Human-in-the-Loop Synthetic Text Data Inspection with Provenance Tracking

Hong Jin Kang*[1], Fabrice Harel-Canada*[1], Muhammad Ali Gulzar[2], Violet Peng[1], and Miryung Kim[1]

[1]University of California, Los Angeles
[2]Virginia Tech

hjkang@cs.ucla.edu fabricehc@cs.ucla.edu

Abstract

Data augmentation techniques apply transformations to existing texts to generate additional data. The transformations may produce low-quality texts, where the meaning of the text is changed and the text may even be mangled beyond human comprehension. Analyzing the synthetically generated texts and their corresponding labels is slow and demanding. To winnow out texts with incorrect labels, we develop INSPECTOR, a human-in-the-loop data inspection technique. INSPECTOR combines the strengths of provenance tracking techniques with assistive labeling. INSPECTOR allows users to group related texts by their *transformation provenance*, i.e., the transformations applied to the original text, or *feature provenance*, the linguistic features of the original text. For assistive labeling, INSPECTOR computes metrics that approximate data quality, and allows users to compare the corresponding label of each text against the predictions of a large language model. In a user study, INSPECTOR increases the number of texts with correct labels identified by $3\times$ on a sentiment analysis task and by $4\times$ on a hate speech detection task. The participants found grouping the synthetically generated texts by their common transformation to be the most useful technique. Surprisingly, grouping texts by common linguistic features was perceived to be unhelpful. Contrary to prior work, our study finds that no single technique obviates the need for human inspection effort. This validates the design of INSPECTOR which combines both analysis of data provenance and assistive labeling to reduce human inspection effort.

1 Introduction

Data augmentation techniques to generate additional training data by transforming existing data can help improve model performance and robustness. However, low-quality texts with garbled text and inappropriate labels may be generated. Figure 1 shows examples of high- and low-quality instances after a transformation is applied. Despite users' inclination to filter out texts of low quality with inappropriate labels, effective debugging of the generated content remains challenging due to the opaqueness of these techniques and the sheer volume of data produced. Investigating the data instances one by one would be extremely demanding and slow.

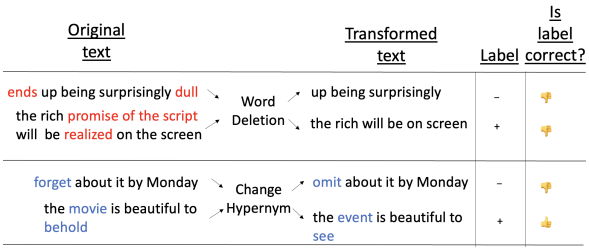

Figure 1: Examples of transformed texts from the SST2 movie review dataset generated during data augmentation. A transformed text can contain garbled text, or have an inappropriate label. As an example, the "Word Deletion" transformation can mangle the text "ends up being surprisingly dull" into "up being surprising", causing its corresponding label "-" (indicating a negative sentiment) to no longer be appropriate. Of the four examples of synthetically generated texts, only one ("the event is beautiful to see") has an appropriate label.

We propose a human-in-the-loop approach, INSPECTOR, for inspecting generated texts to weed out texts with incorrect labels. For reducing human effort, INSPECTOR applies provenance tracking, inspired by work in the database community (Wang et al., 2015), and assistive labeling. INSPECTOR supports analysis of the provenance of each text in two ways. First, INSPECTOR allows users to group the texts by their *transformation provenance*, i.e., the common transformations that have been applied to produce the text. Second, INSPECTOR allows texts to be grouped by their *feature provenance*, i.e., common linguistic features, e.g., if the text contains a negation, obtained from the relations represented in Abstract Meaning Representation

*Authors contributed equally.

Findings of the Association for Computational Linguistics: NAACL 2024, pages 3118–3129

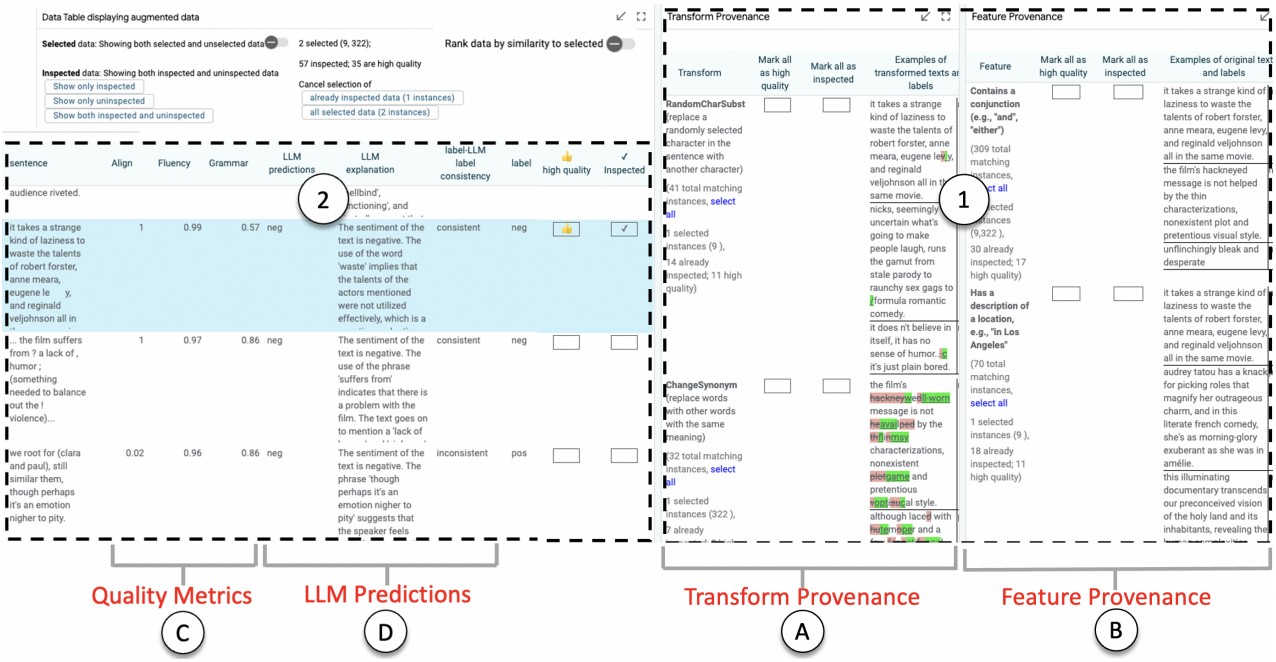

Figure 2: INSPECTOR: The user alternatives between (1) inspecting the provenance of groups of texts and labels following their (A) common transformation, and (B) common linguistic features, and (2) inspecting individual transformed texts with their corresponding labels, with assistive labeling using (C) the quality metrics, alignment, grammaticality, fluency scores, and (D) LLM predictions.

graphs (Banarescu et al., 2012). Provenance tracking allows the inspection of groups of texts with the same applied transformations or underlying feature. For example, the texts in Figure 1 transformed by *WordDeletion* share a common transformation and can be grouped together by INSPECTOR.

For assistive labeling, INSPECTOR provides two techniques. INSPECTOR computes quality metrics, such as label alignment, grammaticality, and fluency, for each generated text and corresponding label. Finally, INSPECTOR provides the predictions from a large language model that users can compare against the corresponding labels of the texts to find discrepancies.

To evaluate INSPECTOR, we ran a within-subject user study with 15 participants. The participants weeded out generated texts with inappropriate labels on two datasets, a sentiment analysis dataset (Socher et al., 2013) and a hate speech detection dataset (Barbieri et al., 2020). We build a baseline by disabling the provenance tracking and assistive labeling features. Participants using INSPECTOR identified 3x and 4x more texts with correct labels (Welch's t-test: $p < 0.005$). Using INSPECTOR, participants were more confident in identifying texts with correct labels, and adopted systematic inspection strategies. The human-selected texts and labels improve model robustness more than randomly sampled data, demonstrating the value of human inspection. No single technique of INSPECTOR was useful to every participant, suggesting that effective inspection of generated texts requires combining complementary techniques.

In summary, INSPECTOR is an approach for inspecting generated texts and corresponding labels using a novel technique for grouping texts by their provenance. INSPECTOR also offers assistive labeling techniques. Our tool is open source (UCLA-SEAL, 2023). A within-subject user study shows that using INSPECTOR enables more effective inspection and that users found grouping texts by their transformation provenance to be the most useful feature. The human-inspected data improves model robustness by up to 32%. We find that no single technique, including LLM-based assistive labeling (Gilardi et al., 2023; Wang et al., 2021), takes away the need for human inspection of generated texts.

2 Design Goals and System Overview

2.1 Design Goals

Acquiring data is a bottleneck for machine learning (Paleyes et al., 2022), but data labeling is not a simple task (Hansen et al., 2013; Kulesza et al., 2014; Chang et al., 2017). In particular, ensuring the quality of data is critical (Liang et al., 2022;

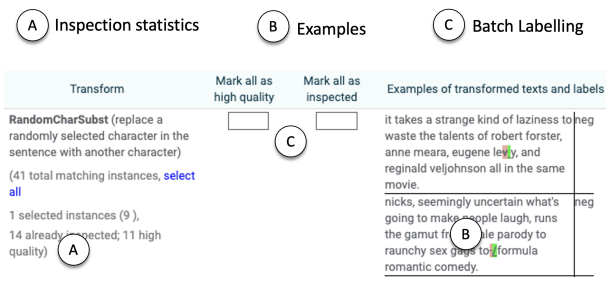

Figure 3: Transform provenance. A user selects texts and inspects the common transforms (e.g., `RandomCharSubset`) in the transformation provenance pane with their (A) inspection statistics (e.g., the user has inspected 14 texts, with 11 marked as high quality), and (B) view other texts sharing the same transform. A user can then (C) mark all instances sharing the same transform to be correct, obviating the need for inspecting individual texts one by one.

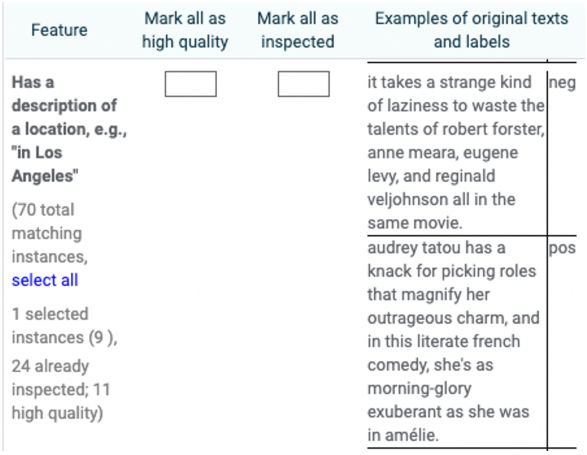

Figure 4: Feature provenance. A user can select texts, and can inspect linguistic features common to the selected texts (e.g., "Has a description of a location") in the transformation provenance pane with their inspection statistics (e.g., the user has inspected 24 texts, with 11 marked as high quality). Then, a user can mark all instances sharing the same feature to be correct.

Sambasivan et al., 2021; Paleyes et al., 2022). We identify challenges for inspecting generated texts:

Scale. Data annotation aims to obtain as much data at the lowest possible cost (Wang et al., 2022). Inspection of generated texts shares a similar goal: INSPECTOR should empower users to identify large subsets of texts with correct labels.

Evaluating each instance. Human inspection is difficult because the data-generating processes are opaque. Moreover, users may not enjoy trivial labeling tasks (Cakmak et al., 2010) and may clean data without rigor (Krishnan et al., 2016). Hence, INSPECTOR should discourage ad-hoc inspection and respect human cognition by providing more support for analyzing each text and discovering systematic insights.

2.2 Overview

Workflow. INSPECTOR supports the workflow shown in Figure 2. Through INSPECTOR, users inspect the generated texts and identify texts with correct labels, which are retained in the dataset. INSPECTOR enables inspection of the provenance (the two panes (2) in Figure 2, Section 3.2), and assistive labeling (on each generated text and corresponding label (the table (1)) in Figure 2, Section 3.3). For grouping generated texts using provenance tracking, INSPECTOR offers information about (1) the transformations applied to the selected data in the Transformation Provenance pane, and (2) the linguistic features present in the text before transformations were applied in the Feature Provenance pane.

For assistive labeling, INSPECTOR provides (1) computed quality metrics (i.e. grammaticality, fluency, and label alignment) for each instance, and (2) the predictions of a large language model.

We envision that a user of INSPECTOR alternates between the inspection of common transformations and features, forming hypotheses about root causes of quality, and the inspection of individual instances with the help of assistive labeling.

2.3 Usage Scenario.

Suppose that Alice is a model developer who wishes to expand a training dataset. Alice turns to data augmentation but is cynical. She previously observed that these techniques produce large quantities of data with the majority of texts garbled or have unsuitable labels. Alice would not trust a model trained with poor quality data (after all, *garbage in, garbage out*). As she wishes to retain only texts with correct labels but finds going through all instances onerous, Alice gives INSPECTOR a try.

Alice provides the original training dataset to INSPECTOR. INSPECTOR applies the data augmentation techniques to generate data. Next, Alice inspects the texts (Figure 2). First, she inspects several individual instances. As she marks their quality, she hypothesizes that the quality of the texts are influenced by a linguistic feature (e.g., in sentiment analysis, deleting a single negation feature in "I do **not** like stand-up." inverts the sentiment of the text), or a transformation that performs

only small-scale modifications. To assess these hypotheses, she selects several instances of interest. Now, the provenance panes display information about the transformations and underlying features of the selected instances (Figure 3).

Alice analyzes the common transformations and underlying linguistic features. Each common transformation and feature has examples of data sharing the same provenance pattern, enabling Alice to understand the transformation or feature. INSPECTOR tracks and displays inspection statistics such as the total number of inspected records and the proportion of records annotated as having suitable labels. As Alice continues to mark data, these statistics are updated. The frequent updates allow Alice to establish trust that the information displayed is meaningful (Lee and See, 2004; Dudley and Kristensson, 2018). From the small proportion of texts with correct labels among the inspected data (in Figure 4), Alice dismisses her hypothesis regarding the linguistic feature. However, the same statistic for a common transform in the transformation provenance pane (Figure 3) appears to support the hypothesis regarding a transformation that rarely distorts the text.

Alice deepens her investigation. She filters the generated texts (Figure 6) to show only texts sharing the common transformation, but there is still too much data. Alice reorders the texts by their grammaticality. She finds that even the lowest-scoring instance has a sufficiently high score (e.g., score > 0.8), suggesting that the texts are never too distorted. Skimming through the texts, she notices that the majority of them have labels that are consistent with the large language model's predictions, suggesting that the transformation usually preserves semantics. With her hypothesis validated, she marks all instances in the group. She notices that some instances with incorrect labels have poor fluency scores. She reorders the texts by their fluency and unmarks the instances with low fluency scores.

Having identified a strategy of finding common transformations among high-quality instances and assessing the grouped instances with the large language model's predictions and quality scores, Alice continues until she believes she has enough data for training the model.

3 Implementation

3.1 Overview

Figure 5 shows an overview of how a user interacts with INSPECTOR. When a user has selected several texts, INSPECTOR displays both the synthetically generated texts, as well as their common transformations and features (Section 3.2). For each individual text instance, INSPECTOR computes quality metrics and displays the predictions of a large language model (Section 3.3).

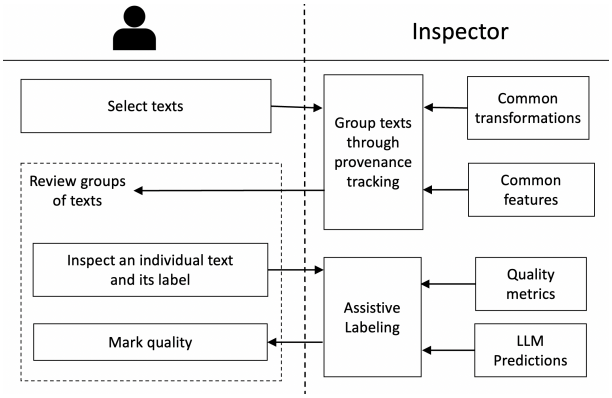

Figure 5: The workflow of a user inspecting data using INSPECTOR.

3.2 Provenance Tracking

The Transform Provenance and Feature Provenance Panes (Figure 3 and Figure 4) surfaces common transformations and linguistic features in the provenance of the user-selected generated texts. Providing details about the transforms and features, INSPECTOR allows users to inspect groups of texts sharing either the same transformations or linguistic features.

The Transformation Provenance pane summarizes recurring patterns in the transformations applied to generate the data. This allows the investigation of root causes of poor quality related to the text transformations, e.g., a random word deletion. The Feature Provenance pane displays common linguistic features (extracted from Abstract Meaning Representation graphs (Banarescu et al., 2012)) in the texts before they were transformed. This allows the investigation of possible root causes of low quality stemming from features, e.g., negations ("not"), in original texts.

The inspection statistics are presented for each common transform or feature (in the example in Figure 3, the user has inspected 14 texts in total, with 11 of them marked as high quality), enabling

the user to make generalizations about the group of data. Examples of other instances in the group are presented, with the transformed parts of the text highlighted to focus human attention, which may increase labeling efficiency (Choi et al., 2019). Users can filter data to view other instances with a shared transformation or feature provenance.

Batch inspection. Users can mark a batch of instances (Ashktorab et al., 2021) on a group of data (e.g., all data produced by the same transformation type). This enables greater inspection efficiency by applying the same decision across the group.

3.3 Assistive labeling

Figure 6 shows the table containing the transformed texts and their corresponding labels. INSPECTOR presents the following quality metrics:

1. Fluency, measured through language perplexity using GPT-2 (Radford et al., 2019),
2. Grammaticality, measured as the degree to which the text is free from grammar errors using `language-tool` (lan, 2023),
3. Label Alignment, measured through predictions of label quality (Northcutt et al., 2021).

While Fluency and Grammaticality are measures of linguistic quality, Label Alignment measures label quality using CleanLab (cleanlab.ai, 2023), a method for identifying mislabelled data (Northcutt et al., 2021). We normalize each score such that they range between 0 and 1, with 0 indicating the lowest quality and 1 the highest. INSPECTOR allows the texts to be sorted by these metrics.

INSPECTOR provides information about the outputs of a large language model (*gpt-3.5-turbo*) prompted to predict the transformed texts' labels: 1) its predictions, 2) explanations of each prediction, 3) consistency of its predictions with the instances' labels. Viewing predictions from the LLM, which may have human-level performance (Gilardi et al., 2023; Wang et al., 2021), can increase labeling efficiency (Lai and Tan, 2019; Desmond et al., 2021) and explanations contextualizing each prediction increases the user's trust in them (Bansal et al., 2021). Inconsistencies between labels and LLM predictions may imply altered semantics after a text transformation. Together with the quality metrics, the outputs of the LLM provide evidence for users to make decisions, guiding them away from ad-hoc assessments.

Table 1: The text transformations for generating data in the user study. For example, given the text "ends up being surprisingly dull" with a "negative" label, "Word Deletion" produces a new text "up being surprising" with the same corresponding label.

Category	Transformation
Swap	ChangeHypernym, ChangeHyponym, ChangeLocation, ChangeName, ChangeNumber, ChangeSynonym, RandomSwap, RandomSwapQwerty
Punctuation	ContractContractions, ExpandContractions, InsertPunctuationMarks
Typos	HomoglyphSwap, WordDeletion, RandomCharDel, RandomCharInsert, RandomCharSubst, RandomCharSwap
Text Insert	RandomInsertion
Emojis	AddNeutralEmoji, RemoveNeutralEmoji

4 User Study

We conducted a within-subject study with 15 participants to assess the effectiveness of INSPECTOR for weeding out low-quality texts. We developed INSPECTOR as a web application. As a baseline, we developed a variant of INSPECTOR without the effort-reduction techniques. We investigate the following questions:

1. Does INSPECTOR increase efficiency in identifying texts with correct labels?
2. How useful is each effort reduction technique offered by INSPECTOR?
3. Are models more robust when trained using data identified using INSPECTOR?

4.1 Study Design

Tasks. The study involves two datasets, the sentiment analysis dataset, SST2 (Socher et al., 2013), and a hate speech detection dataset, TweetEval (Barbieri et al., 2020), described in Table 2. Table 1 shows the list of considered text transformations (Morris et al., 2020; Ribeiro et al., 2020; Karimi et al., 2021; Wei and Zou, 2019; Harel-Canada et al., 2022). For each task completed by the user, we finetune BERT (Devlin et al., 2018) on the identified high-quality data for up to 10 epochs.

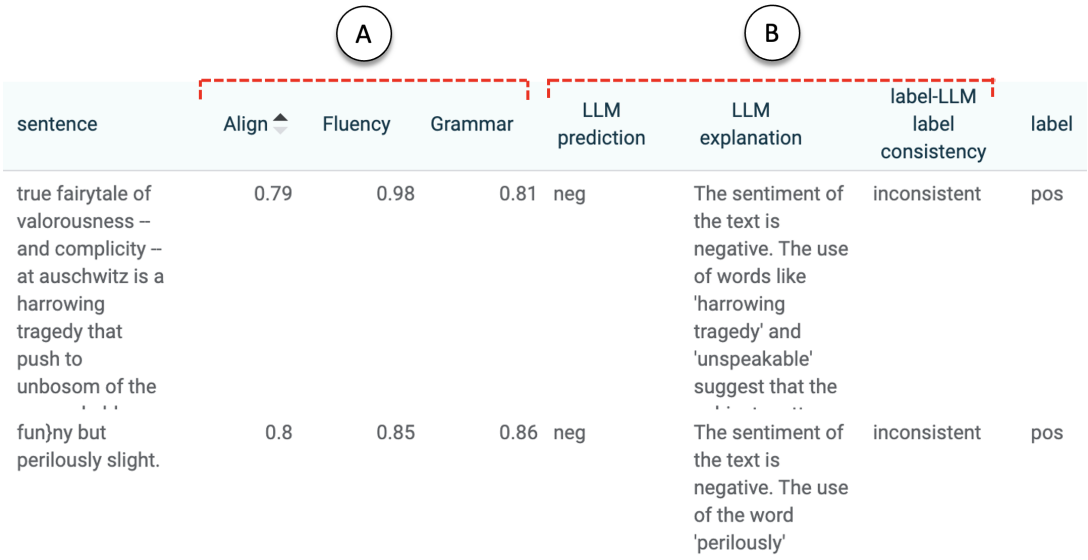

Figure 6: INSPECTOR enables assistive labeling by providing (A) quality metrics such as label alignment, fluency, and grammaticality. Label alignment is a measure of label quality, while fluency and grammaticality are measures of linguistic quality. It also shows the (B) LLM predictions for a text compared to its corresponding label.

Table 2: The two datasets in the user study. We analyze only the synthetic texts generated using the data augmentation techniques. During the course of the user study, 100% and 99% of the SST2 and TweetEval datasets were reviewed by at least one participant.

Dataset	Description	Size
SST-2	Predict the sentiment of a movie review.	613
TweetEval (Hate)	Predict if a tweet contains offensive discourse.	763

Participants. We recruited 15 participants by reaching out to students in the Computer Science department. 11 of them are Ph.D. students, 2 are master students, and 2 are undergraduates. 7 participants had less than 1 year of machine learning experience, 2 had about one year, 2 had 2-5 years of experience, and 2 had more than 5 years of experience. The participants also self-reported their familiarity with inspecting machine learning datasets and understanding mislabelled data on a 7-point Likert scale. The mean familiarity was 3.4, where 1 is "Most unfamiliar" and 7 is "Most familiar". This level of experience is identical to data annotators in industry, who do not have a machine learning background (Wang et al., 2022).

Study Protocol. Our study involves two datasets and two tools. We design a within-subjects study where each participant investigates both datasets and experiences using both tools. For each task, the participant used only the assigned tool, either INSPECTOR or Annotator, the variant of INSPECTOR without the effort reduction techniques. The study requires the completion of 2 tasks, with each task requiring 20 minutes at most. We design our study to be completed in 60 minutes. Before the users started on a task, we asked for the participants' consent to record their usage of the tools. Then, they spent 10 minutes working through a tutorial and warm-up questions. After completing the tasks, the participants were directed to a post-study questionnaire to share their experiences and feedback about the tools. We also solicited responses about the participants' inspection strategy.

5 Results

In this section, we report and analyze the results of our user study. We denote each participant as P#.

5.1 Reduction in Human Effort

To assess inspection efficiency, we counted the number of texts with correct labels identified by the participants. Table 3 shows that using INSPECTOR leads to 3x and 4x more texts on the sentiment analysis dataset and the hate speech detection dataset. Using INSPECTOR, participants identified an average of 277 and 259 high-quality instances compared to 82 and 63 instances using the baseline on the SST2 and TweetEval's Hate Speech dataset, respectively. The differences in efficiency is significant (Welch's t-test, $p < 0.005$).

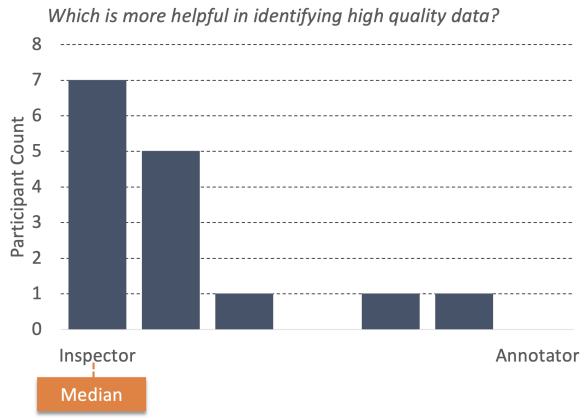

Figure 7: Most participants perceived INSPECTOR to be more helpful in winnowing out generated texts with incorrect labels (such as the examples in Figure 1).

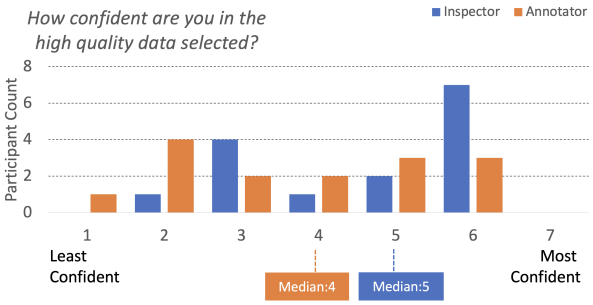

Figure 8: Participants using INSPECTOR were more confident of their inspections, with a median rating of 5 compared to the baseline of 4.

Table 3: The average number of texts with correct labels identified. INSPECTOR improves over a baseline without provenance tracking and assistive labeling by 3× and 4×. SA: Sentiment Analysis, HS: Hate Speech Detection

Approach	SA	HS
INSPECTOR w/o provenance tracking and assistive labeling	82	63
INSPECTOR	277	259

Figure 7 shows that the participants found INSPECTOR more useful and were more confident of their results. 13 of the 15 participants indicated that they preferred using INSPECTOR over the baseline tool. Participants had a higher median level of confidence in their inspections using INSPECTOR (5 vs 4 on a 7-point scale, in Figure 8).

The techniques in INSPECTOR changed the participants' perception of the task's nature. P3 indicated that INSPECTOR "could give initial results. I'm more acting like a verifier". They found INSPECTOR more usable in identifying patterns. Comparing INSPECTOR and the baseline, 13 of the

Table 4: Usefulness of the techniques of INSPECTOR rated by the participants (out of 7). A higher rating indicates the technique was perceived to be more useful for inspection. Grouping texts by common transformations was perceived to be the most useful technique.

Technique	Average Rating
Transform Provenance	4.5
Quality Metrics (grammaticality, fluency, & alignment)	4.3
LLM Guidance	4.3
Feature Provenance	1.9

15 participants found INSPECTOR more helpful in identifying patterns for inspecting the texts. They reported using the techniques of INSPECTOR in their inspection strategies. P8 wrote "My main strategy was to find inconsistent patterns in already labeled data and LLM" while P1 mentioned that "Sentences with a grammar score < 0.92 are almost always low-quality.". Conversely, using the baseline, participants found strategically inspecting the data difficult. P8 wrote that dissecting the data "is utterly not possible" and "Using `ctrl+f` was really painful."

5.2 User Ratings of Individual Features

In the post-study questionnaire, participants rated the techniques of INSPECTOR and described how they inspected the data. Table 4 summarizes the participants' feedback. Grouping data by transformation provenance was the most appreciated technique, followed by the assistive labeling techniques. On a 7-point Likert scale, participants assigned Transform Provenance the highest average rating of 4.5, followed by the two assistive labeling techniques with average ratings of 4.3. P8 indicated that using transform provenance was the "main strategy I used to identify trends". Participant C7 wrote that provenance tracking allows her to "reason about whether a specific transformation can lead to a reduction of data quality".

Diverse inspection strategies. The responses to the post-study questionnaire reflected a wide range of strategies. No single technique of INSPECTOR was found to be useful by every participant. We qualitatively analyze the free responses. A majority (11) of the participants used the transformation provenance to make decisions. 9 participants described using the quality metrics (grammaticality, fluency, label alignment), and 8 participants men-

Table 5: Results comparing the robustness of BERT after finetuning on additional data. We measure the attack success rate of DeepWord (the lower the better). SA: Sentiment Analysis, HS: Hate Speech Detection

Approach	SA	HS
Randomly sampled	0.61	0.50
Human-guided	0.59	0.34

tioned using outputs of the large language model. All participants found at least one technique of INSPECTOR to be useful (rating at least 4 out of 7 and using it in their workflow). This suggests no one technique obviates the need for human inspection.

Batch Inspection. A majority (11 of the 15) of the participants used the batch inspection feature, marking an average of 5 groups. The participants appreciated batch inspection and used it after manual validation of a few representative instances. P13 wrote "if most of the labels are not affected by the transformation and are indeed high-quality by manual inspection, I mark all such entries as high-quality.". However, not all participants found grouping the texts by their provenance useful or meaningful. Two participants applied a strategy of trying to go through the data one by one. P11 wrote that she did not consider data provenance for grouping the texts "in order to give an unbiased opinion" for each text and its label.

5.3 Robustness

Next, we assess the improvements in model robustness when training a model with the texts with correct labels collected by the participants. As a baseline, we construct a randomly sampled dataset with the average number of data instances selected using INSPECTOR. We assess the robustness of the model by measuring the attack success rate of DeepWord (Gao et al., 2018), a method of generating adversarial attacks. Using the implementation of DeepWord in TextAttack (Morris et al., 2020), we generated 100 attacks on each model. More robust models would face fewer successful attacks.

On the SST2 dataset, 4 out of 8 participants marked data that led to more robust models than randomly sampled data. On the TweetEval dataset, all 7 participants identified data that led to more robust models than randomly sampled data. The attack success rate of DeepWord on models trained with randomly selected data was 0.61 on the SST2 dataset and 0.5 on the TweetEval dataset. Using the inspected data, the attack success rate decreases to an average of 0.59 and 0.34 on SST2 and TweetEval, respectively. On TweetEval, this corresponds to a 32% improvement. Overall, the texts identified using INSPECTOR improved model robustness.

6 Implications

Our study showed that INSPECTOR empowered users to be effective and confident in inspecting transformed texts and their corresponding labels.

Transformation provenance was the most useful technique. The participants perceived transformation provenance to be the most useful technique provided by INSPECTOR. Even when participants did not use it to perform batch inspection, they found the additional information useful. P2 considered this information to inspect data, writing that it "showed whether a malformed sentence was in the original text or mangled by a transformation.".

Assistive labeling helped users build trust. The participants were aware of the risk of including incorrect labels when labeling a batch of texts. Thus, the users had to trust the guidance and automation provided by INSPECTOR. Users were able to build trust using the inspection statistics of the groups of generated texts. The assistive labeling techniques were also helpful for building confidence. P8 wrote "When the fluency and grammaticality scores are both high, I am more confident to label the data as high quality."

The linguistic features were perceived to be ineffective. Surprisingly, users found grouping the texts by their common linguistic features to be ineffective. The participants did not always understand their relevance to the task. P14 wrote "the features are very low level and it is not clear how they are related to the labeling quality".

Limited impact on inspection accuracy. We observed that users tended to only mark texts they were confident about. Comparing the participants' inspections to the ground-truth labels annotated by one of the authors of this paper, we find that the use of INSPECTOR only had a limited impact on the accuracy of the user's inspections. On SST2, INSPECTOR slightly decreased labeling accuracy from 88% to 85.3%. On TweetEval, INSPECTOR increased accuracy from 89.1% to 90.0%. These differences are not statistically significant (p-value > 0.05). This suggests that users of INSPECTOR inspected more texts with less effort and increased confidence while maintaining the same level of accuracy.

7 Related Work

7.1 Debugging Machine Learning

Testing models. DynaBench (Kiela et al., 2021) evaluates models with challenging human-provided data. AdaTest (Ribeiro and Lundberg, 2022) generates more test cases similar to user-provided tests. These studies focus on models but do not support debugging data, which is the focus of INSPECTOR.

Grouping data for data inspection. SliceFinder (Chung et al., 2019) and What-If (Wexler et al., 2019) are tools for understanding the subsets of data with poor model performance. Zeno (Cabrera et al., 2023) groups data on properties extracted by user-written Python programs. Tempura (Wu et al., 2020) supports data inspection by grouping data using structural templates abstracted from concrete data instances. Unlike these tools, INSPECTOR groups data by their provenance.

7.2 Cleaning Data

For cleaning tabular data, Data Civilizer (Rezig et al., 2019) is a data pipeline-debugger for identifying low-quality data (e.g., malformed values) that cause incorrect data analysis outputs through inserting breakpoints and the visualization of the intermediate records. Unlike Data Civilizer, INSPECTOR is not a breakpoint debugging tool. Instead, it helps users filter out data with unsuitable labels. Wrangler (Kandel et al., 2011) cleans dirty data by inferring data types (e.g., integers) and semantic roles (e.g., zip code) for identifying anomalous data. Potter's Wheel (Raman and Hellerstein, 2001) cleans dirty data but detecting them may require users to implement an API to define constraints (e.g., date formats). On the other hand, INSPECTOR assists human users in interactively identifying texts with incorrect labels without requiring programming literacy.

Ruler (Choi et al., 2021) and TagRuler (Evensen et al., 2020) learn rules for annotating unlabelled data based on the text. INSPECTOR guides users to weed out texts with incorrect labels. While feature provenance in INSPECTOR is similar to the token-based rules in Ruler or TagRuler, it uses linguistic features rather than tokens.

8 Conclusion

We present INSPECTOR for winnowing synthetic texts with incorrect labels generated during data augmentation. In a within-subject user study, participants using INSPECTOR were $3\times$ and $4\times$ more effective. Users found that grouping data by their shared common transformations to be the most useful technique. Assistive labeling allowed them to build trust in the tool. Surprisingly, users perceived the linguistic features to be ineffective. INSPECTOR is the first interactive human-in-the-loop approach for examining text augmentation data for classification tasks by combining provenance inspection and assistive labeling techniques.

We publicly release INSPECTOR. INSPECTOR is available at https://github.com/UCLA-SEAL/ProvenanceInspector.

9 Limitations

INSPECTOR guides human users using several techniques, including the computation of quality metrics and the use of Abstract Meaning Representation for analyzing feature provenance. Due to these dependencies, INSPECTOR is only applicable to texts in languages that are supported by the tools for computing the metrics (CleanLab, LanguageTool) and converting the text to Abstract Meaning Representation.

INSPECTOR was evaluated through a user study. Although our study was performed with student participants, the participants in our study share a similar expertise level with full-time data annotators in industry — i.e. data annotators in industry are not full-time data scientists or engineers and they generally do not have any machine learning background. Consequently, having student participants is unlikely to impact generalizability.

While we evaluated INSPECTOR on only two datasets, INSPECTOR does not use compute-intensive techniques and would work on large datasets. As the design of INSPECTOR is not specific to the tasks in the user study, we believe that our findings would generalize to other NLP tasks.

Labeling hate speech is known to be inherently ambiguous and influenced by the annotator's beliefs even when labeling guidelines are provided. INSPECTOR does not solve the issue of annotation bias.

Our work investigated only human effort in inspecting texts generated as part of data augmentation. Our insights may, therefore, be specific to data augmentation. We hope to extend this analysis to other data inspection tasks.

10 Ethics Statement

Our work aims to allow human users better insights into generated data. These data may contain toxic, offensive content, and INSPECTOR may expose these content to its users. Alone, INSPECTOR does not generate biased or offensive text, however, it postprocesses the output of data augmentation techniques which may produce harmful texts.

We obtained an exemption from the UCLA IRB to run the user studies.

Acknowledgements

This work is supported by the National Science Foundation under grant numbers 2106838, 1764077, 1956322, and 2106404. It is also supported by funding from Amazon and Samsung, the Meta Sponsor Research Award, the Okawa Foundation Research Grant, and the Amazon Alexa AI Research Award. We thank Akshay Singhal for customizing the initial INSPECTOR interface and Jiayue (Coraline) Sun for developing the early pipelines for provenance tracking and model training. We want to thank the anonymous reviewers for their constructive feedback that helped improve the work.

References

2023. languagetool-org/languagetool. https://github.com/languagetool-org/languagetool.

Zahra Ashktorab, Michael Desmond, Josh Andres, Michael Muller, Narendra Nath Joshi, Michelle Brachman, Aabhas Sharma, Kristina Brimijoin, Qian Pan, Christine T Wolf, et al. 2021. AI-assisted human labeling: Batching for efficiency without overreliance. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–27.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (AMR) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*, pages 1533–1544.

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the whole exceed its parts? the effect of AI explanations on complementary team performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–16.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650.

Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I Hong, and Adam Perer. 2023. Zeno: An interactive framework for behavioral evaluation of machine learning. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–14.

Maya Cakmak, Crystal Chao, and Andrea L Thomaz. 2010. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2(2):108–118.

Joseph Chee Chang, Saleema Amershi, and Ece Kamar. 2017. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2334–2346.

Dongjin Choi, Sara Evensen, Çağatay Demiralp, and Estevam Hruschka. 2021. Tagruler: Interactive tool for span-level data programming by demonstration. In *Companion Proceedings of the Web Conference 2021*, pages 673–677.

Minsuk Choi, Cheonbok Park, Soyoung Yang, Yonggyu Kim, Jaegul Choo, and Sungsoo Ray Hong. 2019. AILA: Attentive interactive labeling assistant for document classification through attention-based deep neural networks. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–12.

Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. 2019. Automated data slicing for model validation: A big data-AI integration approach. *IEEE Transactions on Knowledge and Data Engineering*, 32(12):2284–2296.

cleanlab.ai. 2023. cleanlab/cleanlab. https://github.com/cleanlab/cleanlab.

Michael Desmond, Michael Muller, Zahra Ashktorab, Casey Dugan, Evelyn Duesterwald, Kristina Brimijoin, Catherine Finegan-Dollak, Michelle Brachman, Aabhas Sharma, Narendra Nath Joshi, et al. 2021. Increasing the speed and accuracy of data labeling through an AI assisted interface. In *26th International Conference on Intelligent User Interfaces*, pages 392–401.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

John J Dudley and Per Ola Kristensson. 2018. A review of user interface design for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(2):1–37.

Sara Evensen, Chang Ge, Dongjin Choi, and Çağatay Demiralp. 2020. Data programming by demonstration: A framework for interactively learning labeling functions. *arXiv preprint arXiv:2009.01444*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. ChatGPT outperforms crowd-workers for text-annotation tasks. *arXiv preprint arXiv:2303.15056*.

Derek L Hansen, Patrick J Schone, Douglas Corey, Matthew Reid, and Jake Gehring. 2013. Quality control mechanisms for crowdsourcing: peer review, arbitration, & expertise at familysearch indexing. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 649–660.

Fabrice Harel-Canada, Muhammad Ali Gulzar, Nanyun Peng, and Miryung Kim. 2022. Sibylvariant transformations for robust text classification. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1771–1788.

Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the sigchi conference on human factors in computing systems*, pages 3363–3372.

Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2021. Aeda: An easier data augmentation technique for text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2748–2754.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, et al. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124.

Sanjay Krishnan, Daniel Haas, Michael J Franklin, and Eugene Wu. 2016. Towards reliable interactive data cleaning: A user survey and recommendations. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pages 1–5.

Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. 2014. Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3075–3084.

Vivian Lai and Chenhao Tan. 2019. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 29–38.

John D Lee and Katrina A See. 2004. Trust in automation: Designing for appropriate reliance. *Human factors*, 46(1):50–80.

Weixin Liang, Girmaw Abebe Tadesse, Daniel Ho, L Fei-Fei, Matei Zaharia, Ce Zhang, and James Zou. 2022. Advances, challenges and opportunities in creating data for trustworthy AI. *Nature Machine Intelligence*, 4(8):669–677.

John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. *arXiv preprint arXiv:2005.05909*.

Curtis Northcutt, Lu Jiang, and Isaac Chuang. 2021. Confident Learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411.

Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. 2022. Challenges in deploying machine learning: a survey of case studies. *ACM Computing Surveys*, 55(6):1–29.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Vijayshankar Raman and Joseph M Hellerstein. 2001. Potter's wheel: An interactive data cleaning system. In *VLDB*, volume 1, pages 381–390.

El Kindi Rezig, Lei Cao, Michael Stonebraker, Giovanni Simonini, Wenbo Tao, Samuel Madden, Mourad Ouzzani, Nan Tang, and Ahmed K Elmagarmid. 2019. Data civilizer 2.0: A holistic framework for data preparation and analytics. *Proceedings of the VLDB Endowment*, 12(12):1954–1957.

Marco Tulio Ribeiro and Scott Lundberg. 2022. Adaptive testing and debugging of NLP models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3253–3267.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912.

Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. "everyone wants to do the model work, not the data work": Data cascades in high-stakes AI. In *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank.

In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

UCLA-SEAL. 2023. UCLA-SEAL/ProvenanceInspector. https://github.com/UCLA-SEAL/ProvenanceInspector.

Ding Wang, Shantanu Prabhat, and Nithya Sambasivan. 2022. Whose AI dream? in search of the aspiration in data annotation. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–16.

Jianwu Wang, Daniel Crawl, Shweta Purawat, Mai Nguyen, and Ilkay Altintas. 2015. Big data provenance: Challenges, state of the art and opportunities. In *2015 IEEE international conference on big data (Big Data)*, pages 2509–2516. IEEE.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. *arXiv preprint arXiv:2108.13487*.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65.

Tongshuang Wu, Kanit Wongsuphasawat, Donghao Ren, Kayur Patel, and Chris DuBois. 2020. Tempura: Query analysis with structural templates. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12.