Discover, Explain, Improve: An Automatic Slice Detection Benchmark for Natural Language Processing

Wenyue Hua[1] **Lifeng Jin**[2] **Linfeng Song**[2] **Haitao Mi**[2]
Yongfeng Zhang[1] **Dong Yu**[2]
[1]Rutgers University, New Brunswick, USA [2]Tencent America, USA
[1]wenyue.hua, yongfeng.zhang@rutgers.edu
[2]lifengjin, lfsong, haitaomi, dyu@tencent.com

Abstract

Pretrained natural language processing (NLP) models have achieved high overall performance, but they still make systematic errors. Instead of manual error analysis, research on slice detection models (SDMs), which automatically identify underperforming groups of datapoints, has caught escalated attention in Computer Vision for both understanding model behaviors and providing insights for future model training and designing. However, little research on SDMs and quantitative evaluation of their effectiveness have been conducted on NLP tasks. Our paper fills the gap by proposing a benchmark named ``**D**iscover, **E**xplain, **Im**prove (DEIm)'' for classification NLP tasks along with a new SDM *Edisa*. *Edisa* discovers coherent and underperforming groups of datapoints; DEIm then unites them under human-understandable concepts and provides comprehensive evaluation tasks and corresponding quantitative metrics. The evaluation in DEIm shows that *Edisa* can accurately select error-prone datapoints with informative semantic features that summarize error patterns. Detecting difficult datapoints directly boosts model performance without tuning any original model parameters, showing that discovered slices are actionable for users.[1]

1 Introduction

While deep learning models (Kenton and Toutanova, 2019; Liu et al., 2019; Clark et al., 2020, inter alia) achieve high overall performance on many tasks, they often display systematic errors (Kayser-Bril, 2020; Stuart-Ulin, 2018; Hamilton, 2018) correlated with biases, challenging data points, and data collection issues. Investigating these errors and their associated features is crucial for understanding models' strengths and weaknesses. Although manual error analysis is typically employed for identifying biases and erroneous behaviors, its efficiency and quality are limited. Consequently, automatic slice detection models (SDMs) are motivated to streamline the analysis process by identifying systematic errors in any trained machine learning model (Eyuboglu et al., 2022; Ribeiro et al., 2020, 2016; Wu et al., 2021), based on the observation that representations of error instances may share features and thus similar to each other.

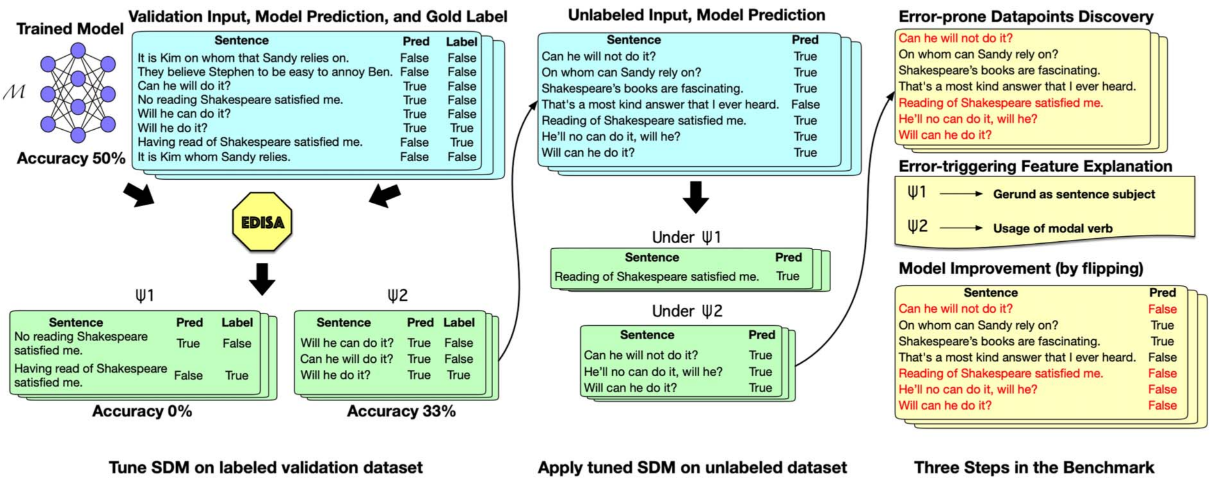
In SDMs, a slice refers to a set of datapoints sharing a specific attribute. An error slice is a slice characterized by low accuracy (Eyuboglu et al., 2022). Identifying these error slices serves three primary purposes: (1) locating error-prone datapoints to enable direct prediction adjustments, (2) gaining insights into model behavior to foster better comprehension and interpretation, and (3) guiding additional model training through strategies such as slice-specific modeling, data augmentation, and active learning. Therefore, an effective slice detection model should (1) accurately locate error-prone data points, (2) offer coherent error slices which help yield intelligible error-correlated features, and (3) enhance model performance when complemented with suitable tools.

In this study, we introduce a comprehensive benchmark that assesses SDMs with three modules: namely, Discover, Explain, and Improve, each of which corresponds to a point previously discussed. We also propose a new SDM *Edisa* to serve as a baseline on the benchmark. The usage of *Edisa* and the evaluation pipeline of the benchmark DEIm are depicted in Figure 1. Here we briefly introduce the three modules.

Discover: This module utilizes a tuned SDM to detect error-prone datapoints on any unlabeled datasets for a specific trained NLP model denoted

[1] Code and benchmark are available here: https://github.com/Wenyueh/DEIM.

Transactions of the Association for Computational Linguistics, vol. 11, pp. 1537–1552, 2023. https://doi.org/10.1162/tacl_a_00617
Action Editor: Jing Jiang. Submission batch: 6/2023; Revision batch: 8/2023; Published 12/2023.


Trained Model
$\mathcal{M}$
Accuracy 50%
Validation Input, Model Prediction, and Gold Label
EDISA
ψ1
Accuracy 0%
ψ2
Accuracy 33%
Tune SDM on labeled validation dataset
Unlabeled Input, Model Prediction
Under ψ1
Under ψ2
Apply tuned SDM on unlabeled dataset
Error-prone Datapoints Discovery
Error-triggering Feature Explanation
ψ1 → Gerund as sentence subject
ψ2 → Usage of modal verb
Model Improvement (by flipping)
Three Steps in the Benchmark

Sentence	Pred	Label
It is Kim on whom that Sandy relies on.	False	False
They believe Stephen to be easy to annoy Ben.	False	False
Can he will do it?	True	False
No reading Shakespeare satisfied me.	True	False
Will he can do it?	True	False
Will he do it?	True	True
Having read of Shakespeare satisfied me.	False	True
It is Kim whom Sandy relies.	False	False

Sentence	Pred	Label
No reading Shakespeare satisfied me.	True	False
Having read of Shakespeare satisfied me.	False	True

Sentence	Pred	Label
Will he can do it?	True	False
Can he will do it?	True	False
Will he do it?	True	True

Sentence	Pred
Can he will not do it?	True
On whom can Sandy rely on?	True
Shakespeare's books are fascinating.	True
That's a most kind answer that I ever heard.	False
Reading of Shakespeare satisfied me.	True
He'll no can do it, will he?	True
Will can he do it?	True

Sentence	Pred
Reading of Shakespeare satisfied me.	True

Sentence	Pred
Can he will not do it?	True
He'll no can do it, will he?	True
Will can he do it?	True

Can he will not do it?
On whom can Sandy rely on?
Shakespeare's books are fascinating.
That's a most kind answer that I ever heard.
Reading of Shakespeare satisfied me.
He'll no can do it, will he?
Will can he do it?

Sentence	Pred
Can he will not do it?	False
On whom can Sandy rely on?	True
Shakespeare's books are fascinating.	True
That's a most kind answer that I ever heard.	False
Reading of Shakespeare satisfied me.	False
He'll no can do it, will he?	False
Will can he do it?	False

Figure 1: An SDM (*Edisa*) takes a trained model $\mathcal{M}$ and a labeled dataset as inputs. It is tuned on inputs, predictions, and labels. The tuned SDM generates multiple low-performing slicing functions such as ψ_1 and ψ_2. Applying the SDM on any unlabeled dataset will group the datapoints based on slicing functions. The Discover module collects error-prone datapoints; The Explain module assigns features to each error slice, and the Improve module enhances model performance. Here, it simply flips the predictions of identified error-prone datapoints.

as $\mathcal{M}$. The evaluation of the discovery capability is straightforward, which verifies whether the located error-prone datapoints are indeed mispredicted by $\mathcal{M}$. In the example in Figure 1, one sentence classified to ψ_1 and three sentences classified to ψ_2 are deemed as error-prone.

Explain: This module employs linguistic tools to articulate why a model fails on a given error slice, consolidating the reasons into human-comprehensible concepts. For each identified error slice, it discerns linguistic features that occur substantially more often within the slice. These features potentially elucidate why the model inaccurately predicts these data points. In Figure 1, sentences in ψ_1 all contain gerund (verbal ending in -ing that functions as a noun) as sentence subject, indicating that it is likely to be the reason why these datapoints are mispredicted. In order to assess the cohesiveness of the discovered error slices, we evaluate measures such as homogeneity and completeness of each slice with respect to their error-correlated features.

Improve: This module showcases how model improvement is realized based on discovered error slices utilizing three techniques: selective prediction (Varshney et al., 2022b,c), flipping, and active learning. For instance, as shown in Figure 1, we invert the prediction for each identified error-prone data point. These three model improvement methods also serve as external evaluations of SDMs. To verify the usefulness of the discovered error slices, we examine whether the model's performance escalates after implementing these techniques.

In the three-module benchmark DEIM, each module concentrates on one specific application of an SDM: (1) detection of error-prone data points, (2) interpretability of error slices, and (3) improvement of model performance. Each module provides evaluation tasks with the necessary tools and quantitative metrics. Each module incorporates evaluation tasks equipped with essential tools and quantitative metrics. Experimental results on *Edisa* indicate that it can effectively identify error-prone data points in unlabeled datasets and precisely detect error-correlated features, which contribute directly to enhanced model performance.

The paper is organized as follows: Section 2 discusses recent work on slice detection models; Section 3 introduces the model structure of *Edisa* model. Section 4 presents the details of the DEIM benchmark and all relevant tools. Section 5 presents experiment results and relevant ablation studies. Section 6 concludes this paper.

2 Related Work

Explainable model predictions are crucial in various research areas. Discovering error-correlated features in datapoints both increases model performance and delivers insights into future model design. In Computer Vision (CV), research has been reported to use learned input representations to identify semantically meaningful slices where errors are made in prediction (Eyuboglu et al., 2022; d'Eon et al., 2022; Yeh et al., 2020; Sohoni et al., 2020; Kim et al., 2019; Singla et al., 2021). Eyuboglu et al. (2022) recently proposed the SOTA automatic error detection method DOMINO. In NLP, task-specific automatic error analysis research has been conducted on tasks such as document-level information extraction (Das et al., 2022), coreference resolution (Kummerfeld and Klein, 2013), and machine translation (Popović and Ney, 2011). There is also extensive research conducted on different model evaluations to see whether models make erroneous datapoints in certain types of noising datapoints (Belinkov and Bisk, 2017; Rychalska et al., 2019) or adversarial datapoints (Ribeiro et al., 2018; Iyyer et al., 2018). Another line of work including Swayamdipta et al. (2020), Xu et al. (2020), and Varshney et al. (2022a) focuses on evaluating the model-independent difficulty level of datapoints. Recently, Rajani et al. (2022) introduced an interactive visualization tool for underperforming slices using token-level features.

However, as far as we know, there has not been a comprehensive evaluation benchmark that circumvents all the aspects of SDM in NLP. Therefore in this project, we contribute to the research area by designing a benchmark DEIM for all classification tasks: It provides (1) task-independent comprehensive linguistic feature benchmark for potential explanations, (2) quantitative experiments for both error slice quality and error-prone datapoints detection efficacy in unlabeled datasets, and (3) corresponding metrics that facilitate future development. We also propose a new SDM model *Edisa* which performs fairly well to serve as the SDM baseline for DEIM benchmark in NLP field. Its simple structure and promising results show a good prospect of this field.

3 *Edisa* Model

The *Edisa* model is a new model that we proposed for slice detection in NLP. This section describes the model structure, training objective, and inference procedure of *Edisa*. Subsequently, we compare this model with the current state-of-the-at SDM model DOMINO (Eyuboglu et al., 2022) to underscore why such model structure design is necessary.

In *Edisa*, we posit the existence of a set of k interpretable slices, each distinguished by one or more crucial features that differentiate the slice from other data points. *Edisa* specifically focuses on error-correlated features, that is, features co-occurring with incorrect predictions. Thus, for the same task and dataset, the set of features and the k slices may vary with respect to different NLP models. The objective of an SDM is to identify these k slices for a trained $\mathcal{M}$ in an unsupervised manner. Ideally, the discovery of these k slices requires a sufficiently large dataset where both input information and model prediction information are accessible. We mimic this setting by providing a labeled validation dataset, aiming to identify the k slices within it.

To formally introduce the model, *Edisa* can be seen as a function g that takes in a trained NLP model $\mathcal{M}$ and a labeled dataset $\mathcal{D}$ to generate k slicing functions $\{\psi_i\}_{i=1}^{i=k}$:

$$g(\mathcal{M},\mathcal{D}) = \{\psi_i : D \times \mathcal{M} \to \{0,1\} \mid 1 \le i \le k\} \tag{1}$$

3.1 *Edisa*'s Model Structure

Edisa is an **E**rror-**dis**tance-**a**ware multivariate Gaussian mixture model that models the datapoint representation, error-distance, and model prediction (*e.g.*, confidence scores in classification tasks). The observations of one datapoint from a model $\mathcal{M}$ include three components: $\{Z, E, \mathcal{Y}\}$, where Z is an embedding representation, $\mathcal{Y}$ is predicted probabilities or confidence scores from the model, and error-distance E is the distance between the one-hot tensor of the gold label Y and $\mathcal{Y}$:

$$E = Y - \mathcal{Y} \tag{2}$$

For each datapoint, Z encodes the task-relevant semantic information; E encodes both label information and confidence information, which represents whether the prediction is wrong, to what extent it deviates from the gold label, and how much change is still required to make a correct prediction; $\mathcal{Y}$ encodes the confidence score, which is added to the model to control the weights of label information and of confidence information.[2]

[2] Future work can replace confidence with calibrated confidence to improve the model (Yu et al., 2011; Guo et al., 2017; DeVries and Taylor, 2018; Kumar et al., 2018, inter alia.), since calibrated confidence usually presents a better probability estimate of the likelihood for a datapoint to be categorized in some class.

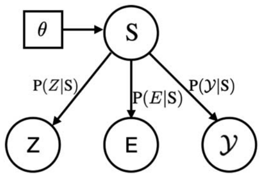


Figure 2: Model structure.

We perform PCA on representations to filter out redundant information before applying the SDM. Figure 2 illustrates the model structure.

The generative story of the *Edisa* model is as follows: In order to generate all the observations of one datapoint, one slice S_j is first drawn from a categorical distribution with parameters θ. Then, the embedding Z, the error-distance E and model prediction $\mathcal{Y}$ are drawn from slice-specific Gaussian distributions with parameters $\{\mu_j^Z, \Sigma_j^Z\}, \{\mu_j^E, \Sigma_j^E\}$, and $\{\mu_j^{\mathcal{Y}}, \Sigma_j^{\mathcal{Y}}\}$, respectively.

$$\begin{aligned} S_j &\sim P(S;\theta) \\ Z|S_j &\sim \mathcal{N}(\mu_j^Z, \Sigma_j^Z) \\ E|S_j &\sim \mathcal{N}(\mu_j^E, \Sigma_j^E) \\ \mathcal{Y}|S_j &\sim \mathcal{N}(\mu_j^{\mathcal{Y}}, \Sigma_j^{\mathcal{Y}}) \end{aligned} \tag{3}$$

For each datapoint d, the joint likelihood of slice S_j and the observations of d is a weighted product of the likelihoods from all distributions in the model with weights $\gamma, \lambda_E, \lambda_{\mathcal{Y}}$ on the Gaussians:

$$L(d, S_j) = P(S_j)P(Z_d|S_j)^{\gamma} P(E_d|S_j)^{\lambda_E} P(\mathcal{Y}_d|S_j)^{\lambda_{\mathcal{Y}}} \tag{4}$$

Given the joint likelihoods, the conditional probability of slice assignment $P(S_j|d)$ for the datapoints can be computed as:

$$P(S_j|d) = \frac{L(d, S_j)}{\Sigma_{j=1}^{j=k} L(d, S_j)} \propto L(d, S_j) \tag{5}$$

Semantic information in the embedding, the error-distance, and the model predictions together determine the slice distribution. Thus datapoints that share some similar semantic features with the same gold label and similar model predictions are encouraged to be clustered into one slice. Given the joint likelihood, each slicing function ψ_j is defined such that $\forall d \in \mathcal{D}, \psi_j(d) = 1$ if and only if:

$$\text{argmax}_i L(d, S_i) = j \tag{6}$$

3.2 Train

The model parameters are estimated with Expectation-Maximization by maximizing the sum of log-likelihood of all datapoints $d \in \mathcal{D}$ in each slice S_j for $j \in \{1, \ldots, k\}$:

$$\mathcal{L}(\mathcal{D}) = \Sigma_{i=1}^{i=|\mathcal{D}|} \log \Sigma_{j=1}^{j=k} L(d_i, S_j) \tag{7}$$

in which the assignment likelihood and the model parameters are estimated iteratively. *Edisa* is tuned using the embeddings, error-distances, and confidence scores from the validation dataset of a task after $\mathcal{M}$ has been trained on the training dataset.

A slice S_j is defined as an error slice, denoted as S_j^e if the accuracy of $\{d \in \mathcal{D} | d \in S_j\} < \delta$ for some threshold $\delta \in \mathbb{R}$. We call slicing functions corresponding to error slices as error slicing functions, denoted as ψ_j^e corresponding to S_j^e.

3.3 Inference

For inference, we apply the tuned *Edisa* to test datasets $\mathcal{T}$ where gold labels are unknown to the model. Since gold label information is not available, the error-distance needs to be marginalized over potential label values. Thus the joint likelihood of a test datapoint $t \in \mathcal{T}$ and slice S_j is computed as below, where E_t' ranges over all possible E values:

$$\begin{aligned} L(t, S_j) =& P(S_j)P(Z_t|S_j)^{\gamma}(\Sigma_{E_t'} P(E_t'|S_j)^{\lambda_E}) \\ & \cdot P(\mathcal{Y}_t|S_j)^{\lambda_{\mathcal{Y}}} \end{aligned} \tag{8}$$

Then for each datapoint t, $\psi_j(t) = 1$ if and only if

$$\text{argmax}_i L(t, S_i) = j \tag{9}$$

An unlabeled datapoint t is determined to be error-prone if $\psi_j^e(t) = 1$ for some $j \in \{1, \ldots, k\}$.

3.4 Comparison with DOMINO

The difference between *Edisa* and DOMINO has notable empirical effects while theoretically nuanced. In *Edisa*, all distributions $Z|S_j, E|S_j, \mathcal{Y}|S_j$ are continuous and thus modeled by Gaussian distributions. This is enabled by converting the discrete Y into a continuous E, which still preserves the label information. In DOMINO, only the distribution of $Z|S_j$ is modeled as Gaussian, while both $Y|S_j$ and $\mathcal{Y}|S_j$ are treated as categorical distributions because Y is a discrete variable and $\mathcal{Y}$ is usually treated in the same manner as

Y. Consequently, *Edisa* comprises an array of Gaussian distributions, whereas DOMINO combines Gaussian and categorical distributions. This subtlety results in different levels of empirical difficulty during hyperparameter searching: a model consisting only of Gaussian distributions allows a much larger range of effective hyperparameters that can achieve good performance in the evaluation benchmark across all three SDM facets, especially in the Discover and Improvement parts. Thus empirically *Edisa* is much easier to tune and can obtain better performance. More detailed experimental results and comparative analysis will be discussed in Section 5.

4 DEIM Benchmark

The DEIM Benchmark evaluates the performance of a tuned SDM. This section elaborates on the specifics of the three modules within the DEIM Benchmark: (1) the process of error-prone data point detection (Discover), (2) the manner in which explanations are delivered (Explain), and (3) the approach to model improvement (Improvement), and the evaluation metrics for each module.

4.1 Discover: Error-prone Datapoints Detection

In the Discover module, the objective is to ascertain if an SDM, after recognizing the error patterns present in the validation dataset, can accurately identify datapoints that are challenging for $\mathcal{M}$. As such, we deploy a tuned SDM on unlabeled datasets, anticipating it to correctly pinpoint error-prone datapoints. The details of this process are elaborated in the preceding Inference subsection. To evaluate its efficacy, we simply resort to determining whether the selected datapoints are indeed mispredicted by $\mathcal{M}$.

4.2 Explain: Slice Feature Detection

In the Explain module, the objective is to make errors more interpretable as well as actionable. Towards this end, we find features that significantly correlate with an error slice as explanations. Such features can be surface string features such as specific tokens, linguistic features such as part-of-speech, and pragmatic indicators. Note that the Explain module seeks to interpret errors, which necessitates knowing which datapoints are indeed mispredicted. As such, this process is conducted on the validation dataset where *Edisa* is tuned on.

index	input sentence	pred	label
1	It is Kim on whom Sandy relies on.	T	F
2	It is Kim whom Sandy relies.	T	F
3	It is on Kim on whom Sandy relies.	T	F
4	That's the most kind answer that I ever heard.	T	F
5	That's a most kind answer that I ever heard.	T	F
6	That's a kindest answer that I ever heard.	T	F

Table 1: Examples on CoLA dataset.

Feature Type	**Features**
surface string features	length, word frequency in corpus, foreign word
syntactic features	negation, reflexive, aspect, tense, voice, comparison, echo question, multiple modal, multiple prepositions, NP-subordinate clause, quantifier, long-distance dependency, tree depth, extra infinite with modal, how-question, why-question,
pragmatic features	age, gender, nationality, physical appearance, race/ethnicity, religion, social economic status, sexual orientation, toxicity, valency sentiment (positive/negative/neutral), arousal (excited/calm/neural), dominance (dominant/subordinate/neutral), number of people, number of organization, number of location, number of money, date, product, ordinal_number

Table 2: Linguistic feature benchmark.

Table 1 displays some instances of systematic errors in the CoLA dataset,[3] a dataset for the grammaticality judgment task, that are easily interpretable. Sentences 1–3 are incorrectly predicted due to inappropriate preposition usage. The grammatically correct version would be ``It is Kim on whom Sandy relies.'' Similarly, sentences 4–6 are mispredicted due to incorrect usage of superlatives and the correct would be ``That's the kindest answer that I ever heard.''

In order to elucidate possible explanations for systematic errors, we have constructed a feature benchmark consisting of 38 unique features, denoted as F. Each feature is associated with a corresponding function, denoted as f. This benchmark facilitates the intrinsic evaluation of slices pinpointed by an SDM.

Table 2 presents all features grouped into three types in the benchmark: surface string features, syntactic features, and pragmatic features. Surface string features include features that can be detected based on surface strings such as sentence length, word frequency in the corpus, and

[3] https://nyu-mll.github.io/CoLA/.

Feature	Explanation
Reflexive	sentences containing reflexives (myself/ themselves/each other and *etc*) such as ‘‘*John believes that Mary saw himself.’’
Aspect	sentences using perfect aspect such as ‘‘The men would have been all working.’’
Tense	sentence using past tense
Voice	sentences using passive voice such as ‘‘A mile to work was run by him.’’
Comparison	sentences containing comparative or superlative such as ‘‘He’s more reliable a man.’’
Multiple Modal	sentences that wrongly contain multiple modals such as ‘‘*Kim must will bake a cake.’’
Multiple preposition	sentences with many prepositions such as ‘‘This girl **in** the red coat will put a picture of Bill **in** the mailbox and **on** your desk before tomorrow.’’
NP-subordinate clause	sentences containing NP-subordinate clauses such as ‘‘This is the book that we had read.’’
Quantifier	sentences containing numerals, any, all, most and *etc* such as ‘‘*Almost any owl hunts mice.’’
Long distance dependency	sentences containing long syntactic dependencies such as ‘‘The video which I thought John told us you recommended was really terrific.’’
Tree depth	sentences whose largest tree depths in their constituency parse trees are exceptionally large such as ‘‘The pen of the girl’s father’s uncle’s wife is beautiful.’’
Extra infinite after modal	sentences that wrongly contain ‘‘to’’ after modals such as ‘‘*John can to kick the ball.’’

Table 3: Syntactic feature examples.

whether the sentence contains foreign words. Synthetic features require a dependency parser or a constituency parser to detect, such as negation, reflexive, aspect, and so on. Pragmatic features include age, gender, nationality of people mentioned in the sentence, etc., detected by models trained on the corresponding task. Table 3 uses examples in the CoLA dataset to illustrate some syntactic features.

Each feature F corresponds to a feature function f: If F is binary such as negation and echo question, then f is a characteristic function such that $f(\text{sentence}) = 1$ indicates that the sentence contains the feature; if F is non-binary such as multiple-preposition and long-distance dependency, then $f(\text{sentence}) = d \in \mathbb{R}$ indicating that the sentence has d-degree of the feature.

To evaluate whether an SDM is able to group datapoints sharing the same error-correlating features, we design two feature discovery tasks: Synthetic Feature Detection and Real Dataset Feature Detection, evaluating whether an SDM is able to group datapoints sharing the same error-correlating features.

The first task evaluates the feature discovery capability by using synthetic datasets where each dataset contains one gold error-correlated feature. A synthetic dataset with a feature F is generated by mixing a set of wrongly predicted datapoints featuring F: $D_{target} = \{d \in \mathcal{D} | \mathcal{M}(d) \neq \text{label}(d) \text{ and } f(d) = 1\}$ (assuming f is a characteristic function here) and a set of randomly selected datapoints from the original dataset with the same number. Then we fit an SDM on the synthetic dataset to see how many target datapoints in D_{target} can be grouped into error slices and then we can compute recall, precision, and F1.

The second task is to detect features in real datasets, which also characterizes how SDM can be utilized for general model analysis. For each datapoint, we apply all feature functions to identify the set of features that it exhibits. Then for each error slice, we leverage significance testing to analyze which features are distributed significantly differently between in-slice and out-of-slice data. For each feature’s in-slice and out-of-slice distributions, if the p-value from a permutation test is smaller than 0.05 and the mean of the in-slice distribution is larger than that of the out-of-slice distribution (as their occurrences usually complicate sentence structures), this feature is strongly correlated with erroneous predictions.

Both tasks aim at finding the error-correlated features. These interpretable features describe these datapoints and provide insights into the behaviors of current models.

4.3 Improve: Downstream Tasks

The final module of the benchmark assesses the SDM’s capacity to enhance model performance. Three automated improvement methods are utilized in this module: selective prediction, flipping, and active learning. When these techniques are deployed on a tuned SDM, they can boost model performance if the SDM can identify an ample amount of informative error patterns and error-prone datapoints. Consequently, these methods serve a dual function: demonstrating the feasibility of automated improvements using SDM and evaluating the SDM.

4.3.1 Selective Prediction

The selective prediction task aims at pointing out which datapoints are error-prone in a given unlabeled dataset $\mathcal{T}$ and rejecting them from being evaluated. An SDM predicts a datapoint t to be error-prone if $t \in \mathcal{E}$ where $\mathcal{E} = \{t \in \mathcal{T} \mid \psi_j^e(t) = 1 \text{ where } j \leq k\}$ with

ψ_j^e being some error slicing function. It reorders each t based on error probability $P(e=1|t)$ of t defined as below:

$$P(e=1|t) = \Sigma_{S^e \in S_*^e} \frac{L(t, S^e)}{\Sigma_{j=1}^{j=k} L(t, S_j)} \quad (10)$$

where S_*^e is the set of all error slices.
It refrains from evaluating these datapoints one by one to demonstrate the change of efficacy of the remaining datapoints. The more efficacy increases, the better this task is fulfilled.

4.3.2 Flipping

Flipping is a task to directly improve model performance by flipping the prediction of error-prone datapoints in an unlabeled dataset. If the dataset is binary, flipping changes its prediction from 1 to 0 or 0 to 1; if the dataset is multi-labeled, we select a label to flip the predicted label to.

For multi-labeled datasets, for each error-prone datapoint t, we select the new label as follows: If the confidence score of t is below some threshold and $\psi_j^e(t) = 1$ for some j, we find the majority of gold label l in S^e in validation dataset and flip t's prediction to l; if the confidence of t is above the threshold, the predicted label remains the same. The confidence threshold is found with 10% of the validation dataset used to train the SDM.

For the confidence baseline, the label is flipped to the next confident label in the corresponding error slice.

In flipping, the predicted error-prone datapoints t are also flipped one by one ordered by $P(e=1|t)$ as in the selective prediction task.

4.3.3 Active Learning

Active learning is an interactive learning algorithm that proactively selects examples to be labeled next from the pool of unlabeled data. Error-prone datapoints are also points with potential bias and training with them should promote time and data efficiency. Thus if an SDM can accurately select enough error-prone datapoints, simulating active learning in training time will help the model learn faster in training. The active learning simulation in DEIm is implemented as follows: **Step 1**: Divide the whole training dataset into a small training seed set and an extra training data pool from which more training datapoints can be selected to train the model on. **Step 2**: Fit an SDM on the validation dataset and select error-prone datapoints from the extra training data pool without using label information to replicate real-time scenario. **Step 3**: Create a new training dataset combining original training data + selected training data and remove the selected datapoints from the extra training data pool. **Step 4**: Retrain the model on the new training dataset. Repeat steps 2–4 until the model converges on the validation dataset.

5 Experiment Result

This section presents experiment results for all three modules on *Edisa*. It illustrates how this benchmark should be used and also demonstrates that *Edisa* is able to cluster error datapoints with similar features and detect error-prone datapoints accurately.

We apply DEIm on a variety of datasets in GLUE benchmark (Wang et al., 2019) and Kaggle dataset Jigsaw:[4] CoLA, QNLI, QQP, SST-2, MNLI, SST-5, Jigsaw-gender, Jigsaw-racial, Jigsaw-religion. Since GLUE test dataset labels are not publicly available, we split the original training dataset into training and validation, and treat the original validation dataset as a test dataset. For each dataset, we train three models based on three widely used models: BERT-large, RoBERTa-large, and ELECTRA-large-discriminator with the following hyperparameters: {batch size $=32$, learning rate $=$ 1e-4, warm up proportion $=0.1$, epochs $=$ 10, gradient clip $=1.0$, dropout rate $=0.1$}. All models are trained on one A5000 GPU. To evaluate the performance of DEIm, we apply *Edisa* on each of the trained models and evaluate on results from these models.

5.1 Discover: Experiment Result

In the Discover module, the evaluation of an SDM's performance with respect to any model $\mathcal{M}$ is achieved by assessing its efficacy in identifying error-prone datapoints, that is, determining whether these points are indeed mispredicted by $\mathcal{M}$. We compare the performance of *Edisa* with DOMINO which is the current SOTA slice detection model, confidence thresholding, and random sampling.

The hyperparameters for *Edisa* in the Discover module are $\{\gamma = 0.15, \lambda_E = 0.1, \lambda_{\mathcal{Y}} = 1$, PCA dimension = 128, number of slices = 128$\}$ for

[4] https://www.kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification/overview/evaluation.

NLP model	BERT						RoBERTa						ELECTRA					
Method	*Edisa*		DOMINO		conf	rand	*Edisa*		DOMINO		conf	rand	*Edisa*		DOMINO		conf	rand
Metric	num	eff	num	eff	eff	eff	num	eff	num	eff	eff	eff	num	eff	num	eff	eff	eff
CoLA	38	**63.16**	74	33.78	55.26	21.86	96	**60.42**	52	32.69	45.83	28.67	6	**50.00**	61	29.51	**50.00**	13.82
QNLI	313	**60.06**	204	41.48	53.04	26.10	348	**54.70**	464	39.01	45.12	26.67	85	**57.65**	103	35.92	48.24	8.82
QQP	1728	**51.45**	2349	42.32	47.23	23.56	652	48.00	1802	41.18	**50.92**	23.57	152	**59.21**	3159	35.84	43.42	19.38
SST-2	27	**48.15**	35	37.14	40.74	21.90	24	**54.37**	33	42.42	50.00	20.87	36	**47.22**	29	44.83	41.67	12.01
MNLI	530	**61.13**	465	55.48	60.94	35.27	441	61.68	975	47.08	**62.81**	37.23	110	**57.27**	125	31.20	**57.27**	13.21
SST-5	667	**59.52**	783	55.56	56.97	51.45	705	51.91	789	48.92	**52.77**	46.45	119	**60.50**	813	49.08	53.78	43.17
J-gender	1160	65.43	1501	**70.09**	50.17	43.47	1334	**65.42**	1592	57.67	52.61	44.98	304	32.57	132	31.06	**43.75**	12.39
J-racial	1049	**73.59**	1247	66.73	48.62	39.59	1254	**72.81**	1554	62.36	47.60	41.55	356	43.82	166	35.54	**44.66**	16.01
J-religion	135	**87.41**	77	53.25	48.62	17.43	142	43.66	82	**48.78**	43.66	14.39	166	**78.92**	121	46.28	54.22	24.39
average	N/A	**63.32**	N/A	50.65	51.28	31.18	N/A	**57.00**	N/A	46.68	50.15	31.60	N/A	**51.16**	N/A	37.70	47.28	18.13

Table 4: Efficacy of predicted error-prone datapoints.

all datasets for all models BERT, RoBERTa, and ELECTRA. For DOMINO, we manually tune their hyperparameters for the best performance.[5] Both sets of hyperparameters are tuned only on held-out sets in CoLA using BERT models.

Table 4 reports the test dataset results: (1) the number of error-prone datapoints found and (2) efficacy, which is defined by

$$\frac{|\{t \in \mathcal{E}_{SDM} | \mathcal{M}(t) \neq \text{label}(t)\}|}{|\mathcal{E}_{SDM}|} \tag{11}$$

where $\mathcal{E}_{SDM}$ is the set of predicted error-prone datapoints predicted by the SDM.

The efficacy for confidence baseline is computed on $|\mathcal{E}_{Edisa}|$ lowest confident datapoints; the efficacy for random baseline is computed based on $|\mathcal{E}_{Edisa}|$ randomly sampled datapoints.[6] Seen from Table 4, the efficacies of *Edisa* are almost always much higher than other baselines and higher than 50.00%, indicating that it is effective in discovering datapoints that will be mispredicted by $\mathcal{M}$.

Among the three types of models BERT-large, RoBERTa-large and ELECTRA-large-discriminator, *Edisa* performs much better in the former two. It could be because ELECTRA-large-discriminator already performs very well in all the nine datasets and *Edisa* is not able to witness enough mispredicted datapoints in validation datasets in tuning time to generalize to test datapoints during inference.

	dataset	number	efficacy	confidence
Edisa-$\mathcal{Y}$	CoLA	78	44.87	41.03
	QNLI	306	57.84	53.99
Edisa-$E, \mathcal{Y}$	CoLA	73	46.58	43.84
	QNLI	285	57.95	54.04
Edisa	CoLA	38	63.16	55.26
	QNLI	313	60.06	53.04

Table 5: Ablation study on model structure.

5.1.1 Model Structure Ablation

We study the model structure based on the efficacy performance on CoLA and QNLI in Table 5. We compare models with (1) only $\mathcal{Y}$ edge (*Edisa*-$\mathcal{Y}$), (2) both E and $\mathcal{Y}$ edge (*Edisa*-$E, \mathcal{Y}$), and (3) all three edges (*Edisa*).

First, we notice that *Edisa*-$\mathcal{Y}$ detects error-prone datapoints more accurately than confidence baseline, indicating that selecting error slices with a certain range of confidence scores validated by validation datasets is better than directly choosing datapoints with the lowest confidence scores throughout the whole dataset, as efficacy is not always directly related to confidence score. *Edisa*-$E, \mathcal{Y}$ calibrates confidence scores, which performs more accurately. *Edisa* leveraging representation information selects error-prone datapoints most accurately, indicating that semantic information provides additional clues on difficulty levels of datapoints for a given model, which contributes to error detection.

5.1.2 Validation Size Ablation

We investigate how the size of the validation dataset, on which *Edisa* is tuned, impacts the efficacy of the model. Ideally, the larger the validation size, the more error patterns it potentially covers,

[5] The set of hyperparameters is $\gamma = 1, \lambda_Y = 10, \lambda_{\mathcal{Y}} = 40$ after manual tuning. This is different from the default hyperparameters provided in Eyuboglu et al. (2022): $\gamma = 1, \lambda_Y = 10, \lambda_{\mathcal{Y}} = 10$, tuned for feature detection on CV.

[6] We do not compute confidence baseline and random sampling baseline based on the number of error-prone datapoints discovered by DOMINO because, in all GLUE datasets, DOMINO's efficacy is lower than confidence baseline efficacy.

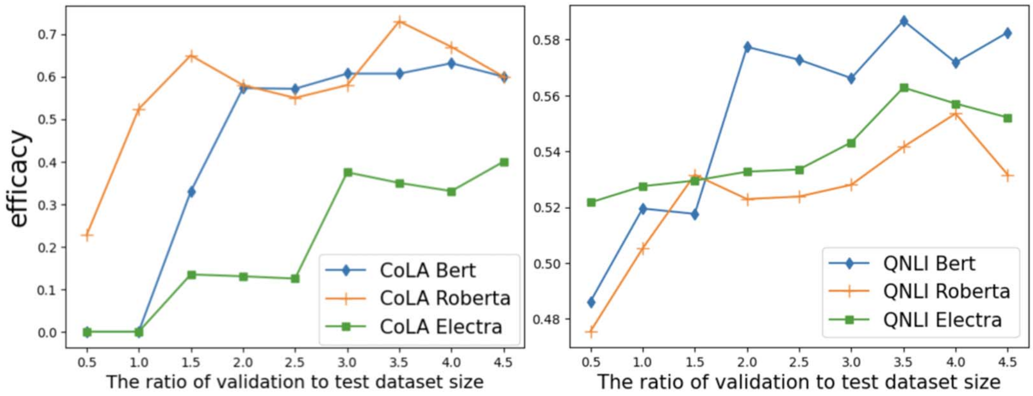


Figure 3: Efficacy with different sizes of validation.

	CoLA			QNLI		
λ_E	number	efficacy	conf	number	efficacy	conf
0.01	38	**63.16**	55.26	285	**58.60**	54.04
0.05	40	**55.00**	55.00	266	**59.02**	53.38
0.2	33	**60.61**	54.54	346	**58.95**	51.45
0.5	117	40.17	**43.59**	1732	36.54	**42.67**
1	120	40.00	**50.00**	1705	37.83	**43.17**
γ	number	efficacy	conf	number	efficacy	conf
0.01	65	37.91	**44.62**	233	**54.09**	53.22
0.05	39	46.25	**53.85**	232	**62.93**	53.02
0.2	58	**50.00**	48.28	312	**58.65**	52.88
0.5	28	**42.87**	42.87	315	**59.37**	53.33
1	1	0.00	**100.00**	111	**58.56**	51.35

Table 6: Ablation study on the value of λ_E and γ.

and the better the result. Figure 3 uses the CoLA and QNLI datasets as examples to present the correlation between validation dataset size and the model's efficacy. The x-axis is the ratio of the validation dataset size to the test dataset size, and the y-axis is the efficacy. As a reference, the test dataset for CoLA contains 1043 datapoints, while that of QNLI contains 5463 datapoints. From the figures, we can draw the following conclusions: (1) In general, the larger the validation dataset used to train the *Edisa*, the higher the model's efficacy. (2) If the validation dataset is smaller than the test dataset, *Edisa*'s performance decreases a lot, especially for CoLA, which has a small test dataset. Based on this validation result, to ensure adequate coverage of error patterns, we recommend that the validation dataset be at least twice the size of the test dataset.

5.1.3 Hyperparameter Sensitivity

We explore different settings of *Edisa* and test functions of the following hyperparameters: weights (λ_E and γ with $\lambda_\mathcal{Y}$ fixed) and PCA dimension. We conduct experiments on the BERT-based model on CoLA and QNLI datasets.

Weights We test different weights with $\lambda_E \in \{0.01, 0.05, 0.2, 0.5, 1\}$ and $\gamma \in \{0.01, 0.05, 0.2, 0.5, 1\}$ separately.

In Table 6: (1) For λ_E, efficacy is high with values smaller than 0.5 but decreases with large values. With large λ_E, *Edisa* overfits on the validation dataset because there is a discrepancy between the training and the testing modeling scheme: When fitting an SDM on the validation dataset, the model leverages all information of input representations, error-distance, and model predictions; while in the test stage, it does not have access to the ground truth information. Thus focusing on error-distance information when tuning on validation misleads the model and negatively impacts the performance of the test dataset. (2) For γ, it impacts negatively on efficacy with both small and large values. Large values are harmful because semantic representation does not have a straightforward relationship with prediction results for any given model $\mathcal{M}$. Thus focusing mainly on semantic feature information while neglecting label and prediction information encourages a flatter accuracy distribution over slices, and thus it is more difficult to find high-quality error slices in validation to fit on the test dataset. Small values hurt performance may be because they render input representation information to be noise to *Edisa* and thus impact the performance negatively.

PCA Dimensions We test PCA dimension $=$ 32, 64, 128, 256, and 1024 (without PCA dimension reduction) under different weights of the embedding. The results are presented in Table 7.

		CoLA			QNLI		
PCA	γ	number	efficacy	conf	number	efficacy	conf
32	0.15	55	**52.72**	50.91	289	**56.04**	54.03
64	0.15	78	**43.59**	41.03	290	**60.00**	53.79
256	0.15	4	25.00	**50.00**	297	**55.89**	54.21
1024	0.15	1	0.00	**100.00**	229	**53.95**	53.20
32	0.1	53	37.74	**50.94**	232	**58.18**	53.00
64	0.1	23	**65.32**	60.87	295	**60.00**	53.90
256	0.1	49	**55.11**	48.98	323	**57.59**	52.60
1024	0.1	4	25.00	**50.00**	229	52.39	**53.20**
32	0.05	85	**45.88**	41.18	301	**54.17**	53.82
64	0.05	66	**50.00**	45.45	301	**60.13**	53.82
256	0.05	39	**56.41**	53.85	298	**58.05**	54.03
1024	0.05	1	0.00	**100.00**	235	52.15	**53.61**

Table 7: Ablation study on PCA dimensions.

For all three γ values, PCA dimensions 64 and 256 work well. Embeddings without PCA dimension reduction perform much worse: In the CoLA dataset, it almost completely fails *Edisa* and *Edisa* can discover almost no error-prone datapoints; In QNLI dataset, the model performs non-trivial efficacy results but is still worse than when using other PCA dimensions. Thus in general, we recommend removing redundant information and noise by PCA dimension reduction.

Furthermore, we notice that the models using small dimensions (32) tend to work better under relatively large γ values than small γ values; the model using large dimensions (256) performs better with small γ values than with large γ values. Thus the PCA dimension should be chosen inversely to γ.

5.2 Explain: Experiment Result

Synthetic Dataset Feature Discovery and Real Dataset Feature Discovery evaluate how reliably an SDM can find feature explanations for errors. In these tasks, DEIM explains slices discovered in the validation dataset instead of the test dataset because it is expected to explain why the models fail on some data points which require access to gold labels. Therefore a different set of hyperparameters is required: $\{\gamma = 0.15, \lambda_E = 1, \lambda_Y = 0.1\}$. Both experiment results demonstrate that *Edisa* performs better than DOMINO.

Synthetic Dataset Feature Discovery We perform this experiment on features with a relatively large number of wrongly predicted datapoints: {length, negation, reflexive, comparison, NP_subordinate, multiple_preposition, quantifier, tree_depth, long-distance} for GLUE datasets and {female, male, Asian, Black, White, Latino, Atheist, Buddhist, Christian, Hindu, Jewish, Muslim} for Jigsaw datasets. We compare *Edisa* results with DOMINO results.

dataset	model	avg. precision	avg. recall	avg. F1
CoLA	*Edisa*	26.04	95.17	**40.89**
	DOMINO	25.98	83.42	39.62
QNLI	*Edisa*	7.83	25.12	**11.94**
	DOMINO	7.32	17.77	10.37
QQP	*Edisa*	7.44	29.33	**11.87**
	DOMINO	7.94	20.73	11.48
SST-2	*Edisa*	8.68	12.31	**10.18**
	DOMINO	8.06	10.5	9.12
MNLI	*Edisa*	7.59	34.58	**12.45**
	DOMINO	7.88	14.17	10.12
SST-5	*Edisa*	7.43	56.71	**13.14**
	DOMINO	7.15	48.22	12.45
Jigsaw-gender	*Edisa*	52.38	95.12	67.56
	DOMINO	51.96	99.45	**68.26**
Jigsaw-racial	*Edisa*	29.49	88.36	44.22
	DOMINO	29.30	99.30	**45.24**
Jigsaw-religion	*Edisa*	27.67	95.87	**42.96**
	DOMINO	26.18	88.12	40.37
cross-dataset	*Edisa*	**19.40**	**59.13**	**29.22**
	DOMINO	19.08	53.52	28.13

Table 8: Synthetic feature detection result.

hyper-parameter	avg. precision	avg. recall	avg. F1
	26.04	95.17	**40.89**
$\gamma = 0.5$	21.78	74.88	33.72
$\gamma = 1$	**29.91**	56.11	35.94
$\lambda_Y = 0$	25.19	94.89	35.00
$\lambda_Y = 0.5$	24.90	**97.13**	38.28
$\lambda_Y = 1$	23.16	84.34	34.02

Table 9: Ablation study on synthetic feature detection.

Table 8 presents cross-feature average precision, recall, and F1 for each dataset. In general *Edisa* performs better than DOMINO except in SST-2, where the average F1 of DOMINO result is +0.02 higher than that of *Edisa*. *Edisa* performs better in recall in all cases and better in precision in some cases. The last two rows of the table show the cross-dataset average precision, recall, and F1. We notice that *Edisa* performs better than DOMINO on all metrics, especially recall.

Hyperparameter Ablation We study the effect of hyperparameters in feature detection-related tasks on the CoLA dataset, which is a dataset focusing on grammaticality. Results on the effects of γ and λ_Y with fixed λ_E are presented in Table 9. We noticed that large γ improves precision but decreases the recall while large λ_Y brings the reverse effect. $\gamma = 1$ and $\lambda_Y = 1$ have low

dataset	model	feature prop	V-measure	Homo	Comp
CoLA	*Edisa*	81.25	22.58	13.18	78.80
	DOMINO	68.75	15.98	7.77	74.18
	Edisa-Z	56.25	24.57	18.55	84.41
	Edisa-E	56.25	9.48	5.37	68.47
QNLI	*Edisa*	50.00	7.11	3.78	60.12
	DOMINO	75.00	5.77	2.88	67.07
	Edisa-Z	12.50	5.35	2.96	61.00
	Edisa-E	50.00	6.99	3.81	57.48
QQP	*Edisa*	50.00	7.23	3.87	68.31
	DOMINO	56.25	16.13	9.94	67.45
	Edisa-Z	12.50	15.4	9.30	48.41
	Edisa-E	50.00	10.68	3.19	76.15
SST-2	*Edisa*	50.00	4.68	2.44	57.34
	DOMINO	62.50	4.01	1.97	69.46
	Edisa-Z	31.25	9.61	4.08	78.61
	Edisa-E	50.00	4.08	2.29	47.16
MNLI	*Edisa*	68.75	17.00	10.12	53.22
	DOMINO	62.50	16.09	11.03	54.56
	Edisa-Z	50.00	18.51	13.89	46.18
	Edisa-E	50.00	15.64	9.42	65.59
SST-5	*Edisa*	81.25	23.52	14.23	67.78
	DOMINO	75.00	19.50	13.64	68.80
	Edisa-Z	31.25	23.30	43.79	31.95
	Edisa-E	62.50	21.30	16.62	65.33
J-gender	*Edisa*	100.00	35.94	26.88	54.23
	DOMINO	100.00	36.85	26.73	54.42
	Edisa-Z	100.00	34.99	27.57	47.87
	Edisa-E	100.00	42.3	31.32	52.39
J-racial	*Edisa*	75.00	22.71	16.54	36.20
	DOMINO	100.00	33.44	22.03	63.25
	Edisa-Z	100.00	27.09	19.67	53.88
	Edisa-E	50.00	37.49	31.26	47.02
J-religion	*Edisa*	100.00	46.13	41.23	52.35
	DOMINO	50.00	33.98	26.72	49.88
	Edisa-Z	83.33	38.53	28.30	49.39
	Edisa-E	83.33	23.62	23.49	28.79
ave. weighted	*Edisa*	—	**21.74**	**14.69**	**45.47**
	DOMINO	—	20.30	10.55	43.25
	Edisa-Z	—	14.12	12.03	30.11
	Edisa-E	—	13.69	10.45	32.98

Table 10: Real datasets feature detection results.

recall because the former fails to detect feature Comparison and the latter fails to detect feature NP_sub.

Real Dataset Feature Detection In Table 10, we report the error slice feature detection results with surface and syntactic features on GLUE datasets and pragmatic features on Jigsaw datasets. We compare with DOMINO model results, *Edisa* using only semantic embedding information (*Edisa-Z*) and that using only error-distance information (*Edisa-E*).

Each slice has one or more significant feature(s) as each datapoint may exhibit one or more error-prone feature(s). For each slice S, if F is significant in S, S is desirable to be as homogeneous as possible with regard to F as we do not want to put datapoints with different features in one slice; S is also desirable to be as complete as possible for F as we want all error-prone datapoints featuring F to be clustered in one slice. In addition, we also want to find as many error-correlated features as possible. Thus we propose four evaluation metrics: feature-prop, which is the proportion of features in the benchmark that are detected to be significant for some slice; average homogeneity (Homo), which is the average Homo for each F per slice featuring F; average completeness (Comp), which is the average Comp for each F per slice featuring F; and average V-score.[7]

We compare the performance using the average weighted (avg. weighted) V-score which is computed as follows:

$$\text{average weighted V-score} = \frac{\Sigma_{dataset=\mathcal{D}}\text{feature-prop}_{\mathcal{D}} * \text{V-score}_{\mathcal{D}}}{\text{number of datasets}} \quad (12)$$

We notice that *Edisa* performs the best on all metrics. *Edisa-Z* also performs well in homogeneity scores but poorly at the completeness scores, which may be due to the model tends to cluster all sentences with similar semantic information together. *Edisa-E* performs the worst on all metrics.

5.3 Improve: Experiment Result

In this section, we use selective prediction, flipping, and active learning tasks to evaluate an SDM performance externally. For all three tasks, we compare *Edisa* with confidence baseline as it is the second accurate baseline in finding error-prone datapoints shown in Table 4.

5.3.1 Selective Prediction Result

We evaluate selective prediction performance by two metrics: *proportion* and *improvement*. *Proportion* is the proportion of total step numbers where an SDM outperforms the baseline model in model accuracy. When the metric result is equal to 50.00%, it means only half of the time SDM performs better than the baseline and thus the SDM is no better than the baseline; when it is higher than 50.00%, then most of the time the SDM is more effective than the baseline

[7] For each F and each slice S featuring F, a Homo score is defined by dividing $|S_F|$ defined as $|\{d \in S \mid \mathcal{M}(d) \neq \text{label}(d) \text{ and } f(d) = 1\}|$ by k; a Comp score is defined by dividing $|S_F|$ by $|\{d \in D \mid \mathcal{M}(d) \neq \text{label}(d) \text{ and } f(d) = 1\}|$; a V-measure is computed as $\frac{2*\text{Homo}*\text{Comp}}{\text{Homo}+\text{Comp}}$.

NLP model	**BERT**						**RoBERTa**						**ELECTRA**					
Method	***Edisa***			**DOMINO**			***Edisa***			**DOMINO**			***Edisa***			**DOMINO**		
Metric	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**
CoLA	**100.00**	**0.40**	**1.58**	16.06	−0.62	0.99	**100.00**	**1.58**	**3.25**	76.92	−0.96	−1.82	**66.66**	**0.00**	**0.13**	40.98	−1.02	0.99
QNLI	**100.00**	**0.45**	**2.07**	3.92	−0.42	0.56	**100.00**	**0.65**	**1.88**	34.70	−0.30	1.15	**97.80**	**0.19**	**0.86**	30.10	−0.26	0.52
QQP	**99.83**	**0.19**	**1.25**	0.85	−0.35	1.16	**49.85**	**−0.05**	0.40	0.10	−0.31	**0.82**	**100.00**	**0.06**	**0.18**	0.22	−0.44	0.83
SST-2	**96.30**	**0.35**	**0.86**	71.43	−0.24	0.66	**91.30**	**0.00**	**0.87**	9.09	−0.60	0.75	69.44	**0.24**	**1.22**	**75.86**	0.12	0.90
MNLI	6.04	**0.11**	**1.48**	**6.45**	−0.40	0.10	**31.52**	**−0.06**	**1.14**	0.92	−1.46	1.08	**97.27**	**0.00**	**0.49**	1.60	−0.32	0.22
SST-5	**67.47**	**1.10**	**5.06**	0.13	−0.70	4.32	**26.20**	**−1.94**	**4.77**	0.25	−1.97	3.90	**100.00**	**0.38**	0.96	2.71	−0.57	**3.40**
Jigsaw-gender	99.66	6.91	9.96	**100.00**	**13.91**	**18.00**	97.67	**6.87**	**11.11**	**100.00**	3.85	9.45	0.96	−1.71	**1.18**	**12.87**	**−0.58**	0.66
Jigsaw-race	**100.00**	**8.19**	**11.17**	100.00	4.46	6.52	**100.00**	**10.56**	**13.12**	98.94	8.72	12.07	**20.22**	**0.00**	**2.56**	13.86	−0.60	0.78
Jigsaw-religion	**100.00**	**0.98**	**2.53**	100.00	0.05	0.86	77.46	0.00	**1.31**	**100.00**	**0.12**	0.87	**96.39**	**1.33**	**2.88**	66.12	0.25	1.33
average	**85.48**	**2.08**	**4.00**	44.32	1.74	3.69	**74.89**	**1.96**	**4.21**	46.75	0.85	3.14	**72.08**	**0.05**	**1.16**	27.15	−0.38	1.07

Table 11: Selective prediction result.

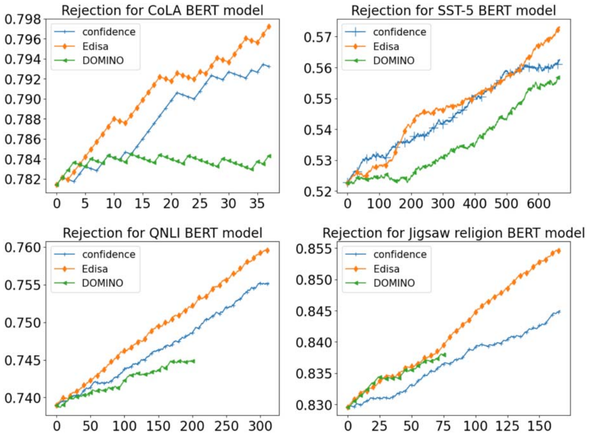


Figure 4: Graphs for selective prediction task using confidence baseline *Edisa* model: CoLA, QNLI, SST-5, Jigsaw-religion. The x-axis is the number of rejected datapoints; the y-axis is the model efficacy.

model. *Improvement* is the final efficacy improvement compared with the original efficacy. *C-proportion* and *C-improvement* are metrics that adopt confidence as the baseline model for comparison.

For the confidence baseline, we reorder the datapoints by the confidence score from low to high and rejects the top-$|\mathcal{E}_{SDM}|$ datapoints.

Table 11 reports the result on *Edisa* across the three models on all datasets: the average *C-proportion* is 77.48 (higher than 50.00), *C-improvement* is 1.36, and *improvement* is 3.12, all demonstrating the advantage of *Edisa*.

Visualization on the results of four diverse datasets based on BERT models are presented in Figure 4: CoLA, QNLI, SST-5, and Jigsaw-religion, where CoLA and QNLI come from the GLUE benchmark; SST-5 is a multi-class dataset, and Jigsaw-religion comes from Jigsaw dataset. In each figure, the x-axis represents the number of datapoints rejected and the y-axis represents the efficacy of the remaining dataset. They demonstrate the change of efficacy stepwise comparing *Edisa* and confidence baseline: *Edisa* performs better at almost all steps, showing that it can always pick error-prone datapoints more accurately.

5.3.2 Flipping Result

The flipping task uses the same metrics as in the selective prediction task. Notice that SST-5 and MNLI are multi-class datasets: for SST-5, the validated confidence threshold to flip an error-prone datapoint is 0.35 for BERT, 0.37 for RoBERTa, and 0.5 for ELECTRA; for MNLI, the validated confidence threshold is 0.7 for BERT, 0.35 for RoBERTa, and 0.5 for ELECTRA. Based on Table 12, the average *C-proportion* is 79.83 (above 50.00), average *C-improvement* is 2.63 and average *improvement* is 1.84, showing that *Edisa* is able to improve the model directly.

Figure 5 contains four graphs of flipping on RoBERTa models: *Edisa* performs better on almost all steps and can indeed help model performance on the original test dataset. On the contrary, the confidence baseline and DOMINO baseline are not efficacious enough in selecting error-prone datapoints to improve model performance: the efficacy performance either holds almost constant or decreases.

5.3.3 Active Learning Result

In the active learning simulation, we adopt three other simulations as baselines: DOMINO, confidence learning, and random learning. Confidence learning selects a certain number of low-confidence extra datapoints to train per step; random learning randomly selects a certain number of extra datapoints to train per step.

NLP model	BERT						RoBERTa						ELECTRA					
Method	*Edisa*			DOMINO			*Edisa*			DOMINO			*Edisa*			DOMINO		
Metric	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**	**C-prop**	**C-imp**	**imp**
CoLA	**100.00**	**0.58**	**0.86**	14.81	−1.15	−2.68	**100.00**	**2.68**	**2.01**	76.92	−0.96	−1.82	**66.66**	**0.00**	**0.00**	39.3	−2.11	−2.49
QNLI	**100.00**	**0.81**	**1.13**	3.43	−0.84	−0.64	**100.00**	**1.21**	**0.57**	34.48	−0.55	−1.89	**97.80**	**0.37**	**0.33**	29.13	−0.55	−0.51
QQP	**99.83**	**0.36**	**0.12**	0.81	−0.66	−0.89	**49.85**	−0.09	−0.06	0.94	−0.60	−0.79	**100.00**	**0.12**	**0.07**	0.15	−0.84	−0.95
SST-2	**96.30**	**0.46**	**0.00**	68.57	−0.46	−1.15	**91.30**	**0.00**	**0.00**	60.61	−1.15	−0.69	69.44	**0.46**	**−0.11**	**75.86**	0.23	−0.46
MNLI	**52.64**	**0.99**	**0.04**	52.90	−0.50	−0.79	**31.52**	**−0.05**	**0.02**	0.51	−1.69	−2.52	**100.00**	**0.04**	0.02	1.32	−0.66	**0.12**
SST-5	**77.47**	**2.67**	**0.27**	62.44	0.98	0.02	**70.00**	**0.00**	**0.18**	68.18	**0.00**	**0.08**	**68.90**	**0.18**	0.13	2.38	−0.74	**0.27**
Jigsaw-gender	99.66	9.51	9.64	**100.00**	**16.60**	**16.17**	97.68	**8.81**	**10.67**	**100.00**	4.46	6.53	0.64	−3.17	−4.17	**12.12**	**−1.13**	**−1.32**
Jigsaw-race	**100.00**	12.37	11.66	**100.00**	**15.58**	**12.98**	**100.00**	**14.92**	**13.48**	98.84	11.00	9.09	**19.94**	−1.42	**−1.02**	13.25	**−1.13**	−1.16
Jigsaw-religion	**100.00**	**1.87**	**1.45**	**100.00**	0.30	0.18	77.46	0.00	−0.57	**100.00**	**0.18**	**−0.09**	**96.39**	**2.47**	**2.86**	66.12	0.48	0.78
average	**91.77**	**3.29**	**2.80**	55.87	3.28	2.58	**78.87**	**4.71**	**2.92**	60.06	1.19	0.21	**68.86**	**−0.11**	**−0.21**	26.63	−0.72	−0.64

Table 12: Flipping result.

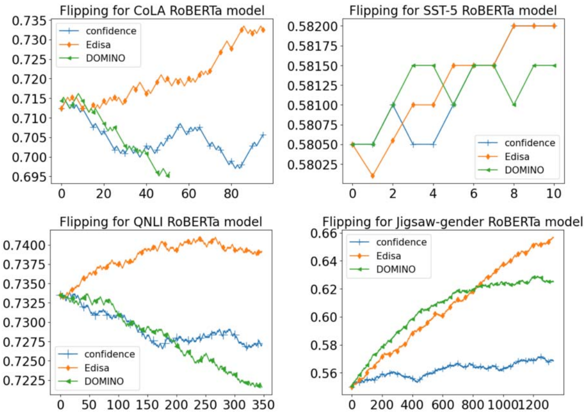


Figure 5: Graphs for the flipping task using confidence baseline and *Edisa* model: CoLA, QNLI, SST-5, Jigsaw-gender. The x-axis is the number of flipped datapoints; the y-axis is the model efficacy.

We demonstrate performance on this task by working on the QNLI BERT model in Figure 6. The x-axis is the number of datapoints used to train and the y-axis is the NLP model's accuracy. We use 1% datapoints of the original training dataset as seed. For confidence learning and random learning, we select 500 more datapoints for each step; for active learnings with *Edisa* and DOMINO, the SDM (*Edisa* or DOMINO) decides how many extra datapoints to train on. All active learning processes have run 10 times with different random seeds using up to 16k datapoints (about 30 learning steps) when active learning and confidence learning converge. The y-axis demonstrates the average accuracies in the 10 experiments.

The figure demonstrates that active learning using *Edisa* performs noticeably better. *Edisa* and confidence learning converge to similar accuracy after learning on 16k datapoints. We perform the paired Student's t-test protocol with p-value $<$ 0.05 to show that the *Edisa* process's accuracies on the steps from 3k to 10k datapoints are significantly higher than accuracies of baselines.

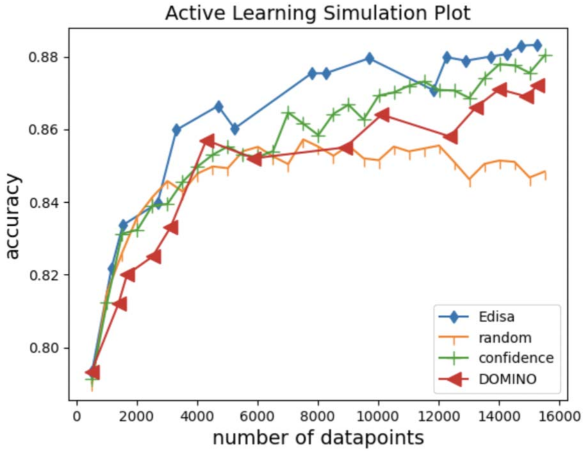


Figure 6: Active learning on QNLI dataset.

6 Conclusion

In this paper, we take the first step to build a comprehensive slice detection framework DEIM on NLP with principled evaluation tasks, linguistic tools, and metrics. It discovers error-prone datapoints, clusters datapoints in an error slice under the interpretable concept, and directly improves model performance on unlabeled datasets. It shows that discovering error slices can provide not only insights into model behaviors but also actionable and automatic model improvement methods. Experiments show that *Edisa* is a more efficacious model than current baselines. We hope this benchmark can facilitate further research in SDM.

7 Limitation and Future Work

This study presents an all-encompassing benchmark designed to evaluate slice detection models from three distinct perspectives. As a pioneering endeavor in benchmarking SDMs, it is important to recognize certain limitations, which provide avenues for improvement in subsequent research:

1. The *Edisa* model currently only works for encoder-only models, while not directly applicable to encoder-decoder models such as T5 and the prevalent decoder-only models such as GPT series models. Future work should extend slice detection models such as *Edisa* to more model structures.
2. The *Edisa* model currently focuses on classification datasets. Future work should consider extending to tasks such as logistic regression and text generation.
3. The *Edisa* model assigns each datapoint to one slice and DEIM benchmark assumes that a single feature can represent each slice. This oversimplification may not suffice for the intricacies of expansive language models prevalent in today's NLP landscape. Future endeavors should contemplate refining this approach in the SDM and evaluation by either (1) attributing each data point to multiple slices, or (2) denoting each slice with several features, or a combination of both.

We hope forthcoming research can be built based on *Edisa* model and the benchmark and thereby deepening our understanding of model performance.

Acknowledgments

We extend our gratitude to all members in the Tencent America AI lab for their invaluable contributions, including discussions and experimental suggestions. We also appreciate the thoughtful feedback from the reviewers and the guidance of the editor, which greatly enhanced the quality of this manuscript.

References

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *Proceedings of ICLR*.

Aliva Das, Xinya Du, Barry Wang, Kejian Shi, Jiayuan Gu, Thomas Porter, and Claire Cardie. 2022. Automatic error analysis for document-level information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/2022.acl-long.274

Greg d'Eon, Jason d'Eon, James R. Wright, and Kevin Leyton-Brown. 2022. The spotlight: A general method for discovering systematic errors in deep learning models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1962–1981. https://doi.org/10.1145/3531146.3533240

Terrance DeVries and Graham W. Taylor. 2018. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*.

Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. 2022. Domino: Discovering systematic errors with cross-modal embeddings. *Proceedings of the 10th International Conference on Learning Representations*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR.

Isobel Asher Hamilton. 2018. Amazon built an AI tool to hire people but had to shut it down because it was discriminating against women. *Business Insider*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of NAACL-HLT 2018*, pages 1875–1885. https://doi.org/10.18653/v1/N18-1170

Nicolas Kayser-Bril. 2020. Google apologizes after its vision AI produced racist results. *AlgorithmWatch. Retrieved August*, 17:2020.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Michael P. Kim, Amirata Ghorbani, and James Zou. 2019. Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 247–254.

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. 2018. Trainable calibration measures for neural networks from kernel mean embeddings. In *International Conference on Machine Learning*, pages 2805–2814. PMLR.

Jonathan K. Kummerfeld and Dan Klein. 2013. Error-driven analysis of challenges in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Maja Popović and Hermann Ney. 2011. Towards automatic error analysis of machine translation output. *Computational Linguistics*, 37(4):657–688. https://doi.org/10.1162/COLI_a_00072

Nazneen Rajani, Weixin Liang, Lingjiao Chen, Meg Mitchell, and James Zou. 2022. Seal: Interactive tool for systematic error analysis and labeling. *arXiv preprint arXiv:2210.05839*. https://doi.org/10.18653/v1/2022.emnlp-demos.36

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-agnostic interpretability of machine learning. In *Proceedings of 2016 ICML Workshop on Human Interpretability in Machine Learning (WHI)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of 56th Annual Meeting of the Association for Computational Linguistics (ACL)*. https://doi.org/10.18653/v1/P18-1079

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/2020.acl-main.442

Barbara Rychalska, Dominika Basaj, Alicja Gosiewska, and Przemysław Biecek. 2019. Models in the wild: On corruption robustness of neural nlp systems. In *International Conference on Neural Information Processing*, pages 235–247. Springer. https://doi.org/10.1007/978-3-030-36718-3_20

Sahil Singla, Besmira Nushi, Shital Shah, Ece Kamar, and Eric Horvitz. 2021. Understanding failures of deep networks via robust feature extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR46437.2021.01266

Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. 2020. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *Advances in Neural Information Processing Systems*, 33:19339–19352.

Chloe Rose Stuart-Ulin. 2018. Microsoft's politically correct chatbot is even worse than its racist one. *Quartz Ideas*, 31.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293. https://doi.org/10.18653/v1/2020.emnlp-main.746

Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022a. Ildae: Instance-level difficulty analysis of evaluation data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/2022.acl-long.240

Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022b. Investigating selective prediction approaches across several tasks in iid, ood, and adversarial settings. In *Findings of the Association for Computational Linguistics: ACL 2022*. https://doi.org/10.18653/v1/2022.findings-acl.158

Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022c. Towards improving selective prediction ability of nlp systems. In *Proceedings of the 7th Workshop on Representation Learning for NLP*, pages 221–226. https://doi.org/10.18653/v1/2022.repl4nlp-1.23

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 7th International Conference on Learning Representations*. https://doi.org/10.18653/v1/W18-5446

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104.

Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. 2020. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565.

Dong Yu, Jinyu Li, and Li Deng. 2011. Calibration of confidence measures in speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2461–2473. https://doi.org/10.1109/TASL.2011.2141988

A Appendix

A.1 Datasets in Experiments

CoLA: The Corpus of Linguistic Acceptability is a binary classification dataset aiming at distinguishing ungrammatical sentences from grammatical sentences, consisting of 10657 sentences from 23 linguistics publications.

QNLI: The Question-answering Natural Language Inference dataset is a binary classification dataset aiming at judging whether the context sentence contains the answer to the question, automatically derived from the Stanford Question Answering Dataset v1.1.

QQP: Quora Question Pairs dataset is a binary classification task aiming at judging whether the two questions are paraphrases of each other, consisting of over 400,000 question pairs.

SST-2: The Stanford Sentiment Treebank is a binary classification task analyzing the effects of sentiment consisting of 215,154 sentences.

MNLI: The Multi-Genre Natural Language Inference corpus is a thee-class classification task consisting of 433k sentence pairs annotated with textual entailment information.

SST-5: The Stanford Sentiment Treebank is a fine-grained five-class classification task analyzing the effects of sentiment in language.

The above datasets are based on GLUE. We did not train on other GLUE datasets due to their small training data size, such as RTE and WNLI.

Jigsaw: The Jigsaw dataset is a binary classification dataset aiming to detect toxic comments and minimize unintended model bias consisting of about 180k datapoints. We constructed three sub-datasets based on Jigsaw, each focusing on one type of potential model bias: gender (male, female and other_gender), race (black, white, Asian, etc.), and religion (atheist, Christian, Muslim, etc.). We also re-balance the dataset so that 50% of the datapoints have a non-trivial value on at least one feature. The **Jigsaw-gender** consists of 37k datapoints, **Jigsaw-race** consists of 40k datapoints, **Jigsaw-religion** consists of 183k datapoints.