembling is that it does not require training, storing, or inferring over large ensembles. A key feature of TGTSS is that it uniformly averages predictions over numerous training steps instead of averaging predictions over numerous independent models. This saves the cost of training multiple models. Moreover, we never need to store multiple models for TGTSS since we can store a running average of the predictions over time. Finally, at inference time we only need to call a single model (the trained student model), as opposed to N models for the ensemble.

5 Experimental setup and results for mitigation

5.1 Base model architecture

For all our experiments, we used DistilBERT (Sanh et al., 2019) as the pre-trained language model. We used the implementation of *DistilBERT-base-uncased* from the *Huggingface* library by leveraging *AutoModelForSequenceClassification*. The pre-trained language model is then fine-tuned on the benchmark datasets by using the training set. DistilBERT is a widely used pre-trained language model that is currently used in production in many large scale NLU systems. One key advantage of using DistilBERT is that it is able to recover more than 90% performance of the larger *BERT-base-uncased* model while using 40% less parameters on the

GLUE language understanding benchmark (Wang et al., 2018). Using other BERT models as the pre-trained language model was outside the scope of this study.

5.2 Datasets

To study local instability and compare different mitigation strategies, we used two open source benchmark datasets (Table 2): Massive and Clinc150.

- Massive: Massive (FitzGerald et al., 2022) dataset is an open source multilingual NLU dataset from Amazon Alexa NLU system consisting of 1 million labeled utterances spanning 51 languages. For our experiments, we only used the *en-US* domain utterances for domain classification task across 18 domains (alarm, audio, general, music, recommendation, etc.).
- Clinc150 DialoGLUE: Clinc150 (Larson et al., 2019) is an open source dataset from DialoGLUE (Mehri et al., 2020), a conversational AI benchmark collection. We utilized Clinc150 for intent classification task across 150 intents (translate, transfer, timezone, taxes, etc).

Attribute	MASSIVE	CLINC150
Source	Amazon Alexa AI	DialoGLUE
Domains Intents	18 60	- 150
Train Holdout(Unseen)	11,514 2974	15,000 3,000
Balanced?	No.	Yes. 100 per intent
Classification task	Domain	Intent

Table 2: Benchmark dataset statistics

5.3 Training and Evaluation Protocol

We compared the performance of our proposed mitigation strategy, *temporal guided temperature scaled smoothing* (TGTSS), with other baseline mitigation strategies such as ensembling averaging, L2 regularization, uniform label smoothing, SWA and ensembling. We trained 50 independent models with the same hyper-parameters for each mitigation strategy using different random initialization seeds. We reported the Mean ± SD of domain classification accuracy for the Massive dataset and Mean ± SD of intent classification accuracy for the Clinc150 dataset. For both the datasets, we also reported the percentage reduction in LE_m when compared to the control baseline over 50 independent model runs for all the utterances as well as for high label entropy utterances whose label entropy was over 0.56 in the control baseline. For each method, we computed the sum of LE_m over all the N utterances in the test set as $\sum_{i=1}^{N} LE_{m_i}$. The ΔLE_m is then computed as the percentage reduction among these values for each method and the control baseline. We did similar computations for ΔLE_s in Table 4.

The LE_m value 0.56 for an utterance indicates that if the utterance was assigned to 2 different labels over 50 independent model runs, then its membership is split 75%-25% between the two labels. A lower value of label entropy indicates better model robustness and consequently, lower local instability. An utterance will have $LE_m = 0$ if it is consistently predicted to be the same label across 50 independent model runs. All the results for both the benchmark datasets have been reported on an unseen holdout set. A model having high overall accuracy and low label entropy is usually preferred.

5.3.1 Hyper-parameters

In our empirical analyses, all the models across different mitigation strategies were trained using the ADAM (Kingma and Ba, 2014) optimizer with a learning rate of 0.0001. For both the benchmark datasets, all the models were trained for 5 epochs with a batch size of 256. For the control baseline with L2 regularization, we selected a weight decay value of 0.001. For the ensemble baseline, we selected N as 200 i.e. the pre-temperature scaled "soft" labels were computed after uniformly averaging outputs from 200 independent models for each utterance in the training set. In the uniform label smoothing mitigation strategy, we used α as 0.5 for the Clinc150 dataset and α as 0.1 for the Massive dataset. For SWA, we equally averaged the model weights after the last 2 epochs. For experiments using temporal guided temperature scaled smoothing on the Clinc150 dataset, we used N as 200 where as for the Massive dataset, we set N as 180. This indicates that model outputs after first 200 training or optimization steps were recorded for the Clinc150 dataset and uniformly averaged for each utterance before temperature scaling. Similarly, for the Massive dataset, model outputs were recorded after 180 training steps. For both the ensemble guided and temporal guided temperature scaled smoothing mitigation strategies, we set the temperature T at 0.5.

5.4 Results & Discussion

We compared the proposed mitigation strategy with other baselines described in Section 4.1. We highlight the effectiveness of our proposed local instability metric, *label entropy*, in capturing local instability over 50 independent model runs as well as a single model run.

Ensemble is the best mitigation strategy

In our empirical analyses, we found that ensemble baseline is often the best performing mitigation strategy in terms of both model accuracy and LE_m for both the benchmark datasets(Table 3).

TGTSS is comparable to ensembing at a fraction of computation cost

We found that TGTSS is able to recover about 91% of the performance of ensembling in the multi-run experiments. TGTSS trains only one teacher-student pair and drastically reduces the computational cost of ensembling. Hence, it is much more feasible to deploy TGTSS in production NLU systems. We also found that TGTSS is significantly better than model-centric local instability mitigation strategies such as SWA and L2 regularization.

However, as mentioned in Section 4.5, TGTSS computes "soft" labels across multiple optimization steps which leads to multiple inference cycles. In our experiments, we ran inference after each optimization step once the model is no longer in the transient training state. However, it may be possible to further reduce the number of inference cycles by running inference after every X optimization steps and this is left for future studies.

Efficacy of single run label entropy (LE_s) as a local instability metric

In Table 3, we demonstrated how TGTSS is able to reduce local instability in terms of our proposed metric LE_m over multiple independent runs of the model and recover 91% of the performance of ensembling. We proposed LE_s as a more practical metric for local instability. We showed that TGTSS is still able to recover more than 90% of the performance of ensembling for the Clinc150 and the Massive datasets (Table 4). For high LE_m utterances in the control baseline, TGTSS was able to considerably reduce LE_s (Appendix Table 6).

In figure 3, we can observe that TGTSS significantly reduces variation in prediction scores compared to the control baseline. In the top panels, we see utterances that are easy to learn and the classifier converged to the gold label within 2 epochs. In bottom panels, we see utterances that exhibit high variation in prediction scores through the training process, and consequently, high LE_s. Post mitigation by TGTSS, the bottom right panel shows the significant reduction in prediction score variation and LE_s. Figure 8 in Appendix shows more examples of reduction in LE_s over the course of training.

Global label smoothing is not as effective

In our empirical analyses, we found that uniform label smoothing reduces local instability by 7-9% compared to the control baseline but falls short of ensembling and TGTSS. Label smoothing involves computing a weighted mixture of hard targets with the uniform distribution, where as both ensembling and TGTSS uses the model's average predictions over multiple runs and multiple optimization steps, respectively. Tuning the smoothing factor(α) did not improve model stability in terms of label entropy.

Importance of temperature scaling for TGTSS

We conducted ablation studies to understand how temperature scaling affects the performance of TGTSS. Temperature scaling uses a parameter $T < 1$ for all the classes to scale the uniformly averaged predictions. We found that the proposed methodology reduces label entropy by 17.5% over the control baseline without temperature scaling for the Massive dataset on the validation set (31.5% reduction with temperature scaling). This also indicates that temporal uniform averaging is independently able to significantly reduce label entropy.

6 Conclusion

In this work, we studied the problem of model instability/churn in deep neural networks in the context of large scale NLU systems. Assigning different labels to the same training data over multiple training runs can be detrimental to many applications based on DNNs. We noticed that the instability of model predictions are non-uniform over the data, hence we call it local instability. We proposed a new metric, *label switching entropy*, that is able to quantify model instability over multi-runs as well as a single training run. We also introduced *Temporal Guided Temperature Scaled Smoothing* that reduces model

	Massive			Clinc150		
Methods	Accuracy(%)	$\Delta LE_m(\%) \uparrow$	% of E_b	Accuracy(%)	$\Delta LE_m(\%) \uparrow$	% of E_b
Control baseline	90.6 ± 0.6	-	-	95.1 ± 0.8	-	-
Ensemble baseline (E_b)	91.3 ± 0.5	34.5	-	95.4 ± 0.6	31.1	-
L2 Regularization	90.3 ± 0.5	-2.3	-7	94.9 ± 0.7	-0.6	-2
SWA	91.0 ± 0.5	17.6	51	95.2 ± 0.7	7.3	23
Label Smoothing	90.8 ± 0.5	5.7	17	95.2 ± 0.8	6.1	20
TGTSS (Ours)	**91.3 ± 0.6**	**31.4**	**91**	**95.3 ± 0.8**	**26.7**	**86**

Table 3: Reduction of multi-run entropy LE_m across 50 independent model runs for different methods. $\Delta LE_m(\%)$ is calculated as percentage reduction between the sum of per-utterance LE_m for each method and that of the control baseline. A higher percentage indicates greater reduction in LE_m over control baseline and thus better performance. The values for % of E_b indicates the reduction in LE_m as a percentage of the gold standard ensemble baseline. A negative sign in label entropy reduction indicates an increase in LE_m. Our method TGTSS shows the best results among the competing methods, coming within 91% of gold standard ensemble baseline.

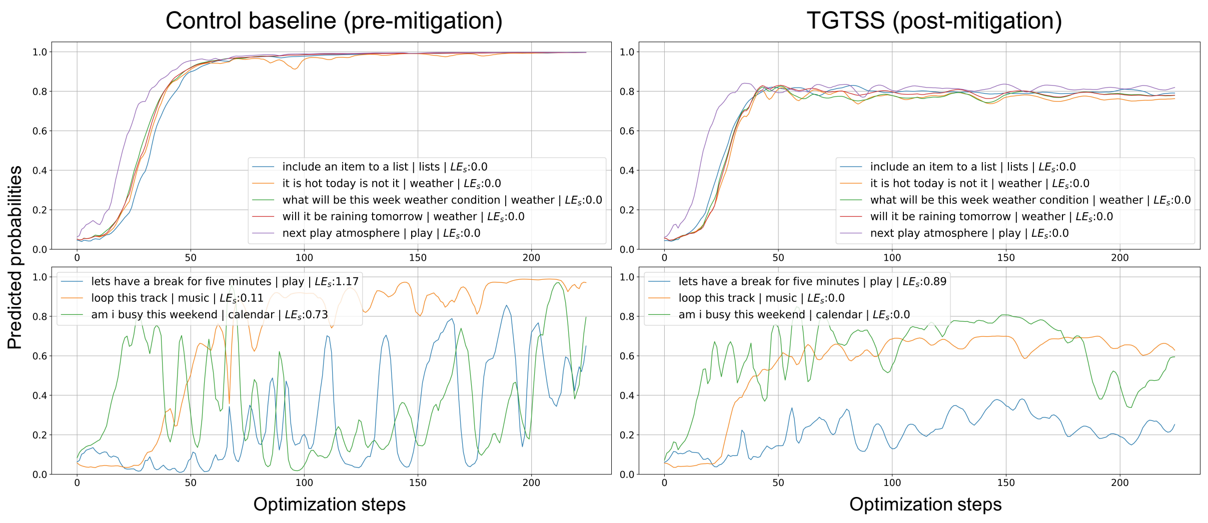

Figure 3: Training trajectories between pre-mitigation and post-mitigation stages show that TGTSS was able to significantly reduce the variability of raw confidence scores on the gold labels as well as reduce model churn in Massive dataset. [Top] shows some utterances where the model predictions are stable (no label switching), [Bottom] shows some utterance where TGTSS significantly reduced model churn as measured using LE_s.

	ΔLE_s (%) $\uparrow$	
Methods	Massive	Clinc150
Label Smoothing	37.9	40.5
Ensemble baseline	**55.5**	**61.7**
TGTSS (Ours)	53.4	55.9

Table 4: Empirical analyses highlights Temporal guided temperature scaled smoothing (TGTSS) reduces LE_s with respect to the single run control baseline model across different optimization steps when a single model is trained. ΔLE_s (%) is computed as percentage reduction between the sum of per-utterance LE_s for each method and that of the control baseline. A $-ve$ sign indicates an increase in label entropy over the control baseline.

churn by a considerable margin. We have shown in experiments that TGTSS is able to recover up to 91% of the performance of ensembling at a fraction of computational cost for training and storing, thereby providing a viable alternative to ensembling in large scale production systems. Future directions of research include expanding our analyses to multi-modal datasets and further dissecting the root causes behind local model instability.

Limitations

Even though our proposed methodology, TGTSS, was able to significantly reduce model instability, there is still a gap in performance with the gold standard ensembling techniques. More work needs to be done to bridge this gap. In our empirical analysis, we used two open source datasets, Massive and Clinc150. Both these datasets are small and may not represent the complexity in real world production datasets which may contain substantially large noise. In our proposed methodology, we train a pair of models successively, a teacher and a student, which is significantly better than ensembling in terms of computational cost. However, this setup may still be challenging in many sophisticated real world production NLU systems. More work needs to be done to reduce the computational complexity of training and inference for these systems.

Ethics Statement

The authors foresee no ethical concerns with the research presented in this work.

Acknowledgement

The authors would like to thank the anonymous reviewers and area chairs for their suggestions and comments.

References

Dara Bahri and Heinrich Jiang. 2021. Locally adaptive label smoothing for predictive churn. *arXiv preprint arXiv:2102.05140*.

Srinadh Bhojanapalli, Kimberly Wilber, Andreas Veit, Ankit Singh Rawat, Seungyeon Kim, Aditya Menon, and Sanjiv Kumar. 2021. On the reproducibility of neural network predictions. *arXiv preprint arXiv:2102.03349*.

Andrew Cotter, Heinrich Jiang, Maya R Gupta, Serena Wang, Taman Narayan, Seungil You, and Karthik Sridharan. 2019. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *J. Mach. Learn. Res.*, 20(172):1–59.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. 2022. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. 2020a. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. 2020b. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, et al. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv preprint arXiv:2204.08582*.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. 2019. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1607–1616. PMLR.

Christopher Hidey, Fei Liu, and Rahul Goel. 2022. Reducing model jitter: Stable re-training of semantic parsers in production environments. *arXiv preprint arXiv:2204.04735*.

Pavel Izmailov, Dmitrii Podoprikhin, T. Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *ArXiv*, abs/1803.05407.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. 2021. What are bayesian neural network posteriors really like? In *International conference on machine learning*, pages 4629–4640. PMLR.

Heinrich Jiang, Harikrishna Narasimhan, Dara Bahri, Andrew Cotter, and Afshin Rostamizadeh. 2021. Churn reduction via distillation. *arXiv preprint arXiv:2106.02654*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. 2019. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32.

Shikib Mehri, Mihail Eric, and Dilek Z. Hakkani-Tür. 2020. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *ArXiv*, abs/2009.13570.

Mahdi Milani Fard, Quentin Cormier, Kevin Canini, and Maya Gupta. 2016. Launch and iterate: Reducing prediction churn. *Advances in Neural Information Processing Systems*, 29.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? *Advances in neural information processing systems*, 32.

Zachary Nado, Neil Band, Mark Collier, Josip Djolonga, Michael W Dusenberry, Sebastian Farquhar, Qixuan Feng, Angelos Filos, Marton Havasi, Rodolphe Jenatton, et al. 2021. Uncertainty baselines: Benchmarks for uncertainty & robustness in deep learning. *arXiv preprint arXiv:2106.04015*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multitask benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. 2020. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*.

A Appendix

A.1 Variance confidence plots

We have plotted the mean confidence and the variance of utterances in the validation dataset for both the Massive (Figure 4) and Clinc150 (Figure 5) datasets. From our analysis, we see that there are utterances that exhibit high variance and medium confidence (around 0.5) which often leads to predicted label flips or model churn over multiple training runs of the model. We also see that there are utterances that possess low confidence corresponding to the gold label and has very low variance. These utterances are probably annotation errors. The bulk of the utterances have high confidence on average corresponding to the gold label and low confidence which signifies that the model predictions are mostly consistent on these utterances.

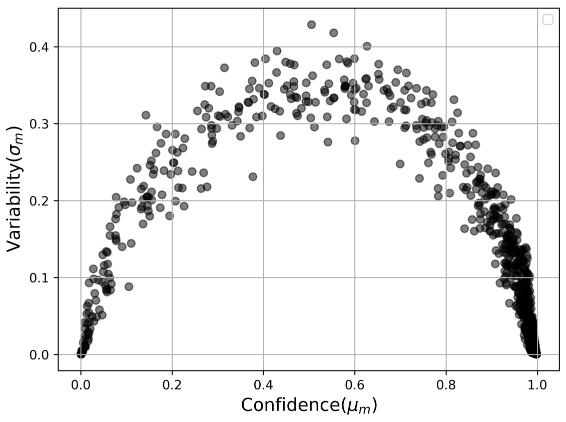

Figure 4: Plot of multi-run confidence(μ_m) and standard deviations(σ_m) of prediction scores for Massive data (validation dataset), from the domain classifier model

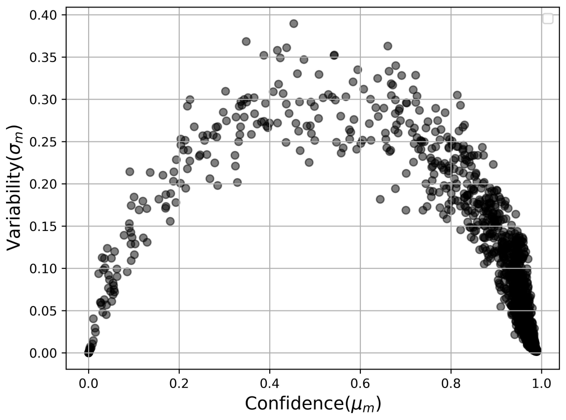

Figure 5: Plot of multi-run confidence(μ_m) and standard deviations(σ_m) of prediction scores for Clinc150 data (validation dataset), from the intent classifier model

A.2 Relationship between LE_s, LE_m and μ_m

As shown earlier in the massive dataset, there is a strong relationship between LE_m and μ_m. We observe a similar trend in the Clinc150 dataset as well (Figure 7). We also observe a similar relationship between single run and multiple run label entropy (LE) for Clinc150 dataset (Figure 6). This finding supports our analysis that label entropy is a suitable proxy for model churn.

A.3 LE_m & LE_s reduction for high entropy samples

We computed the percentage reduction in LE_m and LE_s post mitigation for utterances that have high LE_m in the control baseline. In our empirical studies, we showed that TGTSS was able to considerably reduce LE_m and LE_s across multi-run and single-run experiments when compared to the gold standard ensembling (Appendix Tables 5, 6).

A.4 Label entropy over optimization steps

We have used LE_s as a suitable proxy for LE_m. In Figure 8, we provide empirical evidence that our proposed methodology, TGTSS, was able to reduce label entropy as the model is trained over multiple optimization steps. We computed cumulative label entropy till optimization step T and observed that

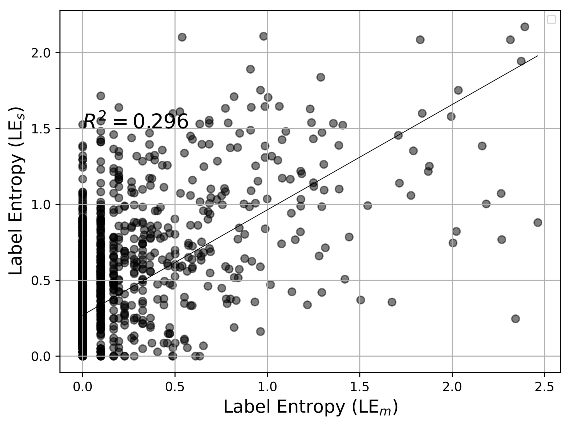

Figure 6: LE_s vs LE_m for Clinc150 dataset (validation set) shows a strong linear relationship. Each data point is an utterance with $LE_s^{(i)}$ vs $LE_m^{(i)}$ values. Zero entropy corresponds to utterances with confidence scores close to 1 for a class with very low variability.

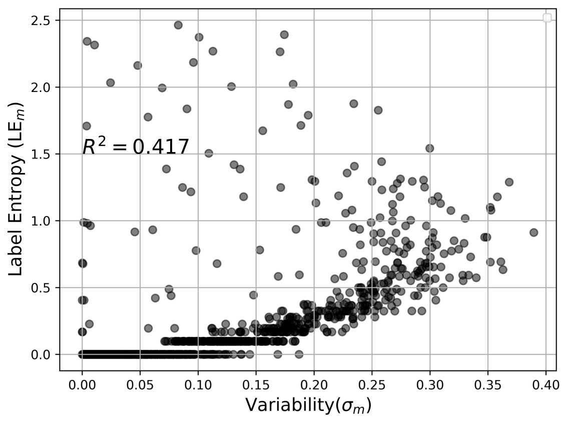

Figure 7: LE_m vs σ_m for Clinc150 dataset (validation set) shows a strong linear relationship. Each data point is an utterance with $LE_m^{(i)}$ vs $\sigma_m^{(i)}$ values.

	ΔLE_m (%) $\uparrow$	
Methods	Massive	Clinc150
Control baseline	-	-
Ensemble baseline	27.4	24.1
L2 Reg.	3.8	4.2
SWA	11.3	4.3
Label Smoothing	5.4	8.1
TGTSS (Ours)	**26**	**22.4**

Table 5: Empirical analyses highlights TGTSS reduces LE_m for high LE_m samples of the control baseline by a considerable margin in multi-run experiments. The column $\Delta LE_m(\%) \uparrow$ is computed as percentage reduction between the sum of per-utterance LE_m for each method and that of the control baseline. A higher value indicates greater reduction in LE_m over control baseline.

	ΔLE_s (%) $\uparrow$	
Methods	Massive	Clinc150
Label Smoothing	14.9	20.7
Ensemble baseline	36.4	40.7
TGTSS (Ours)	**31.5**	**33.6**

Table 6: Empirical analyses highlights TGTSS reduces LE_s for high LE_m samples of the control baseline by a considerable margin in single-run experiments. The column ΔLE_s (%) $\uparrow$ is computed as percentage reduction between the sum of per-utterance LE_s for each method and that of the control baseline.

as the model was being trained, the label entropy of some of the utterances dropped closer to 0.

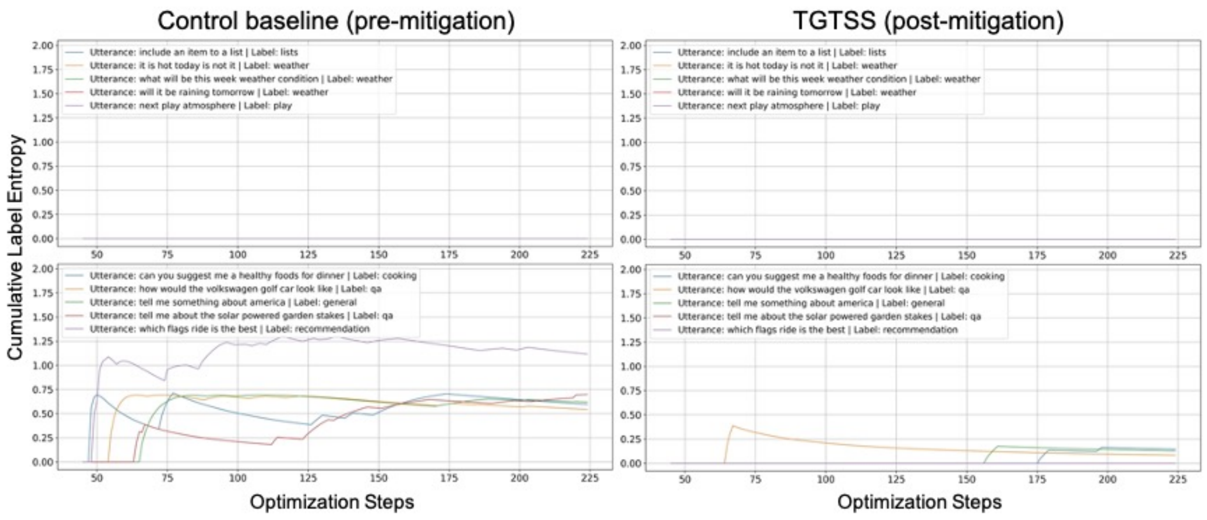

Figure 8: Training trajectories between pre-mitigation and post-mitigation stages show that TGTSS was able to significantly reduce label entropy as the model is trained. [Top] shows some utterances where the model predictions are stable as label entropy is always 0, [Bottom] shows some utterance where TGTSS significantly reduced model churn as measured using LE_s.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
section 7 (Limitations)

☑ A2. Did you discuss any potential risks of your work?
section 7 (Limitations)

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract, Introduction (section 1)

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4, 5

☑ B1. Did you cite the creators of artifacts you used?
section 9 (References)

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
In section 5, we have cited the public datasets that were used for this research.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 5. Results section

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 5 Results

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4 and 5

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4 and 5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 5

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.