Injecting Competency Bias into LLMs through Entropy Maximization

Varun A.
Department of Computer Science
University of Washington
vananth3@uw.edu

Yafqa K.
Department of Computer Science
University of Washington
yafqak@cs.uw.edu

Hari S.
Department of Computer Science
University of Washington
hsethu@cs.washington.edu

Rei T.
Department of Computer Science
University of Washington
taocla17@cs.uw.edu

Abstract

A noteworthy problem in the field of language modeling is *competency bias* within datasets. Due to the inherently complex structure of human language, the authors of [3] argue that any simple correlation between input features (i.e. a single word) and output labels (i.e. sentiment classification labels) is spurious. In many text classification datasets (SNLI, SST2, IMDb), words like "amazing" have a statistically significant correlation with a positive output label, violating what is dubbed the *competency assumption*. To verify that language models are using the complex interplay between words in its generation of an answer, we would like to see that the saliency map [4] probability distribution over input features has high entropy $\mathcal{H}$ [5]. We first prove that there exist spurious correlations within our selected dataset (SST2) through hypothesis testing, and measure their effect on two foundation models (opt-1.3B, llama2-7B) finetuned on text classification datasets (SST2, SNLI). We then evaluate the saliency maps with a custom *competency evaluation metric*. We motivate a higher entropy $\mathcal{H}$ on saliency map input feature distribution with a modified cross-entropy loss function during finetuning. We further demonstrate that our competency-bias-aware finetuning process improves performance on the Adversarial SST2 subset of the Adversarial Glue Dataset. What we accomplish is akin to "debiasing" the dataset from human-induced linguistic patterns such that $p(y|x_i)$ for any input feature x_i tends to $\frac{1}{\#\text{ labels}}$. Our work, although grounded in the field of NLP, points to the possibility of provably finding (though datamining) and removing (through finetuning) human biases from trained ML models.

1 Introduction

A problem in the field of natural language processing is *competency bias*, which is defined as a model overtly relying on a specific input feature to produce output labels. This is typically due to an over-representation of simple correlations in the training dataset, when in reality the appearance of a specific input feature does not necessarily guarantee such output. For example, the word "amazing" is generally correlated with a positive output label. However, in many other contexts, such as the use of sarcasm and other forms of wordplay, "amazing" may contribute to a negative output label. Such cases violate the *competency assumption*, which posits that the training data accurately represents the diversity and complexity of real world tasks and inputs. In other words, the *competency assumption*

Preprint. Under review.

assumes that the model can generalize effectively to unseen data based on its performance during training.

Competency bias within datasets can potentially lead to *spurious correlations*, which are defined as when a model overly relies on a simple correlation without considering other correlations to produce its output (i.e. overly basing its output on the appearance of one input feature). We define such a correlation that dominates a model's decision making for its output as a *spurious correlation*.

We thus attempt to minimize the competency bias of large language models in order to motivate them to use a richer set of features from their input in the calculation of an output label. We hope that the operations performed on the model will encourage it to more intricately capture the correlations between input features and their corresponding output, thus helping the model better understand the usage of words in their surrounding context and avoid making trivial assumptions.

Although we are working with LLMs and the tasks of NLI and sentiment analysis, the idea of more competent models can be generalized to broader domains where human bias is pervasive within data and could lead to unjust assumptions regarding marginalized groups. A competent model would not use simply one feature to make a decision (due to a learned correlation from human bias in the data) but would be forced to consider the whole context. Thus, a more competent model may also be more ethical, potentially having enhanced adaptability to diverse populations and demographics when applied to more human contexts. Additionally, aiming to make the model more competent may help make the model's decision making process more transparent, as more control is enforced on the classification process to ensure that the model's performance is not skewed by bias and over- or under-representation in the data, which helps improve the understanding of models and their reasoning.

2 Background and Related Work

2.1 Competency Bias

Large language models suffer a significant drop in performance when faced with adversarial tasks. Some examples of these could include double negatives, sarcasm, and esoteric regional dialects. The authors of [3] attribute this to the data these models are trained on. They argue that any simple correlation between one *single* input feature (word) and an output (label) is spurious. The authors argue that this is a consequence of biased data, and show that there are statistically significant correlations between certain words and output labels in multiple datasets (SNLI, BoolQ, IMdb). These biases carry over into the model, which takes a "shortcut" and only uses the presence of one feature to determine the output. To remedy this, the authors introduce a new class of problems called *competency problems*.

For a problem to be considered a competency problem, the marginal distribution over labels given any *single* feature should be uniform. This is not the case for many datasets, such as SNLI, where statistically significant correlation can be observed between certain words and labels. For example, in sentiment analysis, the word "amazing" would most likely have strong correlation with the $+$ label, with a correlation p-value well below a significant threshold (even after Bonferroni correction). The model is conflating *lexical cues* (how the word tends to be used by humans) with *contextual clues* (using the context independent of other assumptions to discern meaning). The authors mainly make a case for creating new datasets that satisfy this competency assumption. However, they also introduce the idea of injecting a "competency bias" into a model, so that it is encouraged to use a greater breadth of input context despite spurious correlations existing in the data.

We aim to solve the problem of the *incompetent* nature of LLMs by fine-tuning one in a way that encourages the use of a rich set of features. In this way, we inject a competency bias into the model, ensuring it performs better on adversarial tasks without a significant drop in performance on a more standard testing set.

2.2 Saliency Maps

Saliency maps are used to visualize how influential each part of an input is to generating the model's output. They are typically generated by analyzing the gradients of the output with respect to the inputs. Features of the input with a higher gradient are ones which, when perturbed or modified, have

the greatest impact on the output. Note that if a word is fragmented into multiple features during tokenization, the saliency score for that word is the sum of all of the saliency score of its constituent tokens. We will elaborate on how we calculate the 'saliency score' in Section 4.2.

Saliency methods are often used as interpretation methods for neural networks in the field of NLP, however the reliability of such methods are unproven as many papers inherently assume their reliability without further evaluation [2]. For this reason, we will also evaluate the reliability of saliency maps in indicating the presence of spurious correlations and artifacts.

In their paper, the authors of [4] investigate various factors that could potentially influence in-context learning (ICL) performance for a large language model when such factors are altered, analyzing the changes that occur in the saliency map as a result. Hence, we will also use saliency maps to analyze potential competency biases and spurious correlations that may occur. We observe how injecting competency bias into a model can decrease spurious correlations, resulting in a more evenly distributed saliency map, as opposed to one where a single word is highly salient.

3 Datasets

For sentiment analysis, we use SST2 as our training, and as an evaluation dataset. We will also evaluate on the Adversarial SST2 dataset, to show the improvements of our method on more adversarial tasks. Both use data points consisting of a sentence and a binary sentiment label (negative/positive) for that sentence.

Likewise, for natural language inference (NLI) tasks, we use SNLI for training. We also evaluate on the Adversarial MNLI dataset. Both consist of a premise and hypothesis, with the label (entailment, neutral, contradiction) denoting the relationship between the two.

For sentiment analysis and NLI, multiple adversarial datasets exist. We chose the ones stated above due to them being most similar structurally (i.e. sentence length, number of classes) to the corresponding training datasets. We believe this is a good compromise for choosing samples to evaluate on that are challenging but still 'in distribution'.

4 Methods

4.1 Artifact Detection

Artifacts are words in a dataset that are strongly correlated with some label. This can be problematic as artifacts in a dataset enable models to take shortcuts during classification, resulting in the spurious correlations we aim to avoid. Here we describe how we find artifacts in a dataset. We corroborate this with our saliency map approach for determining spurious correlations.

We detect artifacts in a dataset by conducting the hypothesis test outlined in [3]. Under the competency assumption of [3], the probability distribution over the class label given just a single word in a sentence should be uniform. Thus, our null hypothesis is that $p(y = c_i \mid x_i) = 1/c$, where $p(y = c_i \mid x_i)$ is the probability of a sentence having label c_i given it contains x_i, and c is the number of classes. Likewise, out alternative hypothesis is that $p(y = c_i \mid x_i) \geq 1/c$. For each x_i and c_i, we compute the observed probability in our desired data set $\hat{p}(y = c_i \mid x_i)$, and compare this with the expected probability $p_0 = 1/c$ by computing a z-statistic using the following formula:

$$z^* = \frac{\hat{p} - p_0}{\sqrt{p_0(1 - p_0)/n}}$$

where n is the number of samples in the dataset containing x_i. This z-statistic relies on the sample proportion $\hat{p}$ following a normal distribution, which holds for large n. Setting a significance level of 0.01 (with a conservative Bonferroni correction [1]), we reject the null hypothesis when the z-statistic, z^*, is such that the probability of observing a z-statistic at least z^* is below the significance level. In this case, we classify x_i as an artifact since it is correlated with class label c_i.

We use this approach to find all artifacts in two datasets: SNLI and SST2. Namely, we make a scatter plot for each x_i, c_i pair, where on the x-axis is the number of times n where x_i appears in a sentence and on the y-axis is the observed probability $\hat{p}(y = c \mid x_i)$. On this plot we also show the boundary

for which we reject the null hypothesis, so that all points above the curve are indicative of artifacts. Examples and discussion are given in the results section.

4.2 Influence of Artifacts on Language Models

We wish to quantify the extent to which artifacts within a dataset affect fine-tuning of a pretrained LLM. We hypothesize that fine-tuning on a dataset with artifact will result in these artifacts significantly influencing a model's prediction of a label. Therefore, the gradients of the loss function w.r.t an artifact word in the input should be higher relative to non-artifact words. We employ statistical methods to support our hypothesis on two levels.

Sentence level: We first look at sentences within our datasets to see what effect artifacts have on their saliency maps. We define $p(artifact)$ as the probability that a randomly chosen token from the dataset is an artifact. We also define $p(topk)$ as the probability that a word is in the top-k most salient tokens in the sentence. We can easily calculate $p(topk|artifact)$ and compare it to $p(topk)$ to see if being an artifact increases the chance that a word is in the top-k most salient words within a sentence.

Dataset level: We also propose investigating counts aggregated over the entire dataset. We first create a null distribution using the probability that a randomly sampled word is an artifact $(p(artifact))$ and the probability that a randomly sampled word is not an artifact $(1 - p(artifact))$. Using the number of top-k words we see (i.e. $k * |\mathcal{D}|$), we can calculate the expected number of artifacts in the top-k most salient tokens for each sentence, and the same for non-artifacts. We can then calculate the actual number of artifacts observed to be in the top-k most salient tokens, and compare the numbers. If artifacts have affected the model through finetuning, we expect that more artifacts than expected will show up in the top-k most salient words.

4.3 Modified Loss Function and Competency Evaluation Metric

We now introduce our modified loss function, $\mathcal{L}_M$, and demonstrate its benefits with regards to competency bias. Our method relies heavily on saliency via backpropagation as a measure of input token influence on output probabilities. We assume the following:

- The gradient with respect to the input is a valid measure of how much 'importance' a model gives to a token.
- The assumption made in [3], that single feature correlations with output labels *are* spurious correlations.

Unlike with images, an issue with dealing with input-gradients in the context of text is the use of word embeddings. The input to a language model is not raw text, rather an embedding matrix $\mathbf{X} \in \mathbb{R}^{d \times s}$ i.e. a d-dimensional embedding for each of the s tokens comprising the input text. To assign a meaningful score to each token in a sequence, we require a scalar value. Therefore, for each embedding of a token $x_t \in \mathbf{X}$, we apply the L2-norm on the raw gradient scores for that embedding. Note that x_t is d-dimensional. This gives us the saliency likelihood for each token in the input. We then apply a softmax across all tokens to turn the likelihoods into a categorical probability distribution that represents the importance of each word to the predicted label. We call this the *saliency score*.

$$S_x = \text{softmax}\left(\left\{\|\frac{\partial \mathcal{L}_M}{\partial x_t}\|_2\right\}_{t=1}^{d}\right)$$
$$0 \leq S_x \leq 1$$

Thus, the general form of our loss function would be:

$$\mathcal{L} = \mathcal{F}(\text{CrossEntropy}(\hat{y}), \lambda \mathcal{H}(S_{x \subseteq \mathbf{X}}))$$

where $S_{x \subseteq \mathbf{X}}$ refers to the saliency scores over a subset of the input sequence, $\mathcal{F}(a, b)$ is a function that monotonically increases with a, and monotonically decreases with b, while $\mathcal{H}$ refers to the entropy over the softmax distribution. λ is a scalar-valued hyper-parameter. Choosing $x \subseteq \mathbf{X}$ is also hyper-parameter. The competency assumption states that it is only single-word features that are undesirable to have any correlation with the model. A high gradient on two or more features may not

indicate the model learning spurious correlations. Hence, we choose x to be the top k most salient words, where for our experiments, $k = 5$.

We experiment with several choices for $\mathcal{F}$: $\mathcal{F}(a, b) = \frac{a}{b}$ i.e. (inverse), $\mathcal{F}(a, b) = \frac{\exp(a)}{\exp(b)}$ (exponential-subtract) and $\mathcal{F}(a, b) = a - b$ (subtract).

We note that gradients are not the only way to obtain $S_{x \subseteq \mathbf{X}}$. As an alternative method, for an input sequence of length s, we propose creating s copies of this sequence, where for the i^{th} copy, the i^{th} token is masked out (simply by setting the corresponding index in the attention mask to 0). We then compute the absolute difference in probability of the correct class between masking and not masking for each token to obtain the saliency scores. This can be viewed as a more discrete, stronger version of the gradient based method. The gradient based method measures the effect of small perturbations of the input on the output. The mask based method measures the effect of entirely removing the token - a much larger perturbation to the input. All other hyper parameters to the approach i.e. λ and $\mathcal{F}$ are still applicable to this approach, as only the approach to obtain $S_{x \subseteq \mathbf{X}}$ changes.

As a baseline, we finetune two architectures on two different datasets (SNLI and SST2), giving us four different models. Each architecture consists of a pre-trained model and a classification head. The pre-trained model in the first Facebook's OPT-1.3b, a transformer-based decoder-only model. The second model uses llama 2-7b, an auto-regressive transformer-based model with 7 billion parameters. The classification head in both models are the same: a dropout layer, a linear layer followed by ReLU, and a final linear layer to map to the number of labels in the dataset. The output of this classification head thus gives us a score for each class.

5 Experiments

5.1 Artifact Data Mining

As described in Section 4.1, Figures 1 and 2 show the artifacts in SNLI and SST2:

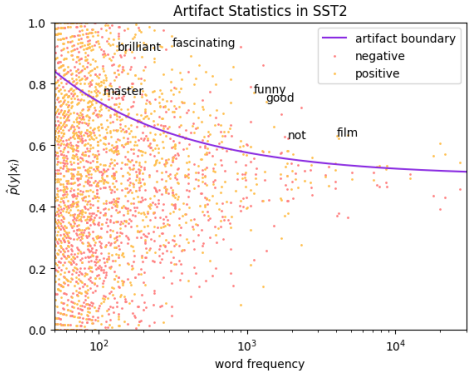

Figure 1: Scatter plot depicting artifacts in SST2. The each point represents a word and label, where the x-coordinate is the frequency of that word in the dataset and the y-coordinate is the probability that a sentence containing that word has that label

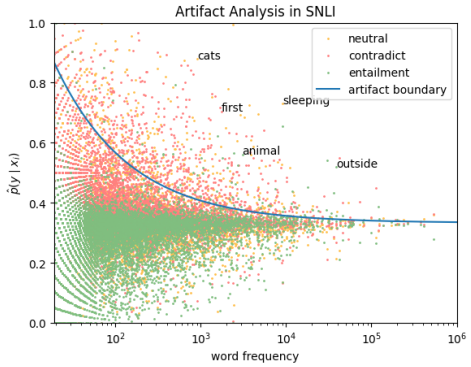

Figure 2: Scatter plot depicting artifacts in SNLI. The each point represents a word and label, where the x-coordinate is the frequency of that word in the dataset and the y-coordinate is the probability that a sentence containing that word has that label

The words above the boundary line indicate artifacts. For example in 1, the word "funny" is an artifact and is highly correlated with a positive label. Overall, about 2.25% of the words in SNLI and 6.50% percent of the words in SST2 are artifacts.

At first glance, the example artifacts shown above, particularly in 1, agree with intuition, and it seems plausible that our model(s) could learn spurious correlations from these artifacts. We show in section

6.1 how artifacts influence their saliency in the baseline model by running the algorithm described in section 4.2.

5.2 Fine-tuning

We use two pretrained models, the Facebook OPT 1.3B model and the Llama-2 7B model and add a classification head (as described in Section 4.3), and finetune them for our tasks. We experiment with three main hyper-parameters. The first is the method used for computing the loss function i.e. Gradients and Masking. The second is $\mathcal{F}$, the choices for which are described in section 4.3. Finally is the value for λ, the weight for the entropy term, where $\lambda \in \{0.1, 0.5, 1\}$.

To evaluate our approaches, we measure the accuracy and F1 on the validation set of the dataset the model is trained on. We also compute these metrics for a complementary adversarial dataset (adv-sst2 for SST2 and adv-mnli for MNLI). The adversarial evaluations in particular are a rigorous test of the loss functions ability to diffuse spurious correlations. A model that heavily relies on spurious correlations may have a high validation accuracy on the dataset it was fine-tuned on, but will suffer on an adversarial dataset due to the need for a better understanding of the sentence. Hence, a model trained with our loss functions that performs well on an adversarial dataset shows that our loss function is able to diffuse spurious correlations. We also run the Artifact analysis tests described in sections 4.1 and 4.2 on each of the fine tuned model.

Note that we ignore pre-finetuned models for our evaluations, including the artifact analysis ones, as they would simply be non-sensical given the randomly initialized classification head.

6 Results

6.1 Expected vs. Actual # of Top-k Artifacts

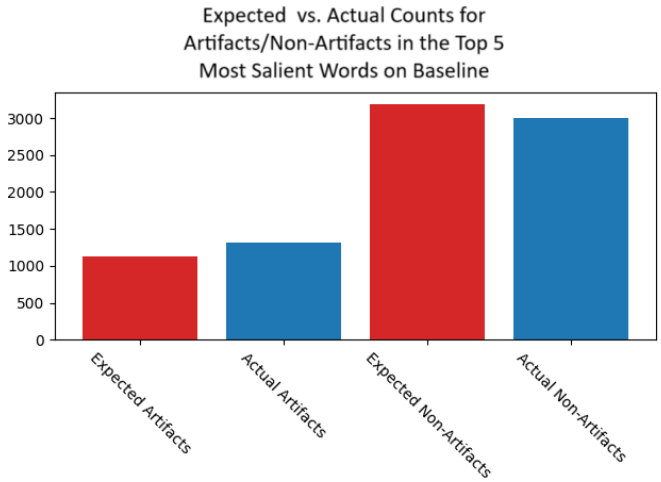

Figure 3: Bar plot displaying the expected vs. observed # of artifacts in the top-k most salient words after fine-tuning the baseline model (opt-1.3)

We used the procedure described in 4.2 to generate Figure 3, in which we trained the baseline opt-1.3 model on SST2. We observe that actual number of artifacts that appeared in the top 5 most salient words is slightly higher that what we expected based on the fraction of artifacts that appear in the dataset. This supports, albeit not strongly, the hypothesis that fine-tuning a model in a dataset with artifacts causes the model to give more saliency to artifacts.

6.2 Artifact analysis on Model with Modified Loss Function

When we computed the influence of artifacts on the model with the modified loss functions, we found hardly any change in the number of actual artifacts and non-artifacts in the top 5 most salient words:

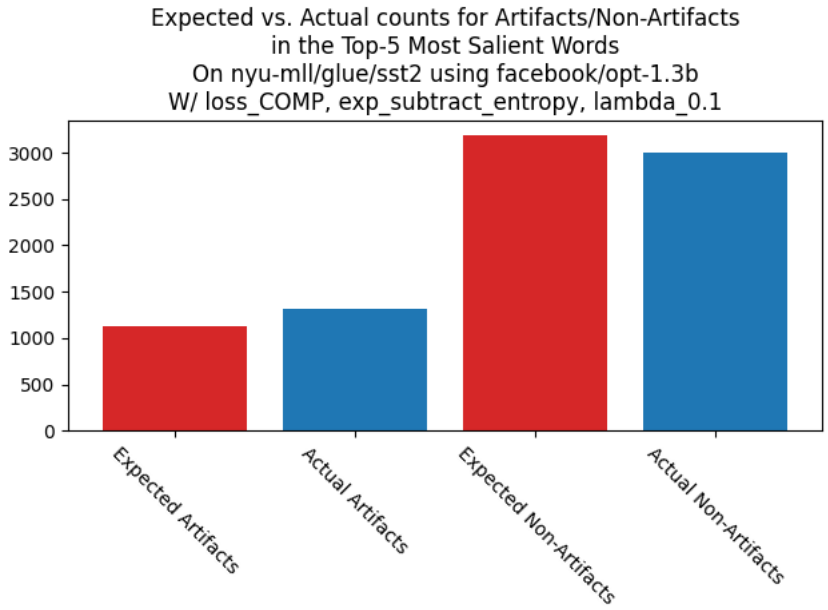

Figure 4: Bar plot displaying the expected vs. observed # of artifacts in the top-k most salient words after finetuning the opt-1.3 with a modified loss function

Above is the results generated by running the algorithm described in section 4.2, after finetuning opt-1.3 with one of our modified loss functions (in particular we used the gradient-based method and the choice $\mathcal{F}(a, b) = \frac{\exp(a)}{\exp(b)}$) on SST2. The expected and actual counts of artifacts are nearly identical to the that of the baseline discussed in section 6.2. This suggests that fintuning with our modified loss function does not significantly impact the relative saliency of artifacts in the dataset, although the relative saliency of artifacts was not much higher than what we expected under the null hypothesis to begin with. One possible reason that the influence of artifacts did not change with our loss function is that the pretained model itself is so large that adding our additional loss function did not noticeably shift the saliency of any artifacts. Another possible reason is that the correlations of artifacts and their respective labels are so strong in the dataset itself that it could not be overcome in the model.

6.3 Visualizing Saliency Maps

We use these fine-tuned models to calculate S_x for data points in SST2. Visualizing these distributions will give us a sense of how these models "view" the input tokens when determining output.

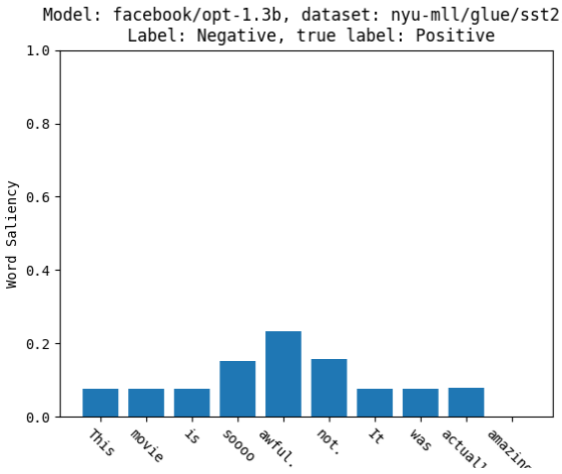

Figure 5: Example of saliency distribution for a datapoint in SST2, on the baseline loss function

The low entropy of these distributions clearly demonstrates that the competency assumption is being violated within these models. This is a sensible conclusion as they were finetuned on datasets that had lots of artifacts in them, as demonstrated in Section 5.1. We have also seen that saliency maps are effective in capturing this effect, as artifacts appear more often in the top-k most salient words for an input. We finally empirically observe low entropy distributions of saliency within the finetuned model. It is therefore reasonable to assume that motivating the model to increase entropy within its saliency maps will lessen the problematic impact that artifacts can carry.

6.4 Validation

Results on the validation set are tabulated in tables 1 and 2 in the Appendix. We observe that applying gradient methods to our loss function generally has better F1 score and accuracy compared to applying masking. We also see that most models have worse accuracy and F1 score than the baseline, the exception being Gradient Subtract Loss, which uses gradient based methods and $\mathcal{F}(a, b) = a - b$. This model has comparable/somewhat higher accuracy than the baseline, but not by a significant amount. Many results for SNLI are missing due to limited compute during training.

6.5 Evaluation on Adversarial Datasets

As a final test, we chose to evaluate a standard fine-tuned LLM and one fine-tuned using our modified loss function on adversarial text examples. The results are tabulated in tables 3 and 4 in the Appendix. We observe that Gradient Inverse loss has significantly lower F1 score than the baseline and somewhat lower accuracy. Gradient Subtract loss also has comparable/somewhat higher F1 scores and accuracy than the baseline, but again most models have lower F1 score and accuracy than the baseline. It is also clear that almost across the board, masking performs significantly worse than the gradient based method. This is likely due to the fundamental meaning and grammatical structure changing when removing/masking a token. This results in the language model assigning the input sentence a much lower probability of occurring, resulting in less meaningful saliency scores.

7 Discussion

From our evaluation, we noticed that while using our methods would sometimes result in improvements in the F1 score for OPT models, it did not have a noticeable improvement in accuracy. However, we noticed that using our methods would sometimes noticeably improve both the F1 score and accuracy for Llama models. For example, the Gradient Subtract Loss model with $\lambda = 0.1$ had an F1 score of 0.7 and an accuracy of 0.716 for the Adversarial SST2 dataset, compared to the baseline model's score of 0.649 and 0.642 respectively, indicating a 0.061 improvement in the F1 score and a 0.074 improvement in the accuracy score.

With the high variance in our adversarial results, as described in section 6.5, it is inconclusive as to weather our loss functions are removing spurious correlations. One possible explation is that hyper parameters are crucial in order for our loss functions to perform well. As shown in table 1, with certain hyper parameter settings, our loss function has significant gains over the baseline method in terms of f1 and accuracy in the adversarial test. We intend to continue working on this project, testing a larger pool of hyper parameters, mainly settings of λ and $\mathcal{F}$, as well as experimenting with more models, which was an aspect we could not afford to test for this project due to compute and time limitations.

It is also possible that artifacts are much more insidious to a model, stemming from artifacts within the much larger pre-training corpus, and the finetuning dataset itself. Merely modifying the fine-tuning procedure is not enough to diffuse spurious correlations, as the lack of any significant change in the Artifact analysis described in section 6.2 shows. Thus, a more data-centric approach to diffusing artifacts may be required.

Finally, we would like to experiment with other approaches to determining saliency scores. One such involves applying the classification head onto every token, i.e. dense-pooling, and using this to measure the saliency of input features.

8 Contributions

Hari: Designing and implementing the custom loss function, finetuning code, managing the codebase, training the models, write-up loss function section, being awesome.

Rei: Helping come up with alternative loss and entropy functions, writing alternative loss and entropy function code, writing alternative dense classification head model code, tabulating results, writing up the report (introduction, background and related work, datasets, discussion).

Varun: Code for calculating expected vs actual number of topk artifacts, code for creating gradient-based saliency maps, paper introduction, abstract, background and related work,

Yafqa: Formulation of artifact detection, coding up of artifact detection for each dataset, plotting graphs of artifacts, coming up with statistical demonstration of artifacts' influence on language models, writing up all sections pertaining to artifact analysis.

9 Appendix

Table 1: OPT 1.3B Validation Set Performance

Model	**SNLI F1**	**SST2 F1**	**SNLI Accuracy**	**SST2 Accuracy**
Baseline	0.891	0.936	0.892	0.937
Gradient Exponential Subtract Loss ($\lambda = 0.1$)	0.949	0.896	0.947	0.897
Gradient Exponential Subtract Loss ($\lambda = 0.5$)		0.583		0.670
Gradient Exponential Subtract Loss ($\lambda = 1$)	0.792	0.670	0.795	0.527
Gradient Inverse Loss ($\lambda = 0.1$)		0.920		0.923
Gradient Inverse Loss ($\lambda = 0.5$)		0.943		0.943
Gradient Inverse Loss ($\lambda = 1$)		0.836		0.856
Gradient Subtract Loss ($\lambda = 0.1$)	0.867	0.949	0.869	0.948
Gradient Subtract Loss($\lambda = 0.5$)		0.939		0.939
Gradient Subtract Loss ($\lambda = 1$)	0.878	0.942	0.880	0.942
Mask Exponential Subtract Loss ($\lambda = 0.1$)	0.493	0.786	0.501	0.749
Mask Exponential Subtract Loss ($\lambda = 0.5$)		0.787		0.761
Mask Exponential Subtract Loss ($\lambda = 1$)		0.760		0.75
Mask Inverse Loss ($\lambda = 0.1$)		0.814		0.786
Mask Inverse Loss ($\lambda = 0.5$)		0.823		0.795
Mask Inverse Loss ($\lambda = 1$)		0.797		0.779
Mask Subtract Loss ($\lambda = 0.1$)	0.496	0.514	0.504	0.514
Mask Subtract Loss ($\lambda = 0.5$)		0.422		0.5
Mask Subtract Loss ($\lambda = 1$)	0.586	0.319	0.587	0.365

Table 2: Llama 2 7B Validation Set Performance

Model	**SST2 F1**	**SST2 Accuracy**
Baseline	0.961	0.960
Gradient Exponential Subtract Loss ($\lambda = 0.1$)	0.959	0.958
Gradient Exponential Subtract Loss ($\lambda = 0.5$)	0.952	0.951
Gradient Exponential Subtract Loss ($\lambda = 1$)	0.674	0.510
Gradient Inverse Loss ($\lambda = 0.1$)	0.306	0.517
Gradient Inverse Loss ($\lambda = 0.5$)	0.964	0.963
Gradient Inverse Loss ($\lambda = 1$)	0.941	0.938
Gradient Subtract Loss ($\lambda = 0.1$)	0.963	0.962
Gradient Subtract Loss($\lambda = 0.5$)	0.965	0.964
Gradient Subtract Loss ($\lambda = 1$)	0.966	0.966

Table 3: OPT 1.3B Adversarial Dataset Performance (ADV-MNLI, ADV-SST2)

Model	**MNLI F1**	**SST2 F1**	**MNLI Accuracy**	**SST2 Accuracy**
Baseline	0.371	0.448	0.372	0.534
Gradient Exponential Subtract Loss ($\lambda = 0.1$)	0.371	0.537	0.372	0.486
Gradient Exponential Subtract Loss ($\lambda = 0.5$)		0.025		0.480
Gradient Exponential Subtract Loss ($\lambda = 1$)	0.245	0.670	0.256	0.527
Gradient Inverse Loss ($\lambda = 0.1$)		0.274		0.426
Gradient Inverse Loss ($\lambda = 0.5$)		0.371		0.473
Gradient Inverse Loss ($\lambda = 1$)		0.101		0.399
Gradient Subtract Loss ($\lambda = 0.1$)	0.364	0.545	0.372	0.527
Gradient Subtract Loss($\lambda = 0.5$)		0.453		0.527
Gradient Subtract Loss ($\lambda = 1$)	0.411	0.439	0.413	0.5
Mask Exponential Subtract Loss ($\lambda = 0.1$)	0.196	0.608	0.314	0.520
Mask Exponential Subtract Loss ($\lambda = 0.5$)		0.472		0.426
Mask Exponential Subtract Loss ($\lambda = 1$)		0.48		0.426
Mask Inverse Loss ($\lambda = 0.1$)		0.491		0.426
Mask Inverse Loss ($\lambda = 0.5$)		0.477		0.492
Mask Inverse Loss ($\lambda = 1$)		0.298		0.459
Mask Subtract Loss ($\lambda = 0.1$)	0.224	0.514	0,322	0.514
Mask Subtract Loss ($\lambda = 0.5$)		0.422		0.5
Mask Subtract Loss ($\lambda = 1$)	0.194	0.319	0.306	0.365

Table 4: Llama 2 7B Adversarial Dataset Performance

Model	**SST2 F1**	**SST2 Accuracy**
Baseline	0.649	0.642
Gradient Exponential Subtract Loss ($\lambda = 0.1$)	0.667	0.669
Gradient Exponential Subtract Loss ($\lambda = 0.5$)	0.638	0.655
Gradient Exponential Subtract Loss ($\lambda = 1$)	0.643	0.520
Gradient Inverse Loss ($\lambda = 0.1$)	0.585	0.473
Gradient Inverse Loss ($\lambda = 0.5$)	0.649	0.649
Gradient Inverse Loss ($\lambda = 1$)	0.618	0.608
Gradient Subtract Loss ($\lambda = 0.1$)	0.7	0.716
Gradient Subtract Loss($\lambda = 0.5$)	0.671	0.669
Gradient Subtract Loss ($\lambda = 1$)	0.643	0.655

References

[1] C. E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R. Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62, 1936.

[2] Shuoyang Ding and Philipp Koehn. Evaluating saliency methods for neural language models, 2021.

[3] Matt Gardner, William Merrill, Jesse Dodge, Matthew E. Peters, Alexis Ross, Sameer Singh, and Noah A. Smith. Competency problems: On finding and removing artifacts in language data, 2021.

[4] Fuxiao Liu, Paiheng Xu, Zongxia Li, Yue Feng, and Hyemi Song. Towards understanding in-context learning with contrastive demonstrations and saliency maps, 2024.

[5] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.