On the Evaluation of Neural Selective Prediction Methods for Natural Language Processing

Zhengyao Gu
Reed College
Portland, OR

Mark Hopkins
Williams College
Williamstown, MA

Abstract

We provide a survey and empirical comparison of the state-of-the-art in neural selective classification for NLP tasks. We also provide a methodological blueprint, including a novel metric called *refinement* that provides a calibrated evaluation of confidence functions for selective prediction. Finally, we supply documented, open-source code to support the future development of selective prediction techniques.

1 Introduction

Deep learning has brought massive improvements to natural language processing over the past decade, but neural networks still make mistakes. Accordingly, there is a current interest in confidence estimation techniques that perform well on deep neural networks. A prominent subarea of confidence estimation is *selective prediction* (El-Yaniv et al., 2010; Geifman and El-Yaniv, 2017). Selective prediction focuses on developing classifiers that choose to abstain when sufficiently uncertain. There is less focus on absolute measures of confidence, and more on a classifier's ability to successfully rank its predictions, enabling techniques that maximize prediction quality given a desired yield (Geifman and El-Yaniv, 2019) or that maximize yield given a desired quality (Geifman and El-Yaniv, 2017).

This paper provides a survey and rigorous empirical comparison of the state-of-the-art in *neural selective classification* (i.e. selective prediction where the underlying classifier is a neural network) specifically as it pertains to natural language processing. Our main contributions: (a) we survey a variety of recent techniques proposed in the ML and NLP literature and compare them across **six** classification tasks from the GLUE benchmark (Wang et al., 2018), do careful hyperparameter tuning for all surveyed techniques, and perform multiple trials of each technique to get an adequate sense of median and variance; (b) inspired by (Xin et al., 2021) and Kendall-tau distance (Kendall, 1948), we propose a simple metric called *refinement* that provides a calibrated measure of the performance of selective classification techniques; (c) we determine that using maximum softmax probability as a confidence indicator remains a strong baseline, but Monte Carlo Dropout (Gal and Ghahramani, 2016) demonstrates significant improvement across multiple tasks and trials; (d) we release a documented Python package called spred (**s**elective **pred**iction) to make our experiments transparent and reproducible. To facilitate evaluation of future techniques, the package provides tutorials about how to add and evaluate novel selective prediction methods.

2 Selective Prediction

2.1 Preliminaries

A *prediction function* is a function $f : \mathcal{X} \to \mathcal{Y}$ that maps an instance space $\mathcal{X}$ to a label space $\mathcal{Y}$. We refer to the output $f(x)$ of the prediction function as its *prediction* for instance $x \in \mathcal{X}$. We use the notation $\checkmark(x)$ to refer to the gold prediction for a particular instance. The following denotes the correctly and incorrectly predicted instances of prediction function f on set $\mathbf{x} \subseteq \mathcal{X}$:

$$\begin{aligned} \mathcal{C}(f, \mathbf{x}) &= \{x \in \mathbf{x} \mid f(x) = \checkmark(x)\} \\ \overline{\mathcal{C}}(f, \mathbf{x}) &= \{x \in \mathbf{x} \mid f(x) \neq \checkmark(x)\} \end{aligned}$$

If we pair a prediction function with a *selection function* $\tilde{g} : \mathcal{X} \to \{0, 1\}$, we obtain a *selective model* $(f, \tilde{g})$. For instance $x \in \mathcal{X}$, a selective model $h = (f, \tilde{g})$ *publishes* its prediction $f(x)$ if $\tilde{g}(x) = 1$, and *abstains* if $\tilde{g}(x) = 0$. In short:

$$h(x) = \begin{cases} f(x) & \text{if } \tilde{g}(x) = 1 \\ \perp & \text{if } \tilde{g}(x) = 0 \end{cases}$$

$$Y = \begin{cases} Y^{(1)} & \text{if } R_Y = 1 \\ ? & \text{if } R_Y = 0 \end{cases}$$

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7888–7899

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
$f^{(A)}(x) = \checkmark(x)?$	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓
$g_1^{(A)}(x)$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$g_2^{(A)}(x)$	0.5	0.1	0.6	0.2	0.3	0.4	0.7	0.8	0.9	1.0
$g_3^{(A)}(x)$	0.1	0.7	0.2	0.8	0.9	1.0	0.3	0.4	0.5	0.6

Figure 1: Three confidence functions for an example prediction function that has an overall accuracy of 6/10 on the evaluation set.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
$f^{(B)}(x) = \checkmark(x)?$	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
$g_1^{(B)}(x)$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$g_2^{(B)}(x)$	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.1	0.9	1.0
$g_3^{(B)}(x)$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	1.0	0.8	0.9

Figure 2: Three confidence functions for a stronger prediction function that has an overall accuracy of 9/10 on the evaluation set.

where $\perp$ is a symbol representing abstention. A convenient way to implement a selection function is to use a *confidence function* $g : \mathcal{X} \to \mathbb{R}$ that assigns a real-valued confidence to any instance $x \in \mathcal{X}$. We can derive a selection function $\tilde{g}_\theta$ from confidence function g by specifying a minimum confidence threshold $\theta \in \mathbb{R}$ for publishing predictions:

$$\tilde{g}_\theta(x) = \mathbb{1}[g(x) > \theta]$$

2.2 Examples

In Figure 1, we show three confidence functions $g_1^{(A)}$, $g_2^{(A)}$, $g_3^{(A)}$ for an example prediction function $f^{(A)}$. The first confidence function $g_1^{(A)}$ is pretty good; it assigns its highest confidences to four out of the six correct predictions, though unfortunately it also gives its lowest confidence to the correct prediction $f^{(A)}(x_1)$. The superior $g_2^{(A)}$ is a best-case confidence function (assigning its highest confidences to the six correct predictions) and $g_3^{(A)}$ is a worst-case confidence function (assigning its lowest confidences to the six correct predictions).

Figure 2 shows three more confidence functions $g_1^{(B)}$, $g_2^{(B)}$, $g_3^{(B)}$ for a stronger prediction function $f^{(B)}$. This time, the first confidence function $g_1^{(B)}$ is not particularly good; it assigns its third-highest confidence to the only incorrect prediction. Again, $g_2^{(B)}$ is a best-case confidence function (assigning its highest confidences to the nine correct predictions) and $g_3^{(B)}$ is a worst-case confidence function (assigning its lowest confidences to the nine correct predictions).

2.3 Evaluation with AUC Metrics

Typically, one evaluates the goodness of a confidence function by quantifying the trade-off between the quality and quantity of its published predictions. Since the prominent approaches – risk/coverage curves (El-Yaniv et al., 2010), receiver-operator (ROC) curves (Davis and Goadrich, 2006), and precision-recall curves (Hendrycks and Gimpel, 2017) – share many of the same benefits and drawbacks (some of which we discuss in this section), we will focus on precision-recall curves, mainly due to the NLP community's increased familiarity with them. In Figure 3, we show the precision-recall curves for the six confidence functions from the previous subsection. The aspiration of any confidence function is to achieve an **A**rea **U**nder the **P**recision-**R**ecall curve (AUPR) of 1, which means that it has perfectly separated the correct and incorrect predictions of the prediction function. Among the examples, this has been achieved by confidence functions $g_2^{(A)}$ and $g_2^{(B)}$.

A drawback with AUPR (and its analogs) is that its value is not interpretable without knowledge of the goodness of the associated prediction function. To see why, suppose we replace confidence function $g_1^{(A)}$ with a random confidence function g_{rnd}, i.e. we choose each confidence $g_{\mathsf{rnd}}(x)$ uniformly at random from the interval $(0, 1)$. Empirically, we observe an average AUPR[1] of approximately 0.64 for a random confidence function g_{rnd} associated with prediction function $f^{(A)}$. Since this is considerably worse than the AUPR (0.863) of confidence function $g_1^{(A)}$, we can surmise that confidence function $g_1^{(A)}$ is "better than random". By contrast, a random confidence function associated with the superior prediction function $f^{(B)}$ yields a empirical average AUPR[2] of approximately 0.91, which is considerably **better** than the AUPR (0.865) of confidence function $g_1^{(B)}$ – thus confidence function $g_1^{(B)}$ is "worse than random".

Both $g_1^{(A)}$ and $g_1^{(B)}$ have AUPRs of approximately 0.86, but the former is "better than random," while the latter is "worse than random."

[1] Averaged over 200 random confidence functions.

[2] Again, averaged over 200 random confidence functions.

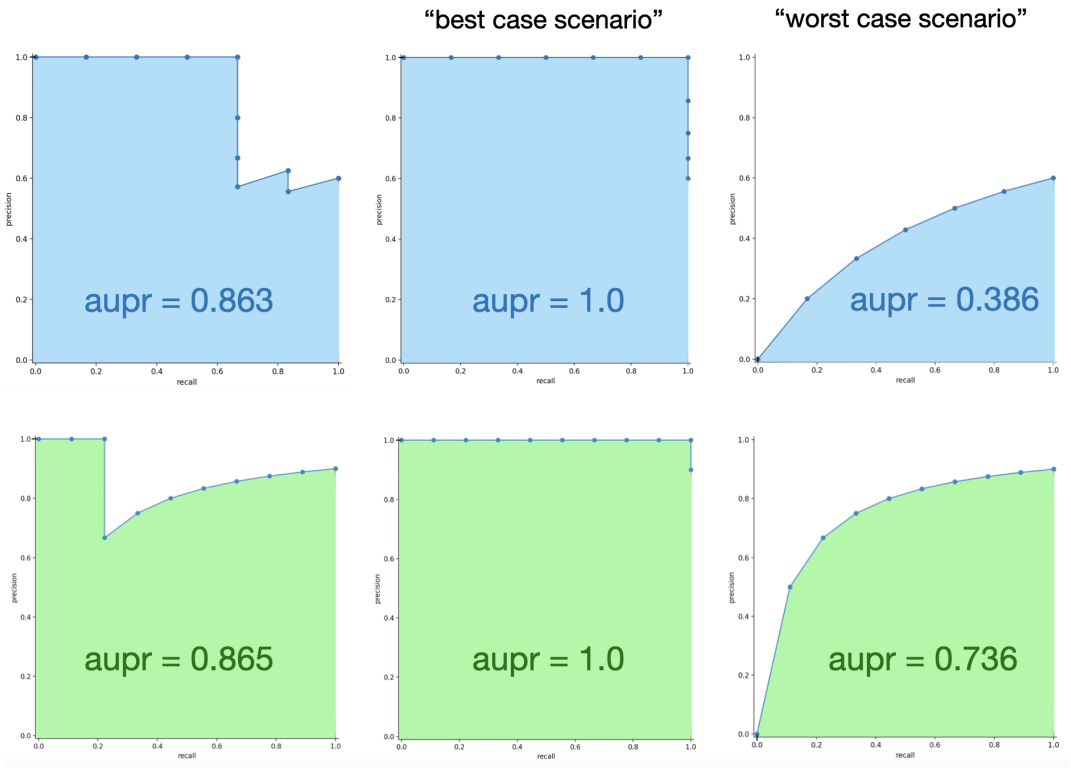


Figure 3: Precision-recall curves for confidence functions $g_1^{(A)}$, $g_2^{(A)}$, $g_3^{(A)}$ (top, blue) and confidence functions $g_1^{(B)}$, $g_2^{(B)}$, $g_3^{(B)}$ (bottom, green).

Thus, AUPR does not provide a standalone indicator of the quality of a confidence function. This is because AUPR is a composite of the goodness of the confidence function and the goodness of the prediction function. One could imagine calibrating AUPR by taking into account the worst-case AUPR (i.e. the AUPRs for worst-case confidence functions $g_3^{(A)}$ and $g_3^{(B)}$) but we will adopt an even simpler approach based on a recent proposal by (Xin et al., 2021).

2.4 Evaluation with Kendall-tau Distance

If we sort predictions by increasing confidence (as in Figure 4), a best-case confidence function (e.g. $g_2^{(A)}$) ranks all incorrect predictions below all correct predictions, while a worst-case confidence function (e.g. $g_3^{(A)}$) ranks all *correct* predictions below all *incorrect* predictions. Observing this, (Xin et al., 2021) proposed a rank-based evaluation metric for selective prediction called **R**eversed **P**air **P**roportion (RPP), which is a normalized count of pairwise ranking errors, i.e. a normalized version of Kendall-tau distance (Kendall, 1948):

$$\tau_{f,\mathbf{x}}(g) = \sum_{\substack{x_{\checkmark} \in \mathcal{C}(f,\mathbf{x}) \\ x_{\times} \in \overline{\mathcal{C}}(f,\mathbf{x})}} \mathbb{1}[g(x_{\checkmark}) < g(x_{\times})]$$

$$\text{RPP}_{f,\mathbf{x}}(g) = \frac{\tau_{f,\mathbf{x}}(g)}{|\mathbf{x}|^2}$$

For instance, confidence function $g_1^{(A)}$ has a Kendall-tau distance of 7 (it misranks correct instance x_1 below 4 incorrect predictions, and instance x_3 below 3 incorrect predictions) and an RPP of $\frac{7}{100}$. For the best-case confidence function $g_2^{(A)}$, $\tau_{f,\mathbf{x}}(g_2^{(A)}) = \text{RPP}_{f,\mathbf{x}}(g_2^{(A)}) = 0$. Unfortunately, using $|\mathbf{x}|^2$ as the normalizer means that RPP suffers the same issue as AUPR: its value cannot be interpreted independently of the goodness of the prediction function. Consider the RPP for our "worse-than-random" confidence function $g_1^{(B)}$. Like $g_1^{(A)}$, it has a Kendall-tau distance of 7 (it misranks incorrect instance x_8 above 7 correct predictions) and thus an RPP of $\frac{7}{100}$. Even though $g_1^{(A)}$ is better than random and $g_1^{(B)}$ is worse than random, they end up with the same RPP.

Fortunately, there is a simple remedy: all we need to do is normalize by the worst-case Kendall-tau distance $c(|\mathbf{x}| - c)$, where c is the number of

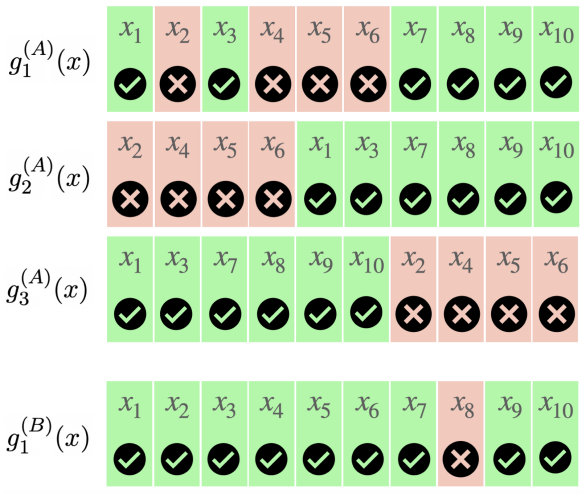


Figure 4: The predictions of four of our example confidence functions, sorted by increasing confidence. The RPP for $g_1^{(A)}$ and $g_1^{(B)}$ are equivalent, although the former is better than random and the latter is worse than random.

correct predictions made by the prediction function. To obtain a metric with a similar interpretation to accuracy and AUPR (for which higher values are "better"), we subtract this new normalized distance from one, resulting in a measurement we call *refinement*:

$$\mathcal{R}_{f,\mathbf{x}}(g) = 1 - \frac{\tau_{f,\mathbf{x}}(g)}{c(|\mathbf{x}| - c)}$$

where $c = |\mathcal{C}(f, \mathbf{x})|$. Refinement has the following appealing properties:

Theorem 1. *If $0 < |\mathcal{C}(f, \mathbf{x})| < |\mathbf{x}|$ (i.e. prediction function f makes at least one correct prediction and at least one incorrect prediction), then:*

$$\min_g \mathcal{R}_{f,\mathbf{x}}(g) = 0$$
$$\max_g \mathcal{R}_{f,\mathbf{x}}(g) = 1$$

Proof. Consider prediction function f and instance set $\mathbf{x}$ such that $0 < |\mathcal{C}(f, \mathbf{x})| < |\mathbf{x}|$, which implies that $c(|\mathbf{x}|-c) > 0$. Since $0 \leq \tau_{f,\mathbf{x}}(g) \leq c(|\mathbf{x}|-c)$, therefore $0 \leq \mathcal{R}_{f,\mathbf{x}}(g) \leq 1$. To prove the theorem, then, we need only show these bounds are tight, i.e. construct two confidence functions $g_{\mathsf{min}}(x)$ and $g_{\mathsf{max}}(x)$ such that $g_{\mathsf{min}}(x) = 0$ and $g_{\mathsf{max}}(x) = 1$. Let:

$$g_{\mathsf{min}}(x) = \begin{cases} 0.4 & \text{if } x \in \mathcal{C}(f, \mathbf{x}) \\ 0.6 & \text{if } x \in \bar{\mathcal{C}}(f, \mathbf{x}) \end{cases}$$

$$g_{\mathsf{max}}(x) = \begin{cases} 0.4 & \text{if } x \in \bar{\mathcal{C}}(f, \mathbf{x}) \\ 0.6 & \text{if } x \in \mathcal{C}(f, \mathbf{x}) \end{cases}$$

Then $\mathcal{R}_{f,\mathbf{x}}(g_{\mathsf{min}}) = 1 - \frac{(|\mathbf{x}|-c)}{c(|\mathbf{x}|-c)} = 0$, and $\mathcal{R}_{f,\mathbf{x}}(g_{\mathsf{max}}) = 1 - \frac{0}{c(|\mathbf{x}|-c)} = 1$. □

In other words, a best-case confidence function has a refinement of 1, whereas a worst-case confidence function has a refinement of 0. Moreover:

Theorem 2. *For a random confidence function g, the expected value of $\mathcal{R}_{f,\mathbf{x}}(g)$ is 0.5.*

Proof. Define binary random variable Z_{ij} such that $Z_{ij} = 1$ iff $g(x_i) < g(x_j)$. According to the definition of Kendall-tau distance:

$$\tau_{f,\mathbf{x}}(g) = \sum_{\substack{x_i \in \mathcal{C}(f,\mathbf{x}) \\ x_j \in \bar{\mathcal{C}}(f,\mathbf{x})}} Z_{ij}$$

By linearity of expectation:

$$\begin{aligned} E(\tau_{f,\mathbf{x}}(g)) &= E\left(\sum_{\substack{x_i \in \mathcal{C}(f,\mathbf{x}) \\ x_j \in \bar{\mathcal{C}}(f,\mathbf{x})}} Z_{ij} \right) \\ &= \sum_{\substack{x_i \in \mathcal{C}(f,\mathbf{x}) \\ x_j \in \bar{\mathcal{C}}(f,\mathbf{x})}} E(Z_{ij}) \\ &= \sum_{\substack{x_i \in \mathcal{C}(f,\mathbf{x}) \\ x_j \in \bar{\mathcal{C}}(f,\mathbf{x})}} 0.5 \\ &= 0.5c(|\mathbf{x}| - c) \end{aligned}$$

And therefore:

$$\begin{aligned} E(\mathcal{R}_{f,\mathbf{x}}(g)) &= 1 - \frac{E(\tau_{f,\mathbf{x}}(g))}{c(|\mathbf{x}| - c)} \\ &= 1 - \frac{0.5c(|\mathbf{x}| - c)}{c(|\mathbf{x}| - c)} \\ &= 0.5 \end{aligned}$$

□

Unlike RPP and the various area under the curve metrics, refinement directly assesses the quality of the confidence function, and its value is calibrated and interpretable (0 = worst case, 0.5 = random confidence, 1.0 = best case) without knowledge of the quality of the associated prediction function.

3 Surveyed Techniques

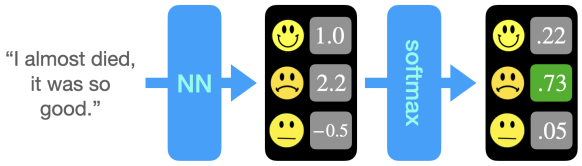
Our main goal in this paper is a reproducible and thorough comparison of a broad range of selective prediction techniques on NLP tasks. In this section, we describe the techniques we compare.

3.1 Confidence Functions

The following are ways to create a confidence function for an already trained neural prediction function.

MAXPROB

For neural prediction functions, the simplest-to-implement confidence function is likely MAXPROB, shown here applied to a three-way sentiment analysis task:



After applying softmax to the neural network output, MAXPROB (sometimes known as SOFTMAXRESPONSE) uses the maximum probability of the resulting distribution as its measure of confidence. The surprising effectiveness of such a simple approach was observed by (Hendrycks and Gimpel, 2017), among others.

Monte Carlo Dropout (MCDM/MCDV)

(Gal and Ghahramani, 2016) proposed leveraging dropout (Srivastava et al., 2014) to assess the uncertainty of a neural network on a particular instance. As usual, dropout is disabled at test time to make the prediction. But then the input instance is re-decoded k times with dropout enabled. This yields k samples for the softmax probability of the prediction. There are two common methods (Kamath et al., 2020) for synthesizing these k samples into a confidence measure: either we take the mean (Lakshminarayanan et al., 2017) of the samples (a strategy we refer to as MCDM), or the negative[3] variance (Feinman et al., 2017; Smith and Gal, 2018) of the samples (a strategy we refer to as MCDV).

TRUSTSCORE

(Jiang et al., 2018) advocated a nearest-neighbor-based confidence function. First, the training instances are converted[4] into vector encodings, and grouped according to their gold labels. Outliers are then filtered from each labeled group. Specifically, they sort the vectors (i.e. points in $\mathbb{R}^d$ space) by the radius of the minimal ball centered at that vector that contains k points from their labeled group. The percentage $\alpha \in [0, 1]$ of points with the largest such radii (i.e. the outliers) are removed. This filtered set[5] is called an *α-high density set*. The confidence assigned to an instance prediction, called TRUSTSCORE, is the ratio of (a) the distance between the instance's vector encoding and the closest α-high density set of a *non-predicted* label, (b) the distance between the instance's vector encoding and the α-high density set of the *predicted* label.

3.2 Specialized Loss Functions

We also survey techniques that simultaneously train a prediction function and an associated confidence function.

Error Regularization (EREG)

(Xin et al., 2021) suggests adding an "error regularization" term to the task's loss function $\mathcal{L}$ that directly penalizes ranking errors made by the confidence function g:

$$\epsilon(f, \mathbf{x}) = \sum_{\substack{x_{\checkmark} \in C(f,\mathbf{x}) \\ x_{\times} \in \overline{C}(f,\mathbf{x})}} \left[\max(0, g(x_{\times}) - g(x_{\checkmark}))\right]^2$$

$$\mathcal{L}_{\text{EREG}}(f, \mathbf{x}) = \mathcal{L}(f, \mathbf{x}) + \lambda \cdot \sum_{\mathbf{b} \in \text{batches}(\mathbf{x})} \epsilon(f, \mathbf{b})$$

where $\lambda \in \mathbb{R}^+$ is a tunable hyperparameter and $\text{batches}(\mathbf{x})$ is the set of minibatches of training set $\mathbf{x}$. At training time, (Xin et al., 2021) uses MAXPROB for the confidence function g, though at test time, they additionally experiment with MCDM and MCDV.

Deep Abstaining Classifiers (DAC)

A Deep Abstaining Classifier (Thulasidasan et al., 2019), abbreviated DAC, explicitly introduces an extra abstention output $\perp$ to the neural network, and trains with a loss function that allows the prediction function to benefit from abstaining on difficult instances:

$$\mathcal{L}_{\text{DAC}}(f, \mathbf{x}) = (1 - p_{\perp})\mathcal{L}(f, \mathbf{x}) + \alpha \log \frac{1}{1 - p_{\perp}}$$

where $p_{\perp}$ is the probability according to abstention output $\perp$ after applying softmax, $\mathcal{L}(f, \mathbf{x})$ is standard cross-entropy loss over the non-abstention outputs, and α is a real-valued weight that is zero

[3] We use the *negative* variance so that a greater value indicates a greater confidence.

[4] They are agnostic about how best to do so. We will return to this issue.

[5] They fix $k = 10$, but treat α as a tunable hyperparameter.

for the first k (warmup) epochs of training, and is linearly scaled from α_{min} to α_{max} during the remaining epochs. The initial value α_{min} is set to be a fixed fraction $\frac{1}{\rho}$ of a moving average of the loss during the warmup epochs. The authors provide code that we use in our experiments. At test time, MAXPROB is used[6] as the confidence function, though with a slight modification – if the probability associated with the abstention label is the maximum softmax probability, then the next highest probability is used as the confidence.

4 Experiment Design

To draw reliable conclusions on a sufficiently varied set of NLP tasks, we evaluated the techniques on six classification[7] tasks of the GLUE benchmark (Wang et al., 2018): COLA, MNLI, MRPC, QNLI, RTE, and SST-2. Bearing in mind that our goal is to compare selective classification techniques, not to produce state-of-the-art prediction functions, we randomly partitioned each training set into two halves, using GLUETRAIN-A for training and GLUETRAIN-B for early stopping and hyperparameter tuning. Since the gold labels for GLUE test sets are not all publicly available, we reserved the development set (GLUEDEV) of each task for final evaluation. We trained the prediction function by fine-tuning BERT-BASE-CASED using the `transformers` package (Wolf et al., 2020), mostly using the training parameters recommended by its `run_glue.py` script (the sole deviation is that we run each training for 6 epochs, rather than 3). For the techniques that required specialized loss functions, we substituted the default BERT loss function with the alternative specified by the selective prediction technique.

4.1 Hyperparameter Tuning

In an effort to fairly evaluate each technique, we began with smaller-scale experiments to determine an appropriate setting of a technique's hyperparameters for the GLUE tasks. For these experiments, we used GLUETRAIN-A for training and GLUETRAIN-B for validation. We selected three GLUE tasks of various sizes and genres (one single-sentence task, one similarity-and-paraphrase task, and one inference task) as proxies: SST-2, MRPC, and RTE. We ran 5 trials for each hyperparameter setting.

MCDM/MCDV

The Monte Carlo Dropout techniques each have a single hyperparameter k: the number of decodings of the training instance with dropout enabled. We experimented with $k \in \{10, 30, 50\}$. We found little discernible difference between $k = 30$ and $k = 50$. Slightly better results with $k = 30$ versus $k = 10$ convinced us to use $k = 30$ for further experiments.

TRUSTSCORE

To use TRUSTSCORE, we need to encode each instance as a vector. Following common practice, we used BERT's final layer encoding (after fine-tuning) of the [CLS] token. To select the hyperparameter settings for TRUSTSCORE, we followed (Jiang et al., 2018) and experimented with several powers of two for hyperparameter α, specifically $\alpha \in \{0.5, 0.25, 0.125\}$. Also, since TRUSTSCORE is too slow in practice to run on large training sets, we sample N training instances (without replacement) prior to running the TrustScore algorithm. In our tuning experiments, we tried the values $N \in \{800, 1600\}$. We found little difference between the six hyperparameter settings and set $N = 800$ and $\alpha = 0.25$ for further experiments.

EREG

EREG has hyperparameter λ (the multiplier for the regularization term). Following the appendix of (Xin et al., 2021), we experimented with $\lambda \in \{0.01, 0.05, 0.1, 0.5\}$. Because EREG uses an alternative loss function that can potentially affect the overall quality of the prediction function, we used AUPR (which blends the quality of the prediction function with the quality of the selection function) as our main evaluation metric. We found high variance between trials, and selected $\lambda = 0.05$ (with the most consistent performance) for further experiments. EREG is also affected by the minibatch size. If the minibatch size is 1, then the loss function reduces to the base loss function for the task. Larger minibatches increase the number of pairwise comparisons incorporated into the regularization term. For the final experiments (Section 5),

[6] We also experimented with using $1 - p_{\perp}$ (i.e. the total probability mass accorded to non-abstention outputs) as the confidence, but this yielded poor results.

[7] We did not include WNLI because the training set was too small to train a prediction function that does better than random guessing. We did not include QQP because we had training difficulties that we could not resolve before the submission deadline. STS-B is a regression task, not a classification task. For evaluating MNLI, we used matched accuracy, since the focus of this paper is not on domain shift.

we used[8] minibatch sizes of 8 and 16.

DAC

One of the virtues of DAC is that it automatically adjusts its weights according to the cross-entropy loss observed during the warmup epochs, but it still has hyperparameters ρ and α_{max} to determine precisely how this is done. In the code accompanying (Thulasidasan et al., 2019), the default settings are $\rho = 64$ and $\alpha_{max} = 1.0$. Given these defaults, we experimented with $\rho \in \{32, 64, 128\}$ and $\alpha_{max} \in \{0.5, 1.0, 2.0\}$. The technique did not appear to be particularly sensitive to the choice of hyperparameters and so we kept the default settings for further experiments. We used two warmup epochs (sufficient to reach decent baseline accuracy for all GLUE tasks), and accordingly increased the total number of training epochs from 6 to 8.

5 Results

For final evaluation, we ran ten experiment trials on the six GLUE tasks. Specifically, we trained ten prediction functions with different random seeds for each loss function: the basic BERT loss (`basic`), BERT loss with error regularization (`ereg.b`, where `b` $\in \{8, 16\}$ is the minibatch size), and the Deep Abstaining Classifier loss (`dac`). For each resulting prediction function, we evaluated the various confidence functions. In all experiment trials, we used the hyperparameter settings established in Section 4. Figure 5 visualizes the results[9] for two GLUE datasets (MRPC and SST-2) using a violin plot[10]. Each "string" of the violin corresponds to the refinement of a single trial on GLUEDEV, while the "body" of the violin is a kernel density estimation of the result distribution. These results indicate that all techniques have considerable variation from trial to trial, and that each outperforms a random confidence baseline (which has an expected refinement of 0.5, according to Theorem 2). Beyond this, it is difficult to eyeball the results and make an informed decision about which technique to use. One can possibly eliminate TRUSTSCORE based on Figure 5, but what should we make of the advantages that MCDM and MCDV seem to offer over the basic MAXPROB approach? The MC Dropout techniques are considerably more expensive to run (since they require multiple independent decodings). Are they meaningfully better than MAXPROB?

We quantify the phrase "meaningfully better" by estimating the likelihood that a candidate technique outperforms the basic MAXPROB baseline. For a candidate technique t and evaluation metric m, define random variable $X_{t,m}$ as the result of the following trial:

- Choose a random task from a probability distribution P_{task} over tasks.
- Execute the candidate selective prediction technique t and the baseline technique (i.e. MAXPROB) and evaluate each using metric m (e.g. refinement or AUPR).
- If the candidate technique t outperforms the baseline according to metric m, return 1. Otherwise, return 0.

The expected value $E(X_{t,m})$ tells us the likelihood that technique t will outperform the baseline according to metric m . Since we performed 10 trials for each of 6 GLUE tasks, we therefore have 60 samples[11] for estimating $E(X_{t,m})$. Figure 6 shows the $E(X_{t,\text{refinement}})$ estimate for eight techniques (including the random confidence baseline), along with a 95% confidence interval. Somewhere between 62% to 85% of the time (with 95% confidence), both MCDM and MCDV improve upon the MAXPROB baseline according to refinement.

On its own, refinement is sufficient to evaluate the techniques that only modify the confidence function (i.e. MCDM, MCDV, and TRUSTSCORE). Techniques that employ specialized loss functions, (e.g. error-regularization or DAC loss) should be evaluated using both refinement and AUPR, since these techniques might improve the efficiency of the confidence function while simultaneously sacrificing the quality of the prediction function. Simultaneous evaluation with AUPR and refinement allows us to distinguish whether improvements are achieved via enhancements to the confidence or the

[8] Unfortunately, many GPU machines (including the machine running our experiments) do not have sufficient memory to finetune PTLMs with larger batch sizes.

[9] For visual clarity, we omit certain loss/confidence pairs from Figure 5, for instance `ereg(mcdm)` and `ereg(mcdv)`. In our experiments, the MC Dropout techniques provided similar improvement for all loss functions.

[10] We used the `seaborn` package to create the plots: https://seaborn.pydata.org/generated/seaborn.violinplot.html.

[11] In this case, the task distribution P_{task} is a uniform distribution over 6 GLUE tasks. Whether this is an effective proxy for NLP tasks in general is a fair question, but the community appears to have adopted GLUE as a useful benchmark.

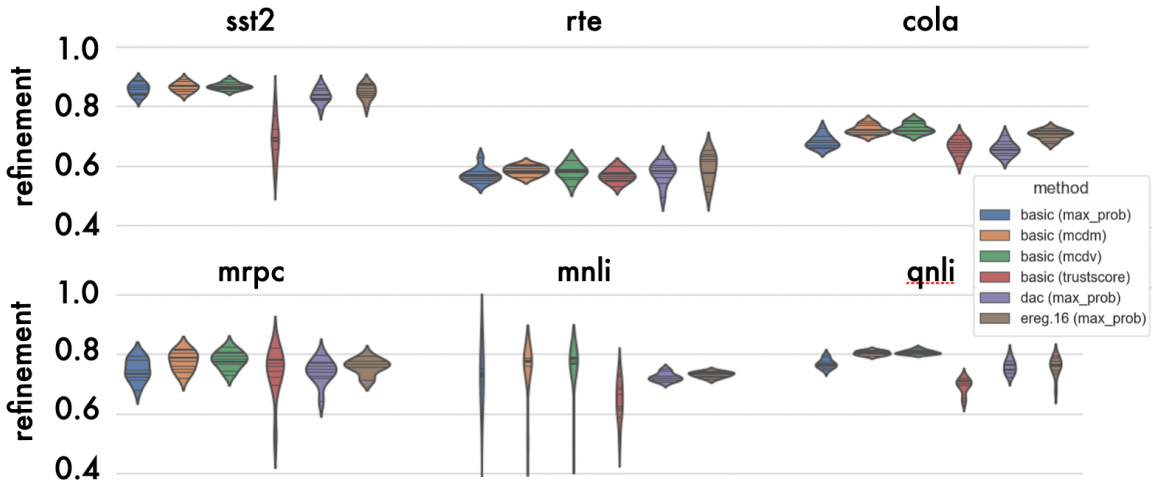


Figure 5: Results of several selective prediction techniques on six GLUE datasets. Each "string" of the violin corresponds to the refinement of a single trial on GLUEDEV, while the "body" of the violin is a kernel density estimation of the result distribution.

prediction function. In these experiments, AUPR and refinement largely agree about the quality of the loss specialization techniques, thus the techniques do not appear to impact the quality of the prediction function. In terms of improving the confidence function, EREG shows promise, but the trajectory of performance from minibatch size 8 to minibatch size 16 suggests that EREG needs a minibatch size of at least 32 to be a superior alternative to MAXPROB.

6 Related Work

Selective prediction has a long tradition in machine learning, dating back to the 1950s (Chow, 1957). There is an extensive literature (Hellman, 1970; Fumera and Roli, 2002; Cortes et al., 2016) on training classifiers with the ability to abstain (also known as the "reject option"), usually specific to alternative classifiers like support vector machines.

There is also a significant literature (Platt et al., 1999; Guo et al., 2017; Kumar et al., 2018; Wang et al., 2020; Desai and Durrett, 2020) on the topic of *calibration*, i.e. the development of probabilistically interpretable confidence measures. In this paper, we restrict our focus to the relative rankings of selective predictors, and not the confidence values themselves.

While our survey focuses on techniques designed to identify ambiguous instances in the evaluation set (and, for certain techniques, to also ignore label noise in the training set), there is also interest in selective prediction techniques that operate successfully under domain shift (Kamath et al., 2020; Liu et al., 2020; Wang et al., 2022), i.e. when the distribution of evaluation instances differs from the training instances. Evaluation of such techniques is beyond the scope of the work described here, but we have plans to expand the spred package to evaluate selective prediction under domain shift.

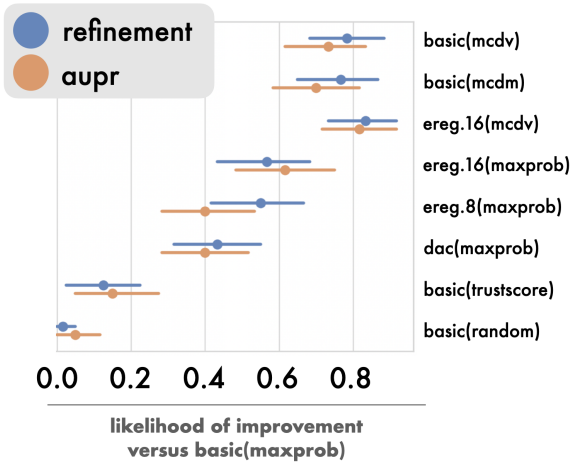


Figure 6: Estimate of the likelihood $E(X_{t,m})$ that technique t outperforms the basic MAXPROB baseline according to metric m (refinement or AUPR). The bars show a 95% confidence interval for this estimate.

7 Conclusion

We have provided a survey and empirical comparison of a diverse set of recent selective prediction techniques on a broad set of tasks. As a companion to the paper, the open-source Python package spred provides reproducible results and transparent methodology. Moreover, it comes with tutorials

and unit tests that demonstrate how new techniques and tasks can be easily added. We hope this will facilitate novel selective prediction research on natural language domains.

8 Limitations

While we have endeavored to include a good cross-section of selective prediction techniques in our empirical study, clearly it is not comprehensive of all work in this space. Moreover, this work does not address confidence calibration, nor does it address the behavior of selective predictive techniques under domain shift. Finally, our focus is on selective classification – we do not address confidence estimation for regression or generation tasks.

References

Chi-Keung Chow. 1957. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, (4):247–254.

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. 2016. Learning with rejection. In *International Conference on Algorithmic Learning Theory*, pages 67–82. Springer.

Jesse Davis and Mark Goadrich. 2006. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 233–240, New York, NY, USA. Association for Computing Machinery.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics.

Ran El-Yaniv et al. 2010. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(5).

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*.

Giorgio Fumera and Fabio Roli. 2002. Support vector machines with embedded reject option. In *International Workshop on Support Vector Machines*, pages 68–82. Springer.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

Yonatan Geifman and Ran El-Yaniv. 2017. Selective classification for deep neural networks. *Advances in Neural Information Processing Systems*, 30:4878–4887.

Yonatan Geifman and Ran El-Yaniv. 2019. Selectivenet: A deep neural network with an integrated reject option. In *International Conference on Machine Learning*, pages 2151–2159. PMLR.

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.

Martin E Hellman. 1970. The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Heinrich Jiang, Been Kim, Melody Y Guan, and Maya R Gupta. 2018. To trust or not to trust a classifier. In *NeurIPS*.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5684–5696, Online. Association for Computational Linguistics.

Maurice George Kendall. 1948. Rank correlation methods.

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.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30.

Weitang Liu, Xiaoyun Wang, John D Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *arXiv preprint arXiv:2010.03759*.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Lewis Smith and Yarin Gal. 2018. Understanding measures of uncertainty for adversarial example detection. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 560–569. AUAI Press.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014.

Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Sunil Thulasidasan, Tanmoy Bhattacharya, Jeff Bilmes, Gopinath Chennupati, and Jamal Mohd-Yusof. 2019. Combating label noise in deep learning using abstention. In *International Conference on Machine Learning*, pages 6234–6243. PMLR.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. 2022. Vim: Out-of-distribution with virtual-logit matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4921–4930.

Shuo Wang, Zhaopeng Tu, Shuming Shi, and Yang Liu. 2020. On the inference calibration of neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3070–3079, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. The art of abstention: Selective prediction and error regularization for natural language processing. 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)*, pages 1040–1051, Online. Association for Computational Linguistics.

ACL 2023 Responsible NLP Checklist

A For every submission:

☐ A1. Did you describe the limitations of your work?
Left blank.

☐ A2. Did you discuss any potential risks of your work?
Left blank.

☐ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☐ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☐ Did you run computational experiments?

Left blank.

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☐ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

☐ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D ☐ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.