INFFEED: Influence Functions as a Feedback to Improve the Performance of Subjective Tasks

Somnath Banerjee, Maulindu Sarkar, Punyajoy Saha, Binny Mathew,
Animesh Mukherjee
Indian Institute of Technology Kharagpur, India
som.iitkgpcse@kgpian.iitkgp.ac.in, animeshm@cse.iitkgp.ac.in
{moulindusarkar, punyajoysaha1998, binnyiitkgp}@gmail.com

Abstract

Recently, *influence functions* present an apparatus for achieving explainability for deep neural models by quantifying the perturbation of individual train instances that might impact a test prediction. Our objectives in this paper are twofold. First we incorporate influence functions as a feedback into the model to improve its performance. Second, in a dataset extension exercise, using influence functions to automatically identify data points that have been initially 'silver' annotated by some existing method and need to be cross-checked (and corrected) by annotators to improve the model performance. To meet these objectives, in this paper, we introduce INFFEED, which uses influence functions to compute the influential instances for a target instance. Toward the first objective, we adjust the label of the target instance based on its influencer(s) label. In doing this, INFFEED outperforms the state-of-the-art baselines (including LLMs) by a maximum macro F1-score margin of almost 4% for hate speech classification, 3.5% for stance classification, and 3% for irony and 2% for sarcasm detection. Toward the second objective we show that manually re-annotating only those silver annotated data points in the extension set that have a negative influence can immensely improve the model performance bringing it very close to the scenario where all the data points in the extension set have gold labels. This allows for huge reduction of the number of data points that need to be manually annotated since out of the silver annotated extension dataset, the influence function scheme picks up $\sim \frac{1}{1000}$ points that need manual correction.

Keywords: Influence Function, Hateful, Offensive, Stance, Sarcasm, Irony

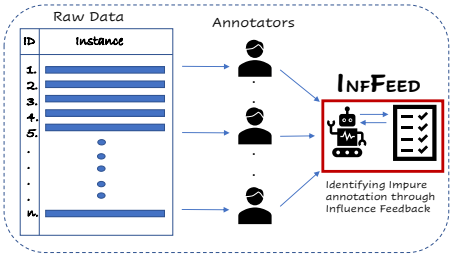

Figure 1: Schematic illustrating our idea of using influence functions to revise the annotations of the target instance.

1. Introduction

In most of the classification problems, the real-world data (training and test instances) are not evenly distributed into classes (Bengio et al., 2020). As a result, the performance of the model suffers significantly, providing motivation to use pre-trained large-scale models. Despite these large models' excellent performance, most deep neural architectures are implemented as a black box and lack algorithmic transparency (Lipton, 2016). Transparency in the method improves the explainability of the model and makes it more trustworthy. Some previous works attempt to explain the predictions of a model (i.e., why the model takes a particular decision) by perturbing the train instances or locally fitting the model on train data (Ribeiro et al., 2016a). In addition, to explain the model, the authors in (Koh and Liang, 2017) formulate influence functions to understand how the model predictions are affected by up-weighting a small amount of training instance loss. The idea is to estimate how much each training sample affects the model's predictions over the test set. Any training sample that causes the test loss to go up is considered less useful and is down-weighted afterward. Given the efficacy of influence-based data resampling in this work, we set a twofold objective. First we show that influence functions can be passed as a feedback to the model to improve its overall performance. Second, for the purposes of extension of annotated datasets, we show that influence functions can automatically identify those data points whose labels need to be cross-checked (and corrected) by annotators out of the full extension set that have been initially 'silver' annotated by some existing model.

Our main contributions to this paper are as follows.

- We propose a framework called INFFEED where we employ the influence function as feedback to adjust the label of a candidate data point based on the labels of its influencers in order to increase the performance of the model.

LREC-COLING 2024, pages 9061–9072
20-25 May, 2024.

- We evaluate the proposed framework on six datasets which are on subjective tasks such as hate speech detection, stance classification, irony, and sarcasm detection.
- We observe that our framework results in an improvement of 4%, 3.5%, 3% and 2% F1 score in the model performance over state-of-the-art baselines for hate speech, stance and irony, and sarcasm classification, respectively.
- For the dataset extension exercise, we show that just manually correcting the labels of the data points that impart a negative influence can result in a performance very close to the case where the whole extension set is gold annotated. The reduction is huge since the negatively influencing set is $\sim \frac{1}{1000}^{\text{th}}$, of the size of the full extension set.

This, we believe, is a first-of-its-kind approach to use influence functions play the role of a pseudo-annotator deciding whether to update the label of target instances in a *text classification* model in order to improve its performance over state-of-the-art baselines.

2. Related work

One of the most critical issues with deep learning models is their interpretability (Guidotti et al., 2018; Lipton and Steinhardt, 2018), and the proneness to learn ambiguous correlations instead of understanding the true nature of the task (Sagawa et al., 2020). These two reasons result in poor outcomes on datasets and cannot meet the expectations (Gururangan et al., 2018; Jia and Liang, 2017; Glockner et al., 2018) resulting in severe biases in model decisions (Blodgett et al., 2020; Sun et al., 2019). This further brings down the overall confidence in the technology (Ribeiro et al., 2016a; Ehsan et al., 2019). Despite great success, the question of "why does the model predict what it predicts?" needs a succinct answer. A satisfactory answer to this question can result in the improvement of the model (Amershi et al., 2015), lead to the development of newer perspectives (Shrikumar et al., 2017), and benefit users by providing explanations of the model actions (Goodman and Flaxman, 2017).

Understanding black-box models by approaches like locally fitting a simpler model around the test point (Ribeiro et al., 2016a) or by perturbing the train point to see how the prediction changes (Simonyan et al., 2013), (Li et al., 2016), (Datta et al., 2016) do not satisfactorily indicate where the model came from (Koh and Liang, 2017). To answer this question, the influence function (Hampel, 1974) was introduced; it was a classic technique based on robust statistics through which the learning algorithm can be inspected, and can be traced back to the most influential training data points which impacts the model to predict what it predicts. A simple and efficient methodology was introduced to align and fit the influence function to the machine learning paradigm, which required access to gradients and Hessian-vector products (Koh and Liang, 2017). It was further demonstrated by (Basu et al., 2020) that non-convex and non-differentiable models, which seem to have limited usefulness, successfully provide significant information while approximated by influence function analysis. On linear models, it can be observed that the influence function is useful in – explaining model predictions, tracking and reducing errors in datasets, debugging models, and even fabricating indistinguishable training set impact[1]. The influence function indicates 'influential' training data points during model prediction and has a plethora of applications. The authors in (Han et al., 2020a) employed them to explain model predictions and uncover data artifacts. They were used by (Yang et al., 2020) in order to determine the quality of synthetic training samples within the framework of data augmentation. The authors in (Kobayashi et al., 2020) investigated what would happen if they used gradient-based approaches in conjunction with influence functions to investigate training history and test stimuli simultaneously. One of the drawbacks of influence functions is that it is highly compute intensive. To circumvent this problem FastIf (Guo et al., 2021), a collection of simple modifications were proposed to significantly improve the runtime for computing influence functions.

Of late, there have been a rising interest in debugging models using explainability techniques (Teso and Kersting, 2019; Lertvittayakumjorn et al., 2020; Guo et al., 2021; Xu and Du, 2020; Nuamah and Bundy, 2020). In (Rajani et al., 2020), the authors suggest utilizing kNN representations to identify training instances responsible for a model's predictions and acquire a corpus-level knowledge of the model's behavior. A recent research (Zylberajch et al., 2021) (HILDIF) has sought to use explainability feedback as input to fine-tune the model for the MNLI dataset. Recently, some comparable tests were carried out using image data, randomly flipping two labels using the influence function (Hao et al., 2020a; Teso et al., 2021; Wang et al., 2018). UIDS by (Wang et al., 2020) and RDIA by (Kong et al., 2022), can both relabel data points based on influence capability using just numeric attributes. To the best of our knowledge, RDIA is the most recent study that addresses the problem of data relabeling followed by a classification task. (Mozes et al., 2023) tried to incorporate LLM and utilized influence functions to relabel the predictions. There is a major gap

[1] https://christophm.github.io/interpretable-ml-book/

between these works and what we can accomplish with the available textual data. Our work differs from these in that it employs influence functions as a pseudo-annotator and leverages the influential instances as feedback to adjust the gold annotation for a target instance, thereby, improving the overall model performance.

3. Preliminaries

Notation: Let us consider a classification task with input text $t \in \mathcal{T} = \{1, 2, ...T\}$ and the label $Y = \{y_1, y_2, ..\}$. Each instance t consists of m no. of words, i.e., $t = \{w_1, w_2, ...w_m\}$. Let us assume that the feature matrix for the input text $\mathcal{T}$ is X. We further denote the training set (texts and their corresponding labels) as (X_{TR}, Y_{TR}). In this work, we have multiple validation sets. The validation set will be denoted by V. For the test data X_{TS}, we have gold labels Y_{TS}, and the predicted label will be denoted by $\hat{Y}_{TS}$.

Influence function: Let us choose an instance (x_i, y_i) from (X_{TR}, Y_{TR}). Let us have a model θ and loss functions $\mathcal{L}((x_i, y_i), \theta)$. Given n number of instances in training set (X_{TR}, Y_{TR}), our objective is to minimize the loss using $\hat{\theta} = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}((x_i, y_i), \theta)$. Now, the objective attempts to identify the influence of the training data points on the learned parameter θ and also on the test data $(x_{ts}, y_{ts}) \in (X_{TS}, Y_{TS})$.

The strength of an influence function is that it attempts to identify the loss locally and tracks the whole model behavior by perturbing or upweighting it. Let us consider that the loss of a particular training data point is denoted by $\pm\delta$. Thus, the influence function for a test data point (x_{ts}, y_{ts}) can be represented as follows.

$$IF\{(x_i, y_i), (x_{ts}, y_{ts})\} \cong \frac{d\mathcal{L}((x_{ts}, y_{ts}), \hat{\theta}_{\pm\delta,(x_i,y_i)})}{d(\pm\delta)} \quad (1)$$

where $\hat{\theta}_{\pm\delta,(x_i,y_i)}$ is the model which has been upweighted or perturbed by $\pm\delta$. The updated loss function thus becomes

$$\hat{\theta} = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} \{\mathcal{L}((x_{ts}, y_{ts}), \theta) + (\pm\delta)\mathcal{L}((x_i, y_i), \theta)\} \quad (2)$$

(Koh and Liang, 2017) have shown that to avoid high computation costs, we can compute the influence function using the approximation below.

$$\begin{aligned} &IF\{(x_i, y_i), (x_{ts}, y_{ts})\} \approx \\ &-\nabla_\theta \mathcal{L}((x_{ts}, y_{ts}), \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}((x_i, y_i), \hat{\theta}) \end{aligned} \quad (3)$$

where $H_{\hat{\theta}}$ is the Hessian matrix of the model parameters. We are interested in identifying the most negatively influential (helpful) data points by considering the perturbation of a data point that leads to a lower loss in a test data point. Thus, if we denote the most negatively influential (helpful) training data point as $(\hat{x_i}, \hat{y_i})$ then it can be presented as

$$(\hat{x_i}, \hat{y_i}) = \arg\min_{(x_i,y_i)\in(X_{TR},Y_{TR})} IF\{(x_i, y_i), (x_{ts}, y_{ts})\} \quad (4)$$

According to (Guo et al., 2021) the computation of equation 4 becomes expensive if the dataset size increases. To overcome this issue, instead of searching those data points in the whole set, we search them in a smaller subset considering minimal changes in the nearest neighbors' quality in retrieving influence-worthy data points. Identification of this subset was based on l_2 distance based on the highly-optimized nearest neighbor search library FAISS (Johnson et al., 2021). So the updated equation becomes

$$(\hat{x_i}, \hat{y_i}) = \arg\min_{(x_i,y_i)\in(\hat{X},\hat{Y})} IF\{(x_i, y_i), (x_{ts}, y_{ts})\} \quad (5)$$

where $(\hat{X}, \hat{Y})$ is a subset of (X, Y) computed using FAISS[2].

Problem definition: Our objective in this paper is to show that the above influence function formulation proposed in the literature can be used to design a feedback mechanism in a learning model to improve upon the performance in any classification task and, in particular, those that are highly subjective in nature.Examples of such subjective tasks include hate speech detection, stance classification, sarcasm, and irony detection. Since these tasks are subjective, there might be 'impure' instances of data points where there are annotator disagreements. In such cases, the idea is whether one can identify other data points that could potentially influence such impure instances. If this hypothesis is valid, one can determine the influence points for the impure point based on the influence function formulation and use the label information of the influence points as a silver label for the impure instances to improve the overall classification performance. We test this hypothesis by having the silver label as feedback in the model. In the next section, we discuss how we design this feedback mechanism.

4. Methodology

In this section, we detail the methodology that we adopt to incorporate the influence function as feedback into the classification model. We also discuss the baselines used in this paper. **Our proposals**: Our proposals include two systems – **System 1** and **System 2**. While **System 1** is the standard classification model, **System 2** is our proposal for incorporating the influence function into the **System 1**. **System 1** is the vanilla

[2]https://github.com/facebookresearch/faiss

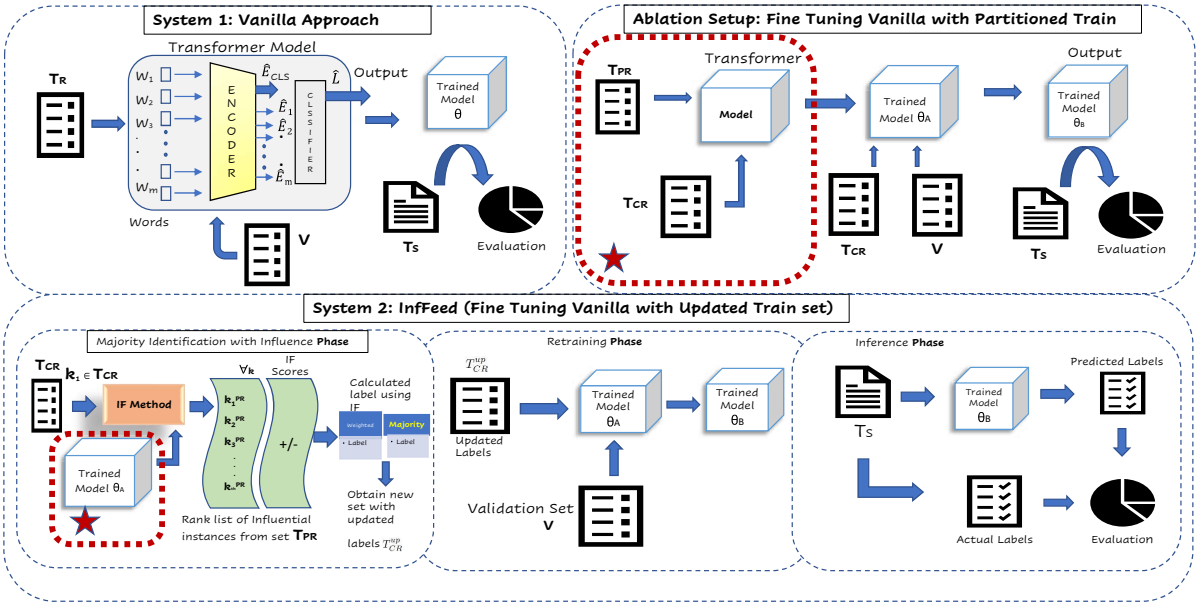

Figure 2: Overview of our proposed approach INFFEED along with **System 1** and the vanilla fine-tuning based ablation setup.

approach where one usually uses a transformer-based classification model having three divisions of a dataset marked as train (T_R), valid V and test T_S. We first train a model with T_R and save the model snapshot θ where the validation loss is minimum and then evaluate the performance using the test data T_S. As shown in Figure 2 (**System 1**) the input text (post/tweet etc.) is split into tokens $\{w_1, w_2, w_3 \cdots w_m\}$ and is passed through a transformer encoder followed by a softmax layer to make the final prediction.

4.1. Influence function to introduce feedback

System 2 (INFFEED): We begin by partitioning the training set[3], denoted by T_R, into a smaller subset T_{CR}, which we designate as a fine-tuning set. Using the remaining part of the training set, $T_{PR} = T_R - T_{CR}$, we then train a model, θ_A. For each instance in T_{CR}, we determine the most influential training instances from T_{PR}, with θ_A and the influence function approach outlined in the preceding section. We revise the label of each instance in T_{CR} based on the majority/weighted voting of the labels from the top-K influential instances identified earlier, producing an updated set T^{up}_{CR}. We proceed to fine-tune θ_A using T^{up}_{CR}. Afterwards, we utilize the held-out validation set V to derive the final model, θ_B. Finally, we evaluate θ_B using the held-out test dataset T_S (Figure 2, **System 2**).

Transformer architectures: We use the BERT (Devlin et al., 2018) and the DistilBERT (Sanh et al., 2019) (a lighter version of BERT) models as transformer architectures throughout this paper.

Baselines: In this paper, we use four state-of-the-art baseline methods taken from the literature – Hao et al. (2020a), Rajani et al. (Rajani et al., 2020), Wang et al. (Wang et al., 2020), and Kong et al. (Kong et al., 2022). As additional baselines, we use two state-of-the-art LLMs GPT-3.5-Turbo[4] and GPT-4[5], in a zero-shot classification setting.

4.2. Influence function to reduce annotation cost

Imagine a scenario where we have T_X training data points already annotated by human annotators and we wish to enhance the performance of the model by extending the training data with gold annotations of another T_Y points. Rather than having all the T_Y points annotated by the humans, we can use INFFEED to selectively annotate a subset of the T_Y points to reduce the overall annotation cost. To this purpose, we first train the model using T_X. Using this trained model we predict the labels for the T_Y points. Thus the T_Y points get silver-annotated. Now we train a fresh model using this silver-annotated T_Y points. For each point in the validation data we get a set of points from T_Y that are most influential using the INFFEED algorithm. Out of these most influential points we concentrate

[3]https://docs.cleanlab.ai/v2.0.0/tutorials/pred_probs_cross_val.html

[4]https://platform.openai.com/docs/models/gpt-3-5

[5]https://platform.openai.com/docs/models/gpt-4

Dataset	Size	#Labels	Name of labels (#instances)
HateXplain (Mathew et al., 2021)	20,148	3	• Hateful (5,935) • Offensive (5,480) • Normal (7,814)
HateSpeech (Davidson et al., 2017)	24,802	3	• Hate speech (1,430) • Offensive (19,190) • Normal (4,163)
WT-WT (Conforti et al., 2020)	51,284	4	• Support (6,663) • Refute (4,224) • Comment (20,864) • Unrelated (19,533)
Stance (Mohammad et al., 2016)	4,163	3	• Favor (1,056) • Against (2,112) • Neither (996)
iSarcasm (Oprea and Magdy, 2020)	4,484	2	• Sarcastic (777) • Non-sarcastic (3,707)
Irony (Van Hee et al., 2018)	3,000	4	• Ironic by clash (1,728) • Situational irony (401) • Other verbal irony (267) • Non irony (604)

Table 1: Dataset details.

on those that negatively influenced the prediction (had negative influence scores). We ask human annotators to check these cases and, if necessary, re-annotate only these points in T_Y. With this revised T_Y we again train the model and find the points negatively influencing the validation data points. Once again these points are re-annotated by humans, if they find it necessary. We repeat this process until in an iteration there are no more negatively influential points.

5. Dataset

The method proposed by us is generic in nature. However, to demonstrate the real effectiveness of the approach, we choose datasets that involve subjective tasks. Our datasets are chosen in a way to cover a wide spectrum of problems and comprise both binary and multiclass scenarios. In specific, we focus on four types of subjective tasks – hate speech detection, stance classification, sarcasm, and irony detection. We evaluate our method on state-of-the-art datasets including – (a) HateXplain (Mathew et al., 2021) and (b) Davidson (Davidson et al., 2017) for hate speech (c) WTWT (Conforti et al., 2020) and (d) (Mohammad et al., 2016) for stance classification, (e) isarcasm (Oprea and Magdy, 2020) for sarcasm detection, (f) (Van Hee et al., 2018) for irony detection. The basic statistics for each of these datasets are given in Table 1.

6. Experimental setup

We use three different setups in our experiment to observe the importance of increasing data. The setups are as follows – (i) S_1: Here, we randomly sample 2500 instances from the dataset. Then, we split these into four parts : T_{PR} (1000 instances), T_{CR} (800 instances), V (200 instances) and T_S (500 instances). (ii) S_2: Here we have 6000 randomly sampled instances and the number of instances in T_{PR}, T_{CR}, V and T_S are 4200, 800, 500 and 500 respectively. (iii) S_3: In this case, the number of randomly sampled instances is 10000. The number of instances in T_{PR}, T_{CR}, V and T_S are 7500, 1500, 500 and 500 respectively. For each setup, we sample the union of T_{PR}, T_{CR}, V three times and compute the performance. We keep the test set T_S fixed across all the setups. We take the average of the three macro F1 scores as the final performance. This result is representative, and the trends remain similar for setups with more than 10000 randomly sampled instances. In the case of the datasets which have less number of instances (less than the total instances in S_2 but more than S_1), we oversample the instances in training data (T_{PR}) using random selection with repetition.

For the baselines (Hao et al., 2020a; Rajani et al., 2020; Wang et al., 2020; Kong et al., 2022) also, we have three such setups; however, during training, we merge T_{PR} and T_{CR} to form a single training set. We let the validation (V) and test (T_S) sets remain the same. For the LLM baselines we query the models with each entry from the test set T_S and record the classification labels in each case.

Model setup: For **System 1** and **System 2** (i.e., INFFEED), we have used two models – BERT-base and DistilBERT. During the fine-tuning, we freeze the first nine layers based on the findings in (Lee et al., 2019) to limit the amount of computation. This leaves us with approximately 14.7M trainable parameters. In the case of DistilBERT, we freeze the first 4 layers to bring down the overall computation cost. For both models, we consider a maximum of 350 tokens. After parameter tuning, the learning rate is set at $2e-5$, the number of epochs at 12, and the batch size at 64. Further, for INFFEED, the weight decay is set to 0.005, the k in kNN to 100, and the Hessian approximation value to 800.

For (Hao et al., 2020a), everything else remaining same as **System 1**, the learning rate has been set to $5e-5$. In the case of this baseline, we treat the hate speech datasets as a two-class classification scenario whereby we merge the 'hateful' and the 'offensive' classes into a single 'abusive' class. Now, during classification, we randomly select 10% of the instances from the entire dataset along with their original labels; we then flip the label for each instance to 'abusive' if the original label is 'normal' and vice versa. We did the same for the sarcasm dataset. For (Rajani et al., 2020), the learning rate and the k in kNN have been set to $5e-5$ and 16, respectively, while everything else remains the same as **System 1**.

For the baselines UIDS (Wang et al., 2020) and RDIA (Kong et al., 2022) we use the Newton-CG algorithm (Martens, 2010) to calculate Influence Functions as mentioned in the paper. For the logistic regression model mentioned in RDIA, we select the regularization term $C = 0.1$.

Setup	HateXplain		WT-WT		IR		ST		iSarcasm		DV	
	Macro F1-score											
	Pretrained embedding											
Wang et al. (Wang et al., 2020) (Lin-UIDS)	0.519		0.490		0.574		0.498		0.502		0.411	
Wang et al. (Wang et al., 2020) (Sig-UIDS)	0.562		0.511		0.624		0.523		0.541		0.497	
Kong et al. (Kong et al., 2022) (RDIA)	0.574		0.536		0.611		0.519		0.546		0.531	
	BBU	**DB**	**BBU**	**DB**	**BBU**	**DB**	**BBU**	**DB**	**BBU**	**DB**	**BBU**	**DB**
Hao et al. (Hao et al., 2020a)	0.623	0.631	-	-	-	-	-	-	0.598	0.577	0.759	0.742
Rajani et al. (Rajani et al., 2020)	0.611	0.585	0.613	0.603	**0.709**	0.626	**0.611**	0.572	0.515	0.524	0.786	0.751
System 1	0.622	0.641	0.613	0.612	0.683	0.680	0.578	0.588	0.603	0.612	0.765	0.746
InfFeed (MV)	0.648	0.639	0.629	0.617	**0.709**	**0.707***	**0.611**	0.603	0.623	0.629	0.784	0.749
InfFeed (WV)	**0.653***	**0.657***	**0.631**	**0.622****	0.701	0.669	**0.605***	**0.605**	**0.629***	**0.635***	**0.799****	**0.770***
	Large Language Models											
gpt-3.5-turbo	0.638		0.629		0.682		0.566		0.493		0.735	
gpt-4	0.644		**0.631**		0.689		0.601		0.541		**0.770**	

Table 2: Macro F1 score for the different models. All bold face entries represent the best performing score and the underlined values represent the best performing baseline. IR: (Van Hee et al., 2018) dataset, ST: (Mohammad et al., 2016) dataset, DV: (Davidson et al., 2017) dataset, BBU: BERT-base-uncased, DB: DistilBERT, MV: majority voting, and WV: weighted Voting. *: Statistically significant results with p-value $<$0.05, and **: Statistically significant results with p-value $<$0.01. Best results are highlighted in bold and second best are underlined.

System setup: We run all of the models described in this study on a Windows-based system equipped with 64 gigabytes of RAM, two 24 gigabytes RTX 3090 GPU connected through SLI, and a Ryzen 9 with a fifth generation, twelve-core CPU.

6.1. Description of the baselines

Hao et al. (Hao et al., 2020a): In this work, authors have proposed an automated weakly supervised scheme along with two metric functions for identifying mislabeled data in a binary classification task. The metric functions are cross entropy loss and the influence function. Cross entropy loss is used to calculate the disparity between ground truth and predicted label. The influence function is used to identify the dependence of the model on the training data. Performance is measured after correcting the mislabeled instances. The authors have conducted the experiments on $\sim$10K images from the real-world clinical questions, i.e., mammographic breast density category classification[6] and breast cancer diagnosis.

Rajani et al. (Rajani et al., 2020): In this work, the authors have proposed a method using k-nearest neighbor representations to identify training instances responsible for prediction. Further, they observed that their proposed method is useful for unveiling learned spurious associations, identifying mislabelled instances, and improving model performance. In order to understand the model behavior, kNN was employed over the hidden representation of the model to identify relevant training instances for a test instance. They then identified the confidence interval where kNN performed better than the model. During inference, they either consider the model's prediction or kNN's prediction based on the confidence ranges where each performed better than the other. They have conducted experiments on multiple datasets such as the Stanford Natural Language Inference (SNLI)[7], the Adversarial NLI (ANLI)[8] and the Heuristic Analysis for NLI Systems (HANS)[9] datasets.

Wang et al. (Wang et al., 2020): In this work, the authors presented a unique Unweighted Influence Data Subsampling (UIDS) approach, and established that the subset-model acquired using the UIDS method can outperform the full-set-model. They separated their whole system into two sections: computing IF and creating probabilistic sampling functions. They created two probabilistic sampling functions, linear sampling (inspired by (Ting and Brochu, 2018)) and sigmoid sampling. This probabilistic sampling strategy manages the worst-case risk across all distributions that are close to the empirical distribution. They demonstrated their abilities on 14 distinct datasets from the medical, text, social, imaging, Physics, CTR, and life domains.

Kong et al. (Kong et al., 2022): In this work, the authors present RDIA, an influence-based relabeling framework for reusing harmful training samples in order to improve model performance. The influence function was used to assess how relabeling a training sample might affect the model's test performance. They conducted their entire experiment on ten distinct datasets (Breast-cancer, Diabetes, News20, Adult, Real-sim, Covtype, Criteo1%, Avazu, MNIST, CIFAR10)[10] based on a set of numerical features. They employed logistic regression (convex optimization) as the classifier. The average test loss with standard deviation

[6] http://www.eng.usf.edu/cvprg/Mammography/Database.html

[7] https://nlp.stanford.edu/projects/snli/

[8] https://huggingface.co/datasets/anli

[9] https://github.com/tommccoy1/hans

[10] https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/

results was used to evaluate performance.

Since UIDS and RDIA models need numerical features as input we obtain pretrained embeddings of all the data points present in our dataset which are then directly fed as input to these models.

7. Influence function as a feedback

In Table 2, we summarize our main results. As our dataset does not have numerical features, we represent the data points using BERT based pretrained embeddings that are fed to UIDS and RDIA as inputs. The **BBU** and **DB** columns show the results using BERT-base-uncased and DistilBERT as the transformer architectures, respectively. All the results are averaged over the three setups S_1, S_2 and S_3. We observe that INFFEED (majority/weighted voting) always outperforms the most competing baselines except for the (Mohammad et al., 2016) dataset, where it is the same as the baseline. In all cases where our models win, the results are statistically significant. In general, INFFEED weighted voting is slightly better than majority voting. Further, for both INFFEED models, the DistilBERT architecture performs better than BERT-base-uncased in most cases. For the baselines (Hao et al., 2020a) and (Rajani et al., 2020), the trends are reversed; BERT-base-uncased generally works better than DistilBERT here.

Our models also outperform the LLM based baselines. The largest performance margin is for the iSarcasm dataset with GPT-4 reporting a macro F1 score of 0.541 compared to INFFEED (WV) at 0.635.

Effect of varying data size: Here we report the performance of the best performing model, INFFEED (majority voting) separately for the three setups – S_1, S_2 and S_3. Figure 3, shows how the performance of the model improves as we increase the dataset size. For some datasets, e.g., (Davidson et al., 2017) and (Conforti et al., 2020), one observes a gain close to 20% as one sweeps from setup S_1 to S_3.

Remark: According to the study by (Koh and Liang, 2017), with N training data points and P parameters, the Hessian matrix computation requires $O(NP^2 + P^3)$ operations, which is unacceptably expensive for massive datasets/models. This is the primary reason for the popularity of the FastIf (Guo et al., 2021) algorithm which is also what we have used here.

Ablation studies: In order to understand the effectiveness of the influence function as a 'pseudo-expert' annotator, we perform two ablation experiments. These are – (a) random flipping and (b) vanilla fine-tuning.

Random flipping: This system uses the same parameters as mentioned in **System 1**. However, here we randomly flip the labels of some of the training instances (around 5%, which is similar in tune to the number of instances updated on average by INFFEED).

Vanilla fine-tuning: As in **System 2**, here also we obtain a model θ_A by training it on T_{CR}. Now rather than computing influence functions, we fine-tune θ_A using T_{CR}.

Subsequently, we use the held-out validation set V and save the new model θ_B where the validation loss is minimum and evaluate the performance with the held-out test set T_S.

The results from the two ablations are reported in Table 3. For *random flipping*, in case of the hate speech datasets, there is an average performance drop of almost 20%. For the stance detection datasets, we can see an average 16% drop, while for the irony and sarcasm datasets, the average drops are nearly 13% and 18%, respectively. In *vanilla fine-tuning* for all the datasets we see an average drop in the range of 2% – 2.5%. Clearly, both the approaches perform worse than INFFEED showing the effectiveness of the influence functions.

Example instances: In Table 4 we show some examples where the incorrect original label gets updated to the correct label based on the votes from the influential instances. This is one of the basic reasons for the better performance of our models.

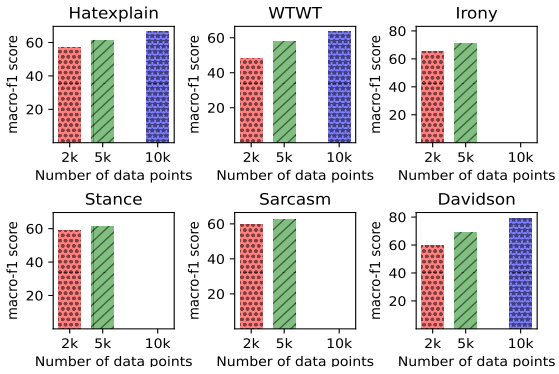

Figure 3: Demonstration of how the macro F1-score improves with increasing data. Stance: (Mohammad et al., 2016) dataset, Irony: (Van Hee et al., 2018) dataset, and Davidson:(Davidson et al., 2017) dataset.

8. INFFEED to reduce annotation cost

Experimental setup: For all the datasets, we use half of T_R as T_X and the other half as T_Y. The validation and the test data are the same as earlier, i.e., V and T_S.

Results: We compare the performance of the

Setup	HateXplain		WT-WT		IR		ST		iSarcasm		DV	
	Macro F1-score											
	BBU	DB	BBU	DB	BBU	DB	BBU	DB	BBU	DB	BBU	DB
Random flipping	0.543	0.510	0.497	0.432	0.590	0.553	0.419	0.397	0.442	0.437	0.543	0.523
Vanilla fine-tuning	0.637	0.656	0.612	0.616	0.691	0.694	0.589	0.599	0.588	0.621	0.770	0.749

Table 3: Ablation studies. IR:(Van Hee et al., 2018) dataset, ST:(Mohammad et al., 2016) dataset, DV:(Davidson et al., 2017) dataset, BBU: BERT-base-uncased, DB: DistilBERT.

Instances	Original label	Influence label
<user> I loveũ more fa**ot	Normal	Offensive
moslem ban and libtard ban but not gun ban	Offensive	Normal
It was her choice Mark	Favor	None
<user> so i got called a nappy headed ni**er jew	Normal	Offensive
make nazi scum lose their jobs	Normal	Offensive
good these mcu fa**ots gonna get schooled on what	Normal	Hatespeech
that shit doesnt even make sense so yes for wetback lmao	Hatespeech	Normal
the white bitch amber guyger has been locked away	Normal	Offensive

Table 4: Samples re-labelled.

BBU model trained on T_Y with all gold annotations (T_Y^{GOLD}), the raw silver annotations of T_Y using the model trained with T_X (T_Y^{SILVER}), and the selectively gold annotated T_Y (T_Y^{INFFEED}) using the INFFEED algorithm repeatedly. The results are shown in Table 5. We observe that the results obtained using T_Y^{INFFEED} are very close to T_Y^{GOLD} and the results from T_Y^{SILVER} are inferior to both of these (except for the iSarcasm dataset). For each dataset, the number of data points in T_Y that had to be re-annotated in total are exceptionally low compared to size of T_Y^{GOLD}.

Dataset	T_Y^{SILVER}	T_Y^{INFFEED}	T_Y^{GOLD}	#re-annotated
HateXplain	61	65	67	17
WT-WT	57	60	61.5	9
IR	66	67	70	11
ST	46	48	55	17
iSarcasm	59	59	61	7
DV	74	75	77	21

Table 5: Comparison of model performance in terms of % accuracy.

9. Error analysis

In this section, we analyze the incorrect predictions from INFFEED and group them into the following types. Here we identify the frequent mistakes made by the model and place them into different categories – **(i)** Presence of slur words may not always correspond to the text being hateful/offensive. Thus, since the model observes such examples in both classes during training it is sometimes confused during prediction. For instance, in the HateXplain dataset, the text “kevin gates make music for dykes and chubby ni**a” is wrongly marked as hate speech by the model owing to the presence of the word “ni**a”, **(ii)** When the major text feature appears only once in a sentence, that feature is given higher importance during annotation. For example, in the (Van Hee et al., 2018) dataset, the statement “Most of us didn’t focus in the #ADHD lecture” is labelled as situational irony while the correct class should be other verbal irony. The influential instances also support this same label even though they do not have any contextual overlap, **(iii)** If there are several targets in the dataset, one individual target instance may be taking influence from another target cluster and producing the incorrect collection of influential instances. For example, in the (Mohammad et al., 2016) dataset, “Jeb Bush is the only sane candidate in this republican lineup” has ‘Donald Trump’ as the target but is influenced by “Letś hope the VOTERS remember" which has ‘Hilary Clinton’ as the target. This may result in possible incorrect predictions, **(iv)** If a post’s contextual information and some label-related input are limited, influential instances are often incapable of capturing the latent semantics of the post. They rather pick up cues from the surface structure. As an example, “Thank user for being so entertaining at the Edinburgh signings! You did not disappoint! I made my flight so will have plenty time to read user” is influenced by instances where the words ‘entertainment’ or ‘flight’ appear in an instance, and **(v)** If the gold annotations are incorrect, influence functions may relate to the wrong instance, resulting in incorrect voting. For example, in the HateXplain dataset, “bare naked islam brought to you by bare naked ladies” is annotated as ‘hate speech’ while it should be actually annotated as ‘offensive’. If this instance is an influencer for a particular target instance then the silver label of the target instance might get incorrectly updated resulting in possible incorrect prediction.

10. Conclusion

We present INFFEED, which, by leveraging influence as feedback, attempts to simulate a pseudo-expert annotator by updating the label of a target instance. This simple approach results in significantly better performance as compared to the state-of-the-art baselines for a series of classification tasks that are subjective in nature. In the dataset extension setting, we observe that even by manually annotating $\sim \frac{1}{1000}^{\text{th}}$ of the full dataset that need

to be extended we obtain comparable performance with the scenario where all the dataset to be extended is gold-annotated. In the future, we would like to investigate if this scheme can be effectively used to replace the need for an expert annotator in a real-world deployment scenario through faster computation.

11. Ethics statement

In our research, we responsibly use social subjective data, originally published in another study and used with appropriate permissions. Acknowledging the sensitive nature of this data, we have undertaken diligent steps to maintain ethical standards. Specifically, we employed expert annotators to revisit and correct any potential misannotations, enhancing the reliability of our data. This process reinforces our commitment to upholding stringent ethical guidelines in our research.

12. Bibliographical References

Naman Agarwal, Brian Bullins, and Elad Hazan. 2016. Second-order stochastic optimization for machine learning in linear time.

Saleema Amershi, David Maxwell Chickering, Steven Mark Drucker, Bongshin Lee, Patrice Y. Simard, and Jina Suh. 2015. Modeltracker: Redesigning performance analysis tools for machine learning. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in NLP. pages 1–7, Berlin, Germany. Association for Computational Linguistics.

Samyadeep Basu, Philip Pope, and Soheil Feizi. 2020. Influence functions in deep learning are fragile.

Y. Bengio, T. Deleu, N. Rahaman, R. Ke, S. Lachapelle, O. Bilaniuk, A. Goyal, and C. Pal. 2020. A meta-transfer objective for learning to disentangle causal mechanisms. In *8th International Conference on Learning Representations (ICLR)*.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Collin Burns, Jesse Thomason, and Wesley Tansey. 2020. Interpreting black box models via hypothesis testing. ACM.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.

Costanza Conforti, Jakob Berndt, Mohammad Taher Pilehvar, Chryssi Giannitsarou, Flavio Toxvaerd, and Nigel Collier. 2020. Will-they-won't-they: A very large dataset for stance detection on twitter.

Anupam Datta, Shayak Sen, and Yair Zick. 2016. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 598–617.

Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial bias in hate speech and abusive language detection datasets. pages 25–35, Florence, Italy. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Upol Ehsan, Pradyumna Tambwekar, Larry Chan, Brent Harrison, and Mark Riedl. 2019. Automated rationale generation: A technique for explainable ai and its effects on human perceptions.

Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2018. A unified deep learning architecture for abuse detection.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Bryce Goodman and Seth Flaxman. 2017. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5).

Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. 2021. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data.

Frank R. Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

Xiaochuang Han and Yulia Tsvetkov. 2020. Fortifying toxic speech detectors against veiled toxicity.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020a. Explaining black box predictions and unveiling data artifacts through influence functions.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020b. Explaining black box predictions and unveiling data artifacts through influence functions.

Degan Hao, Lei Zhang, Jules Sumkin, Aly Mohamed, and Shandong Wu. 2020a. Inaccurate labels in weakly-supervised deep learning: Automatic identification and correction and their impact on classification performance. *IEEE Journal of Biomedical and Health Informatics*, 24(9):2701–2710.

Degan Hao, Lei Zhang, Jules Sumkin, Aly Mohamed, and Shandong Wu. 2020b. Inaccurate labels in weakly-supervised deep learning: Automatic identification and correction and their impact on classification performance. *IEEE Journal of Biomedical and Health Informatics*, 24(9):2701–2710.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Siwon Kim, Jihun Yi, Eunji Kim, and Sungroh Yoon. 2020. Interpretation of nlp models through input marginalization.

Sosuke Kobayashi, Sho Yokoi, Jun Suzuki, and Kentaro Inui. 2020. Efficient estimation of influence of a training instance. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 41–47, Online. Association for Computational Linguistics.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1885–1894. PMLR.

Shuming Kong, Yanyan Shen, and Linpeng Huang. 2022. Resolving training biases via influence-based data relabeling. In *International Conference on Learning Representations*.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning.

Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. 2020. FIND: Human-in-the-Loop Debugging Deep Text Classifiers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 332–348, Online. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model.

Zachary C. Lipton. 2016. The mythos of model interpretability.

Zachary C. Lipton and Jacob Steinhardt. 2018. Troubling trends in machine learning scholarship.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions.

James Martens. 2010. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 735–742, Madison, WI, USA. Omnipress.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17):14867–14875.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. A dataset for detecting stance in tweets. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3945–3952, Portorož, Slovenia. European Language Resources Association (ELRA).

Maximilian Mozes, Tolga Bolukbasi, Ann Yuan, Frederick Liu, Nithum Thain, and Lucas Dixon. 2023. Gradient-based automated iterative recovery for parameter-efficient tuning.

Kwabena Nuamah and Alan Bundy. 2020. Explainable inference in the frank query answering system. In *European Conference on Artificial Intelligence*.

Silviu Oprea and Walid Magdy. 2020. iSarcasm: A dataset of intended sarcasm. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1279–1289, Online. Association for Computational Linguistics.

Nicolas Papernot and Patrick McDaniel. 2018. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning.

Nazneen Fatema Rajani, Ben Krause, Wengpeng Yin, Tong Niu, Richard Socher, and Caiming Xiong. 2020. Explaining and improving model behavior with k nearest neighbor representations.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016a. "why should i trust you?": Explaining the predictions of any classifier.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016b. "why should i trust you?": Explaining the predictions of any classifier.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020. An investigation of why overparameterization exacerbates spurious correlations. In *ICML*, pages 8346–8356.

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

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3145–3153. JMLR.org.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps.

Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. 2009. Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies*, 67(8):639–662.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.

Stefano Teso, Andrea Bontempelli, Fausto Giunchiglia, and Andrea Passerini. 2021. Interactive label cleaning with example-based explanations. In *Advances in Neural Information Processing Systems*.

Stefano Teso and Kristian Kersting. 2019. Explanatory interactive machine learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, page 239–245, New York, NY, USA. Association for Computing Machinery.

Daniel Ting and Eric Brochu. 2018. Optimal subsampling with influence functions. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana. Association for Computational Linguistics.

Tianyang Wang, Jun Huan, and Bo Li. 2018. Data dropout: Optimizing training data for convolutional neural networks.

Zifeng Wang, Hong Zhu, Zhenhua Dong, Xiuqiang He, and Shao-Lun Huang. 2020. Less is better: Unweighted data subsampling via influence function. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Jincheng Xu and Qingfeng Du. 2020. On the interpretation of convolutional neural networks for text classification. In *European Conference on Artificial Intelligence*.

Sanjay Yadav and Sanyam Shukla. 2016. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pages 78–83.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online. Association for Computational Linguistics.

Hugo Zylberajch, Piyawat Lertvittayakumjorn, and Francesca Toni. 2021. HILDIF: Interactive debugging of NLI models using influence functions. In *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing*, pages 1–6, Online. Association for Computational Linguistics.