Abstract

In most machine learning tasks, we evaluate a model \( \mathcal{M} \) on a given data population \( S \) by measuring a population-level metric \( \mathcal{F}(S; \mathcal{M}) \). Examples of such evaluation metric \( \mathcal{F} \) include precision/recall for (binary) recognition, the F1 score for multi-class classification, and the BLEU metric for language generation. On the other hand, the model \( \mathcal{M} \) is trained by optimizing a sample-level loss \( G(S_t; \mathcal{M}) \) at each learning step \( t \), where \( S_t \) is a subset of \( S \) (a.k.a. the mini-batch). Popular choices of \( G \) include cross-entropy loss, the Dice loss, and sentence-level BLEU scores. A fundamental assumption behind this paradigm is that the mean value of the sample-level loss \( G \), if averaged over all possible samples, should effectively represent the population-level metric \( \mathcal{F} \) of the task, such as, that \( \mathbb{E}[G(S_t; \mathcal{M})] \approx \mathcal{F}(S; \mathcal{M}) \).

In this paper, we systematically investigate the above assumption in several NLP tasks. We show, both theoretically and experimentally, that some popular designs of the sample-level loss \( G \) may be inconsistent with the true population-level metric \( \mathcal{F} \) of the task, so that models trained to optimize the former can be substantially sub-optimal to the latter, a phenomenon we call it, Simpson’s bias, due to its deep connections with the classic paradox known as Simpson’s reversal paradox in statistics and social sciences.

1 Introduction

Consider the following standard and general paradigm of NLP training: given a corpus \( S \) consisting of \( n \) samples, each indexed by \( i = \{1, \ldots, n\} \), the training of NLP model \( \mathcal{M} \) aims at optimizing a corpus-level objective \( \mathcal{F}(S; \mathcal{M}) \). For example, a popular training method follows the maximum likelihood estimation (MLE) principle, in which a sample is a \((x_i, y_i)\) pair with \( x_i \) being a decision context, which is usually one or more sentences in NLP tasks, and \( y_i \) being a desired atomic decision, which is usually a token in generative tasks or a class label in discriminative tasks. The corpus-level objective \( \mathcal{F} \) that MLE-oriented training aims at maximizing is the log-likelihood of the whole corpus: \( \mathcal{F}_{\text{MLE}}(S; \mathcal{M}) = \sum_{i=1}^{n} \log \mathcal{M}(x_i, y_i) \).

The MLE objective is relatively easy to optimize because we can construct a sample-level loss function \( G(i; \mathcal{M}) \) such that the sample average \( \bar{G}(S; \mathcal{M}) = \frac{1}{n} \sum_{i=1}^{n} G(i; \mathcal{M}) \) can “effectively represent” \( \mathcal{F}_{\text{MLE}}(S; \mathcal{M}) \) as a surrogate objective of the optimization. Specifically, since \( \mathcal{F}_{\text{MLE}} \) itself is additive with respect to the samples in \( S \), we can simply take the CE loss \( G_{\text{MLE}}(i; \mathcal{M}) = \mathcal{F}_{\text{MLE}}(\{i\}; \mathcal{M}) \), which gives

\[
\mathcal{F}_{\text{MLE}}(S; \mathcal{M}) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{F}_{\text{MLE}}(\{i\}; \mathcal{M}) = \frac{1}{n} \sum_{i=1}^{n} \log \mathcal{M}(x_i, y_i) \propto \mathcal{F}_{\text{MLE}}(S; \mathcal{M}).
\]

The average form of \( \mathcal{F}_{\text{MLE}} \) admits efficient stochastic-gradient optimization (which requires the objective to be a population mean such that its gradient can be unbiasedly estimated by the gradient of the sample mean over a random mini-batch), and the proportionality between \( \mathcal{F}_{\text{MLE}} \) and \( \mathcal{F}_{\text{MLE}} \) guarantees that an optimal (better) solution of the former is also an optimal (better) solution of the latter.

However, it is rare that a task directly uses \( \mathcal{F}_{\text{MLE}} \) as the end-to-end evaluation metric. Instead, common evaluation metrics used in practice include accuracy, precision/recall/F1 (for discriminative tasks), and BLEU (Papineni et al. 2002) (for machine translation and other language generation tasks). While a model trained with \( G_{\text{MLE}} \) may well optimize the corresponding MLE objective \( \mathcal{F}_{\text{MLE}} \), it does not necessarily optimize the true evaluation metric of the task. For this reason, researchers have proposed to optimize alternative objective \( \mathcal{F} \) that is closer to, or in some cases equal to, the true evaluation metric used at testing time. For example, the Dice loss (Li et al. 2020) has been recently proposed for tasks such as Paraphrase Similarity Matching (PSM) and Named Entity Recognition (NER) because of its similarity to the F1 metric used in these tasks. Similarly, sentence-level BLEU scores have been used in sentence-level training for machine translation due to its correspondence to the true corpus-level BLEU metric (Ranzato et al. 2016; Wu et al. 2016; Edunov et al. 2018).

Unfortunately, these alternative learning objectives posed new challenges in optimization. Specifically, metrics like F1 and BLEU (and many others) are not sample-separable, meaning that they cannot be converted proportionally or monotonically into an averaged form \( \bar{G} \) as in the case of MLE. Consequently, while the intended objectives \( \mathcal{F}_{\text{F1}} \) and \( \mathcal{F}_{\text{BLEU}} \) are more aligned with the evaluation metric of the corresponding tasks, what the training algorithms are truly
optimizing is usually the averaged-form objectives \( \tilde{F}_{\text{F1}} \) and \( \tilde{F}_{\text{BLEU}} \), and models thus trained could improve the averaged objective \( \tilde{F} \) while at the same time being worse with respect to the intended objective \( F \).

In this paper, we call the disparity mentioned above, Simpson’s bias. It is a bias between non-separably aggregated objective \( F \) and its corresponding averaged form \( \tilde{F} \). The name is inspired by the classic paradox known as Simpson’s reversal in statistics and social sciences, which refers to a class of conflicting conclusions obtained when comparing two “candidates” based on their aggregated performance and based on their per-case performance. In the following, we will give a systematic analysis on how a similar effect can widely arise in the context of machine learning when designing sample-level loss for many popular metrics including precision, recall, Dice Similarity Coefficient (DSC), Macro F1, and BLEU. We then experimentally examine and verify the practical impacts of the Simpson’s bias on the training of state-of-the-art models in three different NLP tasks: Paraphrase Similarity Matching (with the DSC metric), Named Entity Recognition (with the Macro-F1 metric), and Machine Translation (with the BLEU metric).

2 The Simpson’s Bias

As discussed in the last section, the ultimate goal of NLP training is to optimize a set function \( F(S; M) \) which is a corpus-wise aggregated measurement of model \( M \)’s performance on given data set \( S = \{1 \ldots n\} \). On the other hand, the model \( M \) is typically trained by following the gradient direction of a sample-level loss \( G(i; M) \) on random sample \( i \in S \). \(^1\) Such training is expected to find an extreme point of the averaged performance \( \tilde{F}_G(S; M) = \frac{1}{n} \sum_{i \in S} G(i; M) \).

We will pay special attention to the “naive” sample-level loss \( G_F(i; M) = F(\{i\}; M) \), which uses the same metric \( F \) to measure a single sample. We use the \( \tilde{F} \) without subscript to denote the corpus-wise averaged performance corresponding to this particular sample loss \( G_F \), so \( \tilde{F} = \frac{1}{n} \sum_{i \in S} G_F(i; M) \). Note that every well-defined set function \( F \) is conjugated with such an \( \tilde{F} \), which is the arithmetic average of \( F \) over all singletons of \( S \). On the other hand, the function \( F \) itself, when used as a performance metric in machine learning, often involves some form of “complex averaging” over \( S \) as well. We are interested to understand whether, or to what extent, a model optimized for the arithmetic average \( \tilde{F} \) can also perform well w.r.t. the “complex” average \( F \), for various specific forms of \( F \neq F_{\text{MLE}} \).

2.1 Special case 1: Ratio of Sums (RoS)

This is a very common family of metric \( F \), which computes the ratio of two summations over the set \( S \). Let \( A_i \) and \( B_i \) be two quantities defined on each sample \( i \), the RoS family of \( F \) is generally in the form of

\[
F(S) = \frac{\sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} B_i}
\]

\(^1\)When mini-batch is used, the algorithm generates a random batch \( S_t \subset S \) at each optimization step \( t \) and follows the gradient direction of batch-wise averaged loss \( \frac{1}{|S_t|} \sum_{i \in S_t} G(i; M) \).

and the corresponding “naively”-averaged metric is

\[
\tilde{F}(S) = \frac{1}{n} \sum_{i=1}^{n} F(\{i\}) = \frac{1}{n} \sum_{i=1}^{n} \frac{A_i}{B_i}
\]

In the above, we have omitted \( M \), which is considered given in this section. As a best case, \( F \) of the RoS family equals \( \tilde{F} \) in the following two conditions:

- **Type-1**: If \( B_i = B \) for some constant \( B \), then
  \[
  \tilde{F}(S) = \frac{1}{n} \sum_{i=1}^{n} \frac{A_i}{B} = \frac{1}{nB} \sum_{i=1}^{n} A_i = \frac{\sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} B_i} = F(S)
  \]

- **Type-2**: If \( \frac{A_i}{B_i} = r \) for some constant \( r \), then
  \[
  \tilde{F}(S) = \frac{1}{n} \sum_{i=1}^{n} \frac{A_i}{B_i} = \frac{\sum_{i=1}^{n} rB_i}{\sum_{i=1}^{n} B_i} = \frac{\sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} B_i} = F(S)
  \]

Depending on precise definitions of \( A_i \) and \( B_i \), the RoS family subsumes many concrete metrics used in NLP tasks. We discuss three popular RoS metrics in the following.

**Scenario 1a: Accuracy**

Let \( y_i \) be a ground-truth decision on sample \( i \) and \( \hat{y}_i \) the decision output by the model \( M \), the **accuracy** of \( M \) on data set \( S \) of size \( n \) is

\[
F_{\text{AC}} = \frac{\sum_{i=1}^{n} \mathbb{I}(y_i = \hat{y}_i)}{n}
\]

which is a special case of (1) with \( A_i = \mathbb{I}(y_i = \hat{y}_i) \) and \( B_i = 1 \), where \( \mathbb{I}() \) is the indicator function.

Accuracy is the simplest case in our analysis, which does not suffer from the Simpson’s bias at all, as it satisfies the type-1 condition above. In other words, optimization based on the naive sample-level loss \( G_{\text{AC}}(i; M) = \mathbb{I}(y_i = \hat{y}_i) \) will maximize exactly the accuracy \( F_{\text{AC}} = \tilde{F}_{\text{AC}} \).

Note that in supervised learning, the sample loss \( G \) may further need to be differentiable, in which case the indicator variable \( \mathbb{I}(y_i = \hat{y}_i) \) is usually approximated in practice. For example in binary recognition problems, which ask to judge if each sample \( i \) is positive or negative (w.r.t. some feature of interest), the model \( M \) is usually set to output a probability \( p_i = M(x_i) \), and differentiable sample losses such as \( (p_i - y_i)^2 \) are used, essentially as smoothed variants of the discrete loss \( \mathbb{I}(y_i \neq \hat{y}_i) = 1 - \mathbb{I}(y_i = \hat{y}_i) \).

We do not consider errors from such differentiablization tricks as part of the Simpson’s bias under discussion, as the former is mostly a limit of only specific (types of) learning algorithms. In contrast, the Simpson’s bias that we are studying in this paper is concerned more with intrinsic properties of the learning objectives themselves. For example, the exact sample-level accuracy \( G_{\text{AC}}(i; M) = \mathbb{I}(y_i = \hat{y}_i) \) can indeed be directly optimized through reinforcement learning algorithms, in which case the learning algorithm is equivalently optimizing exactly the corpus-wise accuracy \( F_{\text{AC}} \).

**Scenario 1b: Precision/Recall**

While being applicable to almost all discrete decision tasks, accuracy can be problematic for tasks with imbalanced data. For example, in binary recognition problems, a model always outputting negative would have very high accuracy if positive samples are
rare. Precision and recall are standard evaluation metrics used in binary recognition tasks to solve this problem.

In binary recognition problems, let $y_i \in \{0, 1\}$ be the true label of sample $i$, $y_i = 0$ for negative sample and $y_i = 1$ for positive sample. Let $\hat{y}_i \in \{0, 1\}$ be the predicted label by model $M$, $\hat{y}_i = 0$ for negative output and $\hat{y}_i = 1$ for positive output. The precision on a data set $S$ of size $n$ is

$$F_p = \frac{\sum_{i=1}^{n} y_i \hat{y}_i}{\sum_{i=1}^{n} \hat{y}_i}.$$  \hspace{1cm} (4)

It is clear that $F_p$ can be seen as a RoS metric with $A_i = y_i \hat{y}_i$ and $B_i = \hat{y}_i$. But strictly speaking, $F_p$ is not a completely well-defined metric as its denominator $\sum_{i=1}^{n} \hat{y}_i$ can be zero. This issue becomes more evident when we try to write its naively-conjugated form $\tilde{F}_p = \frac{1}{n} \sum_i \frac{y_i \hat{y}_i}{\hat{y}_i}$. For this reason, we turn to consider the smoothed precision

$$\tilde{F}_p = \frac{\gamma + \sum_{i=1}^{n} y_i \hat{y}_i}{\gamma + \sum_{i=1}^{n} \hat{y}_i}$$  \hspace{1cm} (5)

which is a genuine RoS metric that subsamples the vanilla precision $F_p$ with $\gamma = 0$, and its average form

$$\tilde{F}_p = \frac{1}{n} \sum_i \tilde{F}_p(i) = \frac{1}{n} \sum_i \frac{\gamma + y_i \hat{y}_i}{\gamma + \hat{y}_i}$$  \hspace{1cm} (6)

is always well defined for $\gamma \neq 0, -1$.

Unlike accuracy, the (smoothed) precision metrics do not satisfy either of the two equality conditions above, and may suffer from the Simpson’s bias in general. This is especially true for $\gamma \in [0, 1]$ which is the commonly used smoothing constant in existing practice, as Section 4 will later demonstrate. However, the following theorem shows that the Simpson’s bias for smoothed precision may disappear under a special (and unusual) smoothing term $\gamma^* < 0$, such that the smoothed precision $\tilde{F}_p$, equals precisely to its conjugate metric $\tilde{F}_p$, under this special $\gamma^*$.

**Theorem 1** $\tilde{F}_p = \tilde{F}_p$ if $\gamma = -\frac{n \sum_i y_i \hat{y}_i}{n} \sum_i \hat{y}_i \geq 2$.

More importantly, there turns out to be also a special smoothing term $\gamma^p < 0$, such that the averaged sample-level precision smoothed by this particular $\gamma^p$ happens to equal precisely the original precision metric $F_p$.

**Theorem 2** $\tilde{F}_p = \tilde{F}_p$ if $\gamma = \sum_i y_i \hat{y}_i - 1$.

See the proofs in appendix of our full paper.

According to Theorem 2, the special smoothing term $\gamma^p$ is the negated negative-output-rate of the model $M$. The theorem says that although the original precision metric does suffer from the Simpson’s bias (in the sense that $F_p \neq \tilde{F}_p$), the bias can be completely resolved by using the special smoothing term $\gamma^p$. Note that $\gamma^p$, as a negative smoothing term, is outside the typical value range of smoothing-term tuning in previous works (which usually used $\gamma \in [0, 1])$. \footnote{2We also remark that the smoothing term was previously only used to make the precision metric well defined on singleton samples, not for solving the Simpson’s bias.}

Finally, the recall metric is symmetrically defined as $F_r = \frac{\sum_{i=1}^{n} y_i \hat{y}_i}{\sum_{i=1}^{n} y_i}$, thus all the observations about precision as discussed also symmetrically apply to recall. In particular, we have $F_{r\gamma} = F_{r}$ for $\gamma = \sum_i y_i/n - 1$.

**Scenario 1c: Dice Coefficient** Dice similarity coefficient (DSC) is a measure to gauge the similarity of two overlapped (sub-)sets. In binary recognition problems, DSC is used as a performance metric that combines precision and recall.

Specifically, the DSC metric is the harmonic mean of precision and recall. Following the same formulation with Scenario 1.b, we can write

$$F_{DSC}(S) = \frac{2 \cdot F_p(S) \cdot F_r(S)}{F_p(S) + F_r(S)} = \frac{n \sum_{i=1}^{n} 2 y_i \hat{y}_i}{\sum_{i=1}^{n} (y_i + \hat{y}_i)},$$  \hspace{1cm} (7)

which is a RoS metric with $A_i = 2 y_i \hat{y}_i$ and $B_i = y_i + \hat{y}_i$. We can also similarly generalize DSC to smoothed variant

$$F_{DSC\gamma}(S) = \frac{\gamma + \sum_{i=1}^{n} 2 y_i \hat{y}_i}{\gamma + \sum_{i=1}^{n} (y_i + \hat{y}_i)},$$  \hspace{1cm} (8)

which has conjugated average-form

$$F_{DSC\gamma} = \frac{1}{n} \sum_i G_{DSC\gamma}(i) = \frac{1}{n} \sum_i \frac{\gamma + 2 y_i \hat{y}_i}{\gamma + y_i + \hat{y}_i}$$  \hspace{1cm} (9)

The following theorem shows an interesting connection between DSC and accuracy. See the proofs in appendix of our full paper.

**Theorem 3** $F_{DSC\gamma} = 1 - \frac{[y_i \neq \hat{y}_i]}{(1 + \gamma)n}$ for $\gamma \neq 0, -1, -2$.

When $\gamma \approx 0$, the right-hand side of Theorem 3 is very close to the value of accuracy. So, it turns out that averaging the nearly un-smoothed sample-level DSC gives us the corpus-level accuracy: $F_{DSC\gamma} \approx F_{KC}$ for $\gamma \approx 0$. In other words, Theorem 3 implies that the original DSC metric $F_{DSC}$ (which is approximately $F_{DSC\gamma}$ with $\gamma \approx 0$, see (8)) does not only have the Simpson’s bias, but the bias in this metric is so significant that its average-form conjugate $F_{DSC\gamma}$ with $\gamma \approx 0$ has been completely distorted towards another metric (i.e. towards accuracy $F_{KC}$).

### 2.2 Special case 2: Macro-F1

The DSC metric can be further extended to multi-class classification problems, in which the model $M$ is asked to classify each sample input $x_i$ into one of $K$ predefined classes.

The ground-truth label $y_{i,k} \in \{0, 1\}$ is a categorical variable whose $k$-th component $y_{i,k}$ is 1 if sample $i$ is from class $k$, otherwise $y_{i,k} = 0$. The decision of the model is similarly encoded by a one-hot vector $\hat{y}_{i,k} = \text{hardmax}(p_i) \in \{0, 1\}$, where $p_i = M(x_i) \in \{0, 1\}^K$ is the model output under $x_i$.

For given class $k$, the decision of the model is making binary recognition on the particular class $k$, thus all the metrics discussed so far applies in a per-class sense. Specifically, the model’s precision for class $k$ is $P_k(S) = \frac{\sum_{i=1}^{n} y_{i,k} \hat{y}_{i,k}}{\sum_{i=1}^{n} \hat{y}_{i,k}}$, and its recall for class $k$ is $R_k(S) = \frac{\sum_{i=1}^{n} y_{i,k} \hat{y}_{i,k}}{\sum_{i=1}^{n} y_{i,k} \hat{y}_{i,k}}$. The DSC for class $k$ is, accordingly, $DSC_k(S) = \frac{\sum_{i=1}^{n} y_{i,k} \hat{y}_{i,k}}{\sum_{i=1}^{n} y_{i,k} + \hat{y}_{i,k}}$. The
**F1 score** of the model is the mean DSC value averaged over all classes, denoted as

$$F_1 = \frac{1}{K} \sum_{k=1}^{K} DSC_k(S) = \frac{\sum_{i} \gamma + \gamma y_{i,k} + \gamma y_{i,k}^{+}}{\gamma + \gamma y_{i,k} + \gamma y_{i,k}^{+} / K}$$

(10)

The F1 metric is a linear sum of several RoS metrics, but itself is not a RoS metric. The corresponding (smoothed) average-form F1 is

$$F_1 = \frac{1}{n} \sum_{i=1}^{n} F_1^i = \frac{\sum_{i=1}^{n} \gamma + \gamma y_{i,k} + \gamma y_{i,k}^{+}}{\gamma + \gamma y_{i,k} + \gamma y_{i,k}^{+} / Kn}$$

(11)

From Theorem 3 we know that the average-form F1 (that is, $F_1^i$ with $\gamma \approx 0$) is equivalent to an “mean-accuracy-over-class” metric, which is different from the aggregated F1 metric and is different from the multi-class accuracy metric actually used in multi-classification tasks too.

Despite the Simpson’s bias in F1 as discussed, the average-form F1(11) has inspired Milletari, Navab, and Ahmadi (2016) to introduce the *Dice Loss*, defined as

$$F_{DL}(S) = \frac{1}{n} \sum_{i} G_{DL}(i) = \frac{\sum_{i,k} \gamma + 2 \gamma y_{i,k} + \gamma y_{i,k}^{+}}{\gamma + \gamma y_{i,k} + \gamma y_{i,k}^{+} / Kn}$$

(12)

Besides the differentiable trick, the Dice loss (12) further uses the squared terms $y_{i,k}^{+}$ and $y_{i,k}^{+}$ in denominator for faster training. Li et al. (2020) has proposed to adopt the Dice loss to train models in a number of NLP tasks.

### 2.3 Special case 3: BLEU

BLEU is a widely used evaluation metric in machine translation(MT) and question answering (QA). Given a parallel corpus $S$ consisting of $n$ sentence pairs $(X^{(i)}, Y^{(i)})$, $X^{(i)}$ being the source sentence and $Y^{(i)}$ a reference translation, the MT model $M$ will generate a translation $\hat{Y}^{(i)}$ for each $i \in \{1 \ldots n\}$. The BLEU score of the model $M$ on such a data set $S$ is defined as $BLEU(S;M) = \sum_{i=1}^{n} \log \frac{\text{GM}_{k=1}^{4}(\frac{H_{k}^{(i)}}{L_{k}^{(i)}}) \cdot \min \left(1 - \frac{M_{k}^{(i)}}{L_{k}^{(i)}}, 1\right)}{\log \text{BLEU}(S;M)} - 1$

where $L_{k}^{(i)}$ is the total number of n-grams of length $k$ in $\hat{Y}^{(i)}$, $H_{k}^{(i)}$ is the number of “matched” n-grams of length $k$ in $\hat{Y}^{(i)}$, $M_{k}^{(i)}$ is the total number of 1-grams in $Y^{(i)}$, and $\text{GM}_{k=1}^{4}$ means taking the geometric mean over $k = 1, 2, 3, 4$.

To subsume the BLEU metric into our framework, define

$$F_{BLEU}(S;M) = \log \text{BLEU}(S;M) - 1$$

$$= \frac{1}{4} \log \left(\sum_{k=1}^{4} \frac{H_{k}^{(i)}}{L_{k}^{(i)}}\right) + \frac{1}{4} \log \left(\sum_{k=1}^{4} \frac{H_{k}^{(i)}}{L_{k}^{(i)}}\right) + \frac{1}{4} \log \left(\sum_{k=1}^{4} \frac{H_{k}^{(i)}}{L_{k}^{(i)}}\right)$$

$$+ \frac{1}{4} \log \left(\sum_{k=1}^{4} \frac{H_{k}^{(i)}}{L_{k}^{(i)}}\right) - \max \left(\frac{M_{k}^{(i)}}{L_{k}^{(i)}}, 1\right)$$

(13)

Note that in $F_{BLEU}$, a sample is a sentence, and the metric computes a *sentence-level BLEU* score (Chen and Cherry 2014) for each sentence $i$, then takes the arithmetic mean over all sentence-level scores. Sentence-level training could be conducted based on $F_{BLEU}$, as have been explored by many authors (Ranzato et al. 2016; Shen et al. 2016; Wu et al. 2016; Bahdanau et al. 2017; Wu et al. 2018; Edunov et al. 2018), if the sentence-averaged BLEU indeed serves as a good proxy to the true evaluation metric $F_{BLEU}$, a presumption that we will experimentally examine in later sections.

### 3 Connections to Simpson’s Paradox

Our naming of the bias between corpus-level metric $F$ and its average-form conjugate $\hat{F}$ is largely inspired by its connection with the famous notion, *Simpson’s reversal paradox*, which we will explain in this section.

*Simpson’s reversal* often refers to the statistical observation that a candidate method/model is better in each and every case, but is worse in terms of the overall performance. For example, let $M_1$ be a new medical treatment that is better than the baseline method $M_0$ in terms of survival rate $F$ for both the group of male patients and the group of female patients, it turns out that $M_1$ could have a lower survival rate than $M_0$ for the combined group of all patients, as famously shown by Blyth (1972).

Many people feel surprising, or even paradoxical, when they observe the Simpson’s reversal. Blyth (1972) was the first to call this phenomenon. *Simpson’s paradox*, named after Edward H. Simpson for his technical notes (Simpson 1951) that proposed to study the phenomenon more carefully. On the other hand, Simpson’s reversal, as a mathematical fact, is not too rare in real-world experiences. Pavlides and Perlman (2009) show that the reversal occurs in about 2% of all the possible $2 \times 2 \times 2$ contingency tables. It is then interesting to ask why people consider a not-so-uncommon phenomenon psychologically surprising – the paradoxical feeling appears to suggest some deeply held conviction in people’s mind that the Simpson’s reversal has clashed with.

The *sure-thing principle* has been hypothesized to be such a contradictory conviction behind the Simpson’s paradox (Pearl 2014), which validly asserts that a method that helps in every case must be beneficial in terms of the averaged performance under any mixture distribution. In the medical example above, for instance, the new method $M_1$ improves survival rate for both males and females, which by
the sure-thing principle does entail that $M_1$’s average survival rate under any given gender ratio must improve. However, it is often overlooked that the aggregated survival rate of a method (over both males and females) is not a simple average of its per-gender survival rate, but depends on the specific gender ratio that the method is facing (which may vary between methods). People might feel the Simpson’s reversal paradoxical if they overlooked the difference between the averaged performance and the aggregated performance, in which case the observed reversal clashes with the sure-thing principle in the observer’s mind.

We argue that this often-overlooked disparity between average and aggregate performances, as possibly the real crux behind the Simpson’s paradox, is indeed sometimes overlooked in the context of NLP training, not only regarding its existence, but also regarding its impact to the training. Given presence of this disparity, a model that is better in terms of averaged per-sample performance could turn out to be worse in terms of the aggregate performance measured by applying the same evaluation metric to the whole data set directly. This reversal in ranking NLP models (or model parameters) can not only lead to biases in the gradient estimation for SGD (which is based on the average performance), causing inefficiency or failure to optimize the model towards better aggregate performance, but more severely, can cause the training to land in sub-optimal solutions (in terms of aggregate performance) even if an oracle optimization procedure is given (which can at its best maximize the average performance). As both the aforementioned issue in model training and the classic Simpson’s paradox in statistical sciences are fundamentally rooted from the disparity between two different ways to compute the same metric (averaged or aggregated), we call this disparity, the Simpson’s bias, so as to highlight the intrinsic connections between the two.

4 Experiments

This section experimentally studies (1) how significant the Simpson’s bias can be in standard NLP benchmarks and (2) how the bias affects the NLP training in those benchmarks. In the following, we report observations about these two questions in three common NLP tasks: Paraphrase Similarity Matching (PSM), Named Entity Recognition (NER) and Machine Translation (MT).

4.1 Experiment Design

The first question is relatively easy to address. Let $M$ be a NLP model trained for a task with training corpus $S$ and testing metric $F$, the significance of the Simpson’s bias of $F$ on model $M$ is denoted by

$$
\epsilon(M) = |F(S; M) - \tilde{F}(S; M)|
$$

where $\tilde{F}$ is the average-form metric corresponding to $F$. Note that model $M$ is not necessarily trained with $F$, but we can generally measure the Simpson’s bias between $F$ and $\tilde{F}$ on an arbitrary model. In our experiments, we will measure the bias $\epsilon$ in various tasks with various metrics $F$, and on models trained with various loss functions under various hyper-parameter and pre-processing settings.

The second question, i.e. to measure the impact of the Simpson’s bias, is more tricky. Ideally, one would want to directly compare the performances (in terms of $F$) between models trained with sample-level objective $F$ and those trained with corpus-level objective $\tilde{F}$. However, a key obstacle here is that we cannot easily compute/estimate the gradient of the corpus-level objective $\tilde{F}$ (over any corpus beyond modest size) to optimize it, which is exactly why people turned to the sample-level objective $F$ in the first place. In our experiments we instead observe the impact of Simpson’s bias to NLP training from three indirect perspectives.

First, we seek to observe how consistent $F$ and $\tilde{F}$ can be when used to compare a given pair of models. Such a model pair essentially serves as a highly degenerate model/parameter space (of size 2), over which we want to see if the optimum of $\tilde{F}$ is also the optimum of $F$. In this paper we focus on comparing pairs of models obtained from consecutive learning steps in a training process. For a learning step $t$, we measure the changing directions at $t$ by calculating the $\Delta F^t$ and $\Delta \tilde{F}^t$ according to:

$$
\begin{align*}
\Delta F^t &= F^t - F^{t-1} \\
\Delta \tilde{F}^t &= \tilde{F}^t - \tilde{F}^{t-1}
\end{align*}
$$

The sign of $\Delta F^t$ or $\Delta \tilde{F}^t$ represents the changing direction. $\Delta F^t \cdot \Delta \tilde{F}^t > 0$ indicates that $F$ and $\tilde{F}$ are consistent in evaluating the models at $t$ and $t-1$. $\Delta F^t \cdot \Delta \tilde{F}^t \leq 0$ suggests that $F$ and $\tilde{F}$ have changed in opposite directions in step $t$, indicating inconsistent model evaluation. We call
such an inconsistent \((\Delta F_1, \Delta \tilde{F}_1)\), a reversal pair. If reversal pairs are rare throughout the whole training process, we can say that the changes of \(F\) and \(\tilde{F}\) are highly consistent. In other words, we can maximize \(F\) by optimizing \(\tilde{F}\). Alternatively, if there are a large number of reversal pairs, we may at least need a longer time to reach the optimal \(F\). Moreover, a tremendous amount of inconsistent directions increase the risk that \(F\) can be significantly sub-optimal.

Our second experiment to observe the impact of Simpson’s bias is to compare models trained with \(F\) to those trained with the standard CE loss. In particular, some previous NLP works, such as Li et al. (2020), proposed to replace the CE loss with smoothed Dice loss for imbalanced data sets due to its similarity to the F1 metric. Instead of assigning if models thus trained are competitive to those trained directly with F1, we ask: How much can the models trained with Dice loss (at least) outperform those with CE loss? As our theoretical analysis (Theorem 3 in particular) has pointed out, optimizing smoothed average-form DSC is actually equivalent to optimize the accuracy. One may then expect comparable learning results between smoothed Dice loss and CE loss. If this were indeed the case, it would indirectly indicate that the models trained with Dice loss (corresponding to \(F\)) might be substantially sub-optimal in F1 (corresponding to \(F\)), assuming that the CE loss (which is not F1-oriented) cannot fully optimize F1 (which was the general premise to consider conjugated loss at all).

Our third experiment on the impact of Simpson’s bias is to examine the correlation between the bias and the training quality (in varying training settings). If high significance-of-bias is correlated with low training quality, it may potentially imply some deeper causal relationships between the two.

### 4.2 Dataset and Setting

For PSM, we use two standard data sets: Microsoft Research Paragraph Corpus (MRPC) (Dolan and Brockett 2005) and Quora Question Pairs (QQP) (Wang et al. 2018). We adopt the pre-trained BERT-base-uncased model with different training objectives (CE and Dice loss). The officially recommended parameter settings (Wolf et al. 2019) are leveraged, including max sequence length=128, epoch number=3, train batch size=32, learning rate=2e-5, and \(\gamma=1\).

For NER, we fine-tune BERT multilingual cased model with different loss function (CE / Dice) on GermEval 2014 dataset (Benikova, Biemann, and Reznicek 2014). For-
between $F_{\text{train}}$ and $F_{\text{Dice}}$, it seems $F_{\text{train}}$ converges early in Figure 1c, but it is not. Actually, in the whole training process, $F_{\text{train}}$ increases rapidly and then changes with small-scale.

At this time, $F_{\text{train}}$ increases slowly and finally converges to about 0.4. For MT task, Figure 1d shows the changes of the $F_{\text{BLEU}}$ and $F_{\text{BLEU}}$ scores over time during training. As they both increase, we can see clear disparity between them.

Through these observations, we find that (1) smooth strategy in these NLP tasks is of limited use for eliminating bias; (2) during the whole training process, the value of bias is significant and cannot be ignored.

### 4.4 Impact of Simpson’s Bias

**Consistency testing** This experiment seeks to observe how consistent $F$ and $F_{\text{train}}$ can be when used to compare a given pair of models. For PSM task, Figure 2a and 2b show a clear inconsistency between the changes in $F_{\text{DSC}}$ and $F_{\text{DSC}}$ on MRPC and QQP task. By tracking the tendency of the DSC value changes at the $F_{\text{DSC}}$ and $F_{\text{DSC}}$, we find out of the 115 training steps, 59 (or half of them) show an opposite trends between $\Delta F_{\text{DSC}}$ and $\Delta F_{\text{DSC}}$. 46 out of 100 sample dots pairs in Figure 2b have different change directions, the red dots indicate the disparity between $\Delta F_{\text{DSC}}$ and $\Delta F_{\text{DSC}}$. For NER task, there some extreme values in model early training, which reflect the fastest improvements. But the existence of these extreme values hinder our analysis, so it does not exist in Figure 2c. It can be seen from Figure 2c, in most cases, the change directions of $F_{\text{train}}$ and $F_{\text{train}}$ are completely inconsistent. For MT task, we plotted the scattered dots for each ($\Delta F_{\text{BLEU}}, \Delta F_{\text{BLEU}}$) pairs to see whether they both increase or decrease in the same direction. There are 77 / 195 sampled dots have different changing directions in total. There are a larger number of reversal pairs on these NLP tasks, $F$ may at least need a longer time to reach the optimal. Moreover, the high degree of inconsistency between $F$ and $F_{\text{train}}$ may increase the difficulty for $F$ optimization.

**Comparison with CE** This experiment is to observe the impact of Simpson’s bias by comparing models trained with $F$ to those trained with the standard CE loss. For PSM task, as show in Table 1, BERT trained with the CE loss (a.k.a. $F_{\text{MLE}}$) outperforms the model parameters trained with Dice loss (i.e., BERT + Dice) by a small margin: +0.78/0.45 in terms of F1 score on MRPC/QQP task. For NER task, as the Table 1 shows, the model trained with CE is about 3.53 point higher than that trained with Dice. All the result in Table 1 indicates the fact that the Dice did not achieve better performance may suggest that it does not necessarily drive the optimization toward high DSC scores, despite of their similarity. And using smoothing constants $\gamma \in [0, 1]$ does not work to eliminate Simpson’s bias on these tasks.

**Impacts on training quality** We conduct more experiments under different settings to get various $F$ variant on MRPC task. No matter how to modify the hyper-parameter, this bias between $F$ and $F_{\text{train}}$ is still significant, there are still a lot of reversed pairs and the performance of the model trained with $F$ is worse than that of CE. Meanwhile, we find a negative relation between the model quality on train dataset $F_{\text{Dice}_{\text{train}}}$ and the significance of bias $\epsilon$. Figure 3 is a scatter plot that shows the significance of bias and training quality. As can be seen from the figure, when $F_{\text{Dice}_{\text{train}}}$ tends to decrease as $\epsilon$ increases. These experiments results suggest that the Simpson’s bias is a common phenomenon in NLP training and not changing with model tuning. See more discussions in appendix of our full paper.

### 5 Conclusions

In this paper we coined a new concept, Simpson’s bias, for its similar role in inducing sub-optimal training in ML and in inducing the Simpson’s paradox in statistics. We presented a theoretical taxonomy for the Simpson’s bias in ML, revealing how similar effect is embodied in a wide spectrum of ML metrics, from ones as simple as Accuracy, to ones as sophisticated as BLEU. For some aggregate-form metrics, we show that it is possible to construct provably unbiased average-form surrogate through adding special and uncommon (e.g. negative) smoothing constants. But the Simpson’s bias is generally a factor with important impact in a variety of NLP tasks, as our experiments showed. We observed both noticeable margins of the bias and a significant number of “reversed” SGD steps in all the different tasks, data-sets, and metrics. Our experiments also show that models trained with “naively-conjugated” objectives (such as dice loss to F1) can be even worse than those trained with non-conjugated objectives (such as CE loss to F1), which could potentially reflect a significant sub-optimality when training using (seemingly-)conjugated objectives. Finally, a clear correlation between the Simpson’s bias and training quality is consistently observed. We believe these results indicate that the Simpson’s bias is a serious issue in NLP training, and probably in machine learning in general, that deserves more studies in the future.

### References


---

Table 1: Performance(F1 Score) of various training objective on dev set for MRPC/ QQP task, and test set for NER task.

<table>
<thead>
<tr>
<th>Loss</th>
<th>MRPC</th>
<th>QQP</th>
<th>NER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE Loss</td>
<td>89.78</td>
<td>87.84</td>
<td>86.14</td>
</tr>
<tr>
<td>Dice Loss</td>
<td>89.00</td>
<td>87.39</td>
<td>82.61</td>
</tr>
</tbody>
</table>

Figure 3: Significance of bias $\epsilon$ vs $F_{\text{Dice}_{\text{train}}}$. 


