The Art of Abstention: Selective Prediction and Error Regularization for Natural Language Processing

Ji Xin, Raphael Tang, Yaoliang Yu, and **Jimmy Lin**

David R. Cheriton School of Computer Science, University of Waterloo
Vector Institute for Artificial Intelligence

{ji.xin,r33tang,yaoliang.yu,jimmylin}@uwaterloo.ca

Abstract

In selective prediction, a classifier is allowed to *abstain* from making predictions on low-confidence examples. Though this setting is interesting and important, selective prediction has rarely been examined in natural language processing (NLP) tasks. To fill this void in the literature, we study in this paper selective prediction for NLP, comparing different models and confidence estimators. We further propose a simple error regularization trick that improves confidence estimation without substantially increasing the computation budget. We show that recent pre-trained transformer models simultaneously improve both model accuracy and confidence estimation effectiveness. We also find that our proposed regularization improves confidence estimation and can be applied to other relevant scenarios, such as using classifier cascades for accuracy–efficiency trade-offs. Source code for this paper can be found at https://github.com/castorini/transformers-selective.

1 Introduction

Recent advances in deep learning models have pushed the frontier of natural language processing (NLP). Pre-trained language models based on the transformer architecture (Vaswani et al., 2017) have improved the state-of-the-art results on many NLP applications. Naturally, these models are deployed in various real-world applications. However, one may wonder whether they are always reliable, as pointed out by Guo et al. (2017) that modern neural networks, while having better accuracy, tend to be overconfident compared to simple networks from 20 years ago.

In this paper, we study the problem of selective prediction (Geifman and El-Yaniv, 2017) in NLP. Under the setting of selective prediction, a model is allowed to *abstain* from making predictions on uncertain examples (Figure 1) and thereby reduce the error rate. This is a practical setting in a lot of realistic scenarios, such as making entailment judgments for breaking news articles in search engines (Carlebach et al., 2020) and making critical predictions in medical and legal documents (Zhang et al., 2019). In these cases, it is totally acceptable, if not desirable, for the models to admit their uncertainty and call for help from humans or better (but more costly) models.

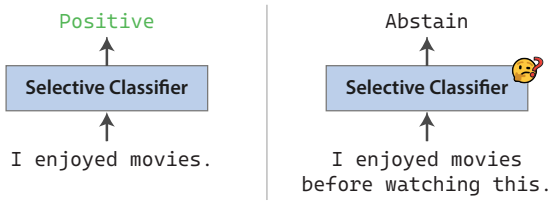

Figure 1: Example of a selective classifier that makes a prediction for a confident example (left) and abstains for an uncertain one (right).

Under the selective prediction setting, we construct a selective classifier by pairing a standard classifier with a confidence estimator. The confidence estimator measures how confident the model is for a certain example, and instructs the classifier to abstain on uncertain ones. Naturally, a good confidence estimator should have higher confidence for correctly classified examples than incorrect ones. We consider two choices of confidence estimators, softmax response (SR; Hendrycks and Gimpel, 2017), and Monte-Carlo dropout (MC-dropout; Gal and Ghahramani, 2016). SR interprets the output of the final softmax layer as a probability distribution and the highest probability as confidence. MC-dropout repeats the inference process multiple times, each time with a different dropout mask, and treats the negative variance of maximum probability as confidence. Confidence estimation is critical to selective prediction, and therefore studying this problem also helps relevant tasks such as active

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 1040–1051
August 1–6, 2021.

learning (Cohn et al., 1995; Shen et al., 2018) and early exiting (Schwartz et al., 2020; Xin et al., 2020; Zhou et al., 2020; Xin et al., 2021).

In this paper, we compare selective prediction performance of different NLP models and confidence estimators. We also propose a simple trick, error regularization, which can be applied to any of these models and confidence estimators, and improve their selective prediction performance. We further study the application of selective prediction on a variety of interesting applications, such as classification with no valid labels (no-answer problem) and using classifier cascades for accuracy–efficiency trade-offs. Experiments show that recent powerful NLP models such as BERT (Devlin et al., 2019) and ALBERT (Lan et al., 2020) improve not only accuracy but also selective prediction performance; they also demonstrate the effectiveness of the proposed error regularization by producing better confidence estimators which reduce the area under the *risk–coverage curve* by 10%.

2 Related Work

Selective prediction has been studied by the machine learning community for a long time (Chow, 1957; El-Yaniv and Wiener, 2010). More recently, Geifman and El-Yaniv (2017, 2019) study selective prediction for modern deep learning models, though with a focus on computer vision tasks.

Selective prediction is closely related to confidence estimation, as well as out-of-domain (OOD) detection (Schölkopf et al., 2000; Liang et al., 2018) and prediction error detection (Hendrycks and Gimpel, 2017), albeit more remotely. There have been many different methods for confidence estimation. Bayesian methods such as Markov Chain Monte Carlo (Geyer, 1992) and Variational Inference (Hinton and Van Camp, 1993; Graves, 2011) assume a prior distribution over model parameters and obtain confidence estimates through the posterior. Ensemble-based methods (Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017; Geifman et al., 2019) estimate confidence based on statistics of the ensemble model's output. These methods, however, are computationally practical for small models only. Current large-scale pre-trained NLP models, such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), are too expensive to run multiple times of inference, and therefore require lightweight confidence estimation.

Previously, selective prediction and confidence estimation have been studied in limited NLP scenarios. Dong et al. (2018) train a separate confidence scoring model to explicitly estimate confidence in semantic parsing. Kamath et al. (2020) introduce selective prediction for OOD question answering, where abstention is allowed for OOD and difficult questions. However, selective prediction for broader NLP applications has yet to be explored, and we hope to draw the attention of the NLP community to this problem.

There are two notable related topics, confidence calibration and unanswerable questions, but the difference between them and selective prediction is still nontrivial. Calibration (Guo et al., 2017; Jiang et al., 2018; Kumar et al., 2018; Wang et al., 2020; Desai and Durrett, 2020) focuses on adjusting the *overall* confidence level of a model, while selective prediction is based on *relative* confidence among the examples. For example, the most widely used calibration technique, temperature scaling (Platt, 1999), globally increases or decreases the model's confidence on all examples, but the ranking of all examples' confidence is unchanged. Unanswerable questions are considered in previous datasets, e.g., SQuAD2.0 (Rajpurkar et al., 2018). The unanswerable questions are impossible to answer even for humans, while abstention in selective prediction is due to model uncertainty rather than model-agnostic data uncertainty.

3 Background

We introduce relevant concepts about selective prediction and confidence estimators, using multi-class classification as an example.

3.1 Selective Prediction

Given a feature space $\mathcal{X}$ and a set of labels $\mathcal{Y}$, a standard classifier f is a function $f : \mathcal{X} \to \mathcal{Y}$. A *selective classifier* is another function $h : \mathcal{X} \to \mathcal{Y} \cup \{\perp\}$, where $\perp$ is a special label indicating the abstention of prediction. Normally, the selective classifier is composed of a pair of functions $h = (f, g)$, where f is a standard classifier and g is the *selective function* $g : \mathcal{X} \to \{0, 1\}$. Given an input $x \in \mathcal{X}$, the output of the selective classifier is as follows:

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

and we can see that the output of g controls prediction or abstention. In most cases, g consists of a

confidence estimator $\tilde{g} : \mathcal{X} \to \mathbb{R}$, and a *confidence threshold* θ:

$$g(x) = \mathbb{1}[\tilde{g}(x) > \theta]. \tag{2}$$

$\tilde{g}(x)$ indicates how confident the classifier f is on the example x, and θ controls the overall prediction versus abstention level.

A selective classifier makes trade-offs between *coverage* and *risk*. Given a labeled dataset $S = \{(x_i, y_i)\}_{i=1}^{n} \subset \mathcal{X} \times \mathcal{Y}$ and an error function $\mathcal{L}$ to calculate each example's error $l_i = \mathcal{L}(f(x_i), y_i)$, the coverage and the selective risk of a classifier $h = (f, g)$ on S are, respectively,

$$\gamma(h) = \frac{1}{|S|} \sum_{(x_i,y_i)\in S} g(x_i), \tag{3}$$

$$r(h) = \frac{\sum_{(x_i,y_i)\in S} g(x_i) l_i}{\sum_{(x_i,y_i)\in S} g(x_i)}. \tag{4}$$

The selective classifier aims to minimize the selective risk at a given coverage.

The performance of a selective classifier $h = (f, g)$ can be evaluated by the risk–coverage curve (RCC; El-Yaniv and Wiener, 2010), which is drawn by varying the confidence threshold θ (see Figure 2 for an example). Quantitatively, the *area under curve* (AUC) of RCC measures the effectiveness of a selective classifier.[1]

In order to minimize the AUC of RCC, the selective classifier should, intuitively, output $g(x) = 1$ for correctly classified examples and $g(x) = 0$ for incorrect ones. Therefore, an ideal $\tilde{g}$ has the following property: $\forall (x_i, y_i), (x_j, y_j) \in S$, $\tilde{g}(x_i) \le \tilde{g}(x_j)$ iff $l_i \ge l_j$. We propose the following metric, *reversed pair proportion* (RPP), to evaluate how far the confidence estimator $\tilde{g}$ is to ideal, given the labeled dataset S of size n:

$$\text{RPP} = \frac{\sum_{1\le i,j\le n}^{n} \mathbb{1}[\tilde{g}(x_i) < \tilde{g}(x_j), l_i < l_j]}{n^2}. \tag{5}$$

RPP measures the proportion of example pairs with a reversed confidence–error relationship, and the n^2 in the denominator is used to normalize the value. An ideal confidence estimator has an RPP value of 0.

3.2 Confidence Estimators

In most cases for multi-class classification, the last layer of the classifier is a softmax activation, which outputs a probability distribution $P(y)$ over the set of labels $\mathcal{Y}$, where $y \in \mathcal{Y}$ is a label. In this case, the classifier can be written as

$$f(x) = \hat{y} = \arg\max_{y\in\mathcal{Y}} P(y), \tag{6}$$

where $\hat{y}$ is the label with highest probability.

Perhaps the most straightforward and popular choice for the confidence estimator is softmax response (Hendrycks and Gimpel, 2017):

$$\tilde{g}_{\text{SR}}(x) = P(\hat{y}) = \max_{y\in\mathcal{Y}} P(y). \tag{7}$$

Alternatively, we can use the difference between probabilities of the top two classes for confidence estimation. We refer to this method as PD (probability difference).

Gal and Ghahramani (2016) argue that "softmax outputs are often erroneously interpreted as model confidence", and propose to use MC-dropout as the confidence estimator. In MC-dropout, $P(\hat{y})$ is computed for a total of R times, using a different dropout mask at each time, producing $P_1(\hat{y}), P_2(\hat{y}), \cdots, P_R(\hat{y})$. The variance of them is used to estimate the confidence:

$$\tilde{g}_{\text{MC}}(x) = -\text{Var}[P_1(\hat{y}), \cdots, P_R(\hat{y})]. \tag{8}$$

We use the negative sign here because a larger variance indicates a greater uncertainty, i.e., a lower confidence (Geifman and El-Yaniv, 2017; Kamath et al., 2020). By using different dropout masks, MC-dropout is equivalent to using an ensemble for confidence estimation, but does not require actually training and storing multiple models. Nevertheless, compared to SR, the inference cost of MC-dropout is multiplied by R, which can be a problem when model inference is expensive.

4 Error Regularization

4.1 Regularizers

SR and MC-dropout are often used directly out of the box as the confidence estimator. We propose a simple regularization trick that can be easily applied at training (or fine-tuning for pre-trained models) time and can improve the effectiveness of the induced confidence estimators.

Considering that a good confidence estimator should minimize RPP defined in Equation 5, we

[1] AUC in this paper always corresponds to RCCs.

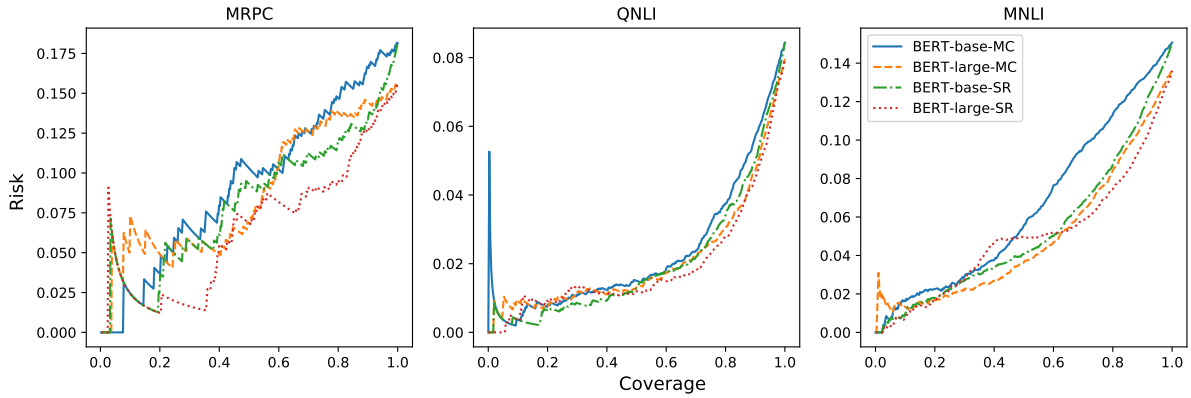

Figure 2: Risk–coverage curves of BERT-base and BERT-large models with SR and MC confidence estimators. The legend applies to all sub-plots.

add the following regularizer to the original training loss function:

$$L_{\text{total}} = \sum_{i=1}^{n} H(f(x_i), y_i) + \lambda L_{\text{reg}}, \quad (9)$$

$$L_{\text{reg}} = \sum_{1 \le i,j \le n} \Delta_{i,j} \ \mathbb{1}[e_i > e_j], \quad (10)$$

$$\Delta_{i,j} = \max\{0, \tilde{g}_{\text{SR}}(x_i) - \tilde{g}_{\text{SR}}(x_j)\}^2. \quad (11)$$

Here, $H(\cdot, \cdot)$ is the task-specific loss function such as cross entropy (H is not the same with the error function $\mathcal{L}$), λ is the hyperparameter for regularization, $\tilde{g}_{\text{SR}}$ is the maximum softmax probability defined in Equation 7, and e_i is the error of example i at the current iteration—details to calculate it will be explained in the next paragraph. We use SR confidence here because it is easily accessible at training time, while MC-dropout confidence is not. The intuition of this regularizer is as follows: if the model's error on example i is larger than its error on example j (i.e., example i is considered more "difficult" for the model), then the confidence on example i should *not* be greater than the confidence on example j.

In practice, at each iteration of training (fine-tuning), we can obtain the error e_i in one of the two following ways.

- **Current iteration error** We simply use the error function $\mathcal{L}$ to calculate the error of the example at the current iteration, and use it as e_i. In the case of multi-class classification, $\mathcal{L}$ is often chosen as the 0–1 error.

- **History record error** Since we intend to use e_i to quantify how difficult an example is, we draw inspiration from *forgettable examples* (Toneva et al., 2019). We calculate example error with $\mathcal{L}$ throughout the training process, and use the error averaged from the beginning to the current iteration as e_i. In this case, e_i takes value from $[0, 1]$.

4.2 Practical Approximations

In practice, it is computationally prohibitive to either strictly compute L_{reg} from Equation 10 for all example pairs, or to calculate history record error after every iteration. We therefore make the following two approximations.

For L_{reg} from Equation 10, we only consider examples from the mini-batch of the current iteration. For current iteration error, where e_i takes value from $\{0, 1\}$, we consider all pairs where $e_i = 1$ and $e_j = 0$. For history record error, where e_i takes value from $[0, 1]$, we sort all examples in the mini-batch by their errors, and divide the mini-batch into 20% of examples with high error values and 80% of examples with low error values;[2] then we consider all pairs where example i is from the former 20% and j from the latter 80%.

For calculating history record error, we compute and record the error values for the entire training set 10 times per epoch (once after each 10% iterations). At each training iteration, we use the average of error values recorded so far as e_i.

5 Experiments

We conduct experiments of selective prediction on NLP tasks. Since the formulation of selective prediction is model agnostic, we choose the

[2] We choose this 20–80 division to mimic the current iteration error case, where roughly 20% of training examples have an error of 1 and 80% have an error of 0.

Model	Confidence Estimator	MRPC F1(↑)	MRPC AUC(↓)	MRPC RPP(↓)	QNLI Acc(↑)	QNLI AUC(↓)	QNLI RPP(↓)	MNLI-(m/mm) Acc(↑)	MNLI-(m/mm) AUC(↓)	MNLI-(m/mm) RPP(↓)
LSTM	SR	81.8	101.5	9.0	62.7	1539.1	9.0	65.4/64.3	1984.0/1990.0	6.8/6.6
	MC		137.0	11.3		2039.8	11.6		3554.1/3548.1	11.7/11.7
BERT base	SR	87.7	33.8	3.7	91.6	111.9	1.2	84.9/84.6	514.8/491.7	2.6/2.4
	MC		38.3	4.5		130.1	1.3		639.0/677.5	3.4/3.5
BERT large	SR	89.0	27.0	3.2	92.0	105.6	1.1	86.4/86.0	486.1/470.0	2.5/2.4
	MC		35.9	4.3		114.6	1.2		482.0/510.2	2.5/2.6
ALBERT base	SR	90.9	16.0	2.1	90.9	122.8	1.2	84.7/85.4	469.3/453.5	2.3/2.3
	MC		43.9	5.6		160.0	1.6		921.1/878.3	5.0/4.7

Table 1: Comparing selective prediction performance of different models and confidence estimators. All metrics except AUC are in percentages.

Dataset	#Train	#Dev (m/mm)	#Labels
MRPC	3.7k	0.4k	2
QNLI	104.7k	5.5k	2
MNLI	392.7k	9.8k/9.8k	3
SST-5	8.5k	1.1k	5
bMNLI	261.8k	9.8k/9.8k	2+1
bSST-5	6.9k	1.1k	2+1

Table 2: Dataset statistics. bMNLI/bSST-5 are binarized version of MNLI/SST-5, with two normal labels and a special *no-answer* label.

following representative models: (1) BERT-base and BERT-large (Devlin et al., 2019), the dominant transformer-based models of recent years; (2) ALBERT-base (Lan et al., 2020), a variant of BERT featuring parameter sharing and memory efficiency; (3) Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber, 1997), the popular pre-transformer model that is lightweight and fast.

In this section, we compare the performance of selective prediction of these models, demonstrate the effectiveness of the proposed error regularization, and show the application of selective prediction in two interesting scenarios—the no-answer problem and the classifier cascades.

5.1 Experiment Setups

We conduct experiments mainly on three datasets: MRPC (Dolan and Brockett, 2005), QNLI (Wang et al., 2018), and MNLI (Williams et al., 2018). In Section 5.4, we will need an additional non-binary dataset SST-5 (Socher et al., 2013). Statistics of these datasets can be found in Table 2. Following the setting of the GLUE benchmark (Wang et al., 2018), we use the training set for training/fine-tuning and the development set for evaluation (the test set's labels are not publicly available); MNLI's development set has two parts, *matched* and *mismatched* (m/mm). These datasets include semantic equivalence judgments, entailment classification, and sentiment analysis, which are important application scenarios for selective prediction as discussed in Section 1.

The implementation is based on PyTorch (Paszke et al., 2019) and the Huggingface Transformers Library (Wolf et al., 2020). Training/fine-tuning and inference are done on a single NVIDIA Tesla V100 GPU. Since we are evaluating the selective prediction performance of different models instead of pursuing state-of-the-art results, we do not extensively tune hyperparameters; instead, most experiment settings such as hidden sizes, learning rates, and batch sizes are kept unchanged from the Huggingface Library. Further setup details can be found in Appendix A.

5.2 Comparing Different Models

We compare selective prediction performance of different models in Table 1. For each model, we report the performance given by the two confidence estimators, softmax response (SR) and MC-dropout (MC); the results of using PD for confidence estimation are very similar to those of SR, and we report them in Appendix B due to space limitations. The accuracy and the F1 score[3] measure the effectiveness of the classifier f, RPP measures the reliability of the confidence estimator $\tilde{g}$, and AUC is a comprehensive metric for both the classifier and the confidence estimator. The choice of confidence estimator does not affect the model's accuracy. We also provide risk–coverage curves (RCCs) of different models and confidence estima-

[3]We henceforth refer to both accuracy and F1 scores simply as *accuracy* for the sake of conciseness.

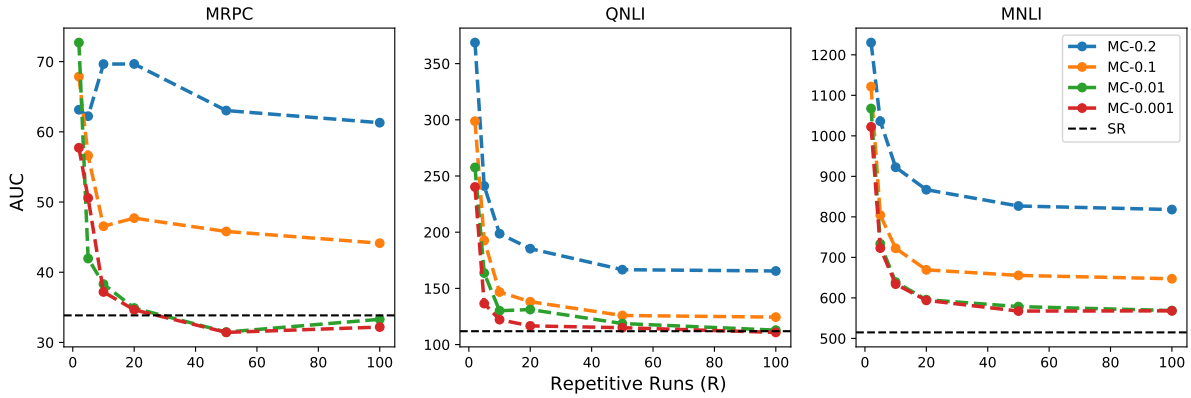

Figure 3: Selective prediction performance of MC-dropout with different numbers of repetitive runs (x-axis) and dropout rates (marked in the legend). BERT-base is used here. The legend applies to all sub-plots.

Model	Reg.	MRPC			QNLI			MNLI-(m/mm)		
		F1(↑)	AUC(↓)	RPP(↓)	Acc(↑)	AUC(↓)	RPP(↓)	Acc(↑)	AUC(↓)	RPP(↓)
LSTM	none	81.8	101.5	9.04	62.7	1539.1	8.99	65.4/64.3	1984.0/1990.0	6.81/6.60
	curr.	81.2	94.0	7.70	64.7	1376.2	8.51	65.5/64.4	1976.0/1990.5	6.80/6.64
	hist.	81.2	**92.3**	7.58	64.7	**1368.0**	8.42	65.3/64.6	**1974.4/1987.4**	6.74/6.71
BERT base	none	87.7	33.8	3.70	91.6	111.9	1.16	84.9/84.6	514.8/491.7	2.55/2.41
	curr.	88.1	31.2	3.49	91.9	**100.1**	1.08	84.6/84.6	**479.1/461.8**	2.39/2.27
	hist.	87.9	**30.3**	3.51	91.4	113.9	1.20	84.4/84.5	490.7/472.5	2.42/2.32
BERT large	none	89.0	27.0	3.17	92.0	105.6	1.06	86.4/86.0	486.1/470.0	2.45/2.39
	curr.	89.7	**20.6**	3.05	91.2	**98.2**	1.04	86.5/85.5	417.7/434.4	2.17/2.17
	hist.	89.0	24.4	3.30	92.1	99.4	0.94	85.5/85.9	**404.9/400.6**	2.25/2.25
ALBERT base	none	90.9	16.0	2.13	90.9	122.8	1.21	84.7/85.4	469.3/453.5	2.32/2.30
	curr.	91.4	**13.2**	1.82	90.9	**104.3**	1.23	84.7/85.2	**451.2**/463.9	2.25/2.23
	hist.	91.0	16.2	2.18	91.2	117.5	1.12	84.6/85.2	461.1/**429.8**	2.26/2.30

Table 3: Comparing different regularizers (Reg.) for different models and datasets. Selective prediction performance is measured by AUC and RPP. All metrics except AUC are in percentages.

tors in Figure 2. MC in the table and the figure uses a dropout rate of 0.01 and repetitive runs $R = 10$.

We first notice that models with overall higher accuracy also have better selective prediction performance (lower AUC and RPP). For example, compared with LSTM, BERT-base has higher accuracy and lower AUC/RPP on all datasets, and the same applies to the comparison between BERT-base and BERT-large. Since the classifier's effectiveness does not directly affect RPP, the consistency of RPP's and accuracy's improvement indicates that sophisticated models simultaneously improve *both* model accuracy and confidence estimation. This is in contrast to the discovery by Guo et al. (2017) that sophisticated neural networks, despite having better accuracy, are more easily overconfident and worse calibrated than simple ones.

We also notice that MC-dropout performs consistently worse than softmax response, shown by both AUC and RPP. This shows that for NLP tasks and models, model confidence estimated by MC-dropout fails to align well with real example difficulty. We further study and visualize in Figure 3 the effect of different dropout rates and different numbers of repetitive runs R on MC-dropout's selective prediction performance. We can see that (1) a dropout rate of 0.01 is a favorable choice: larger dropout rates lead to worse performance while smaller ones do not improve it; (2) MC-dropout needs at least 20 repetitions to obtain results comparable to SR, which is extremely expensive. Although MC-dropout has a sound theoretical foundation, its practical application to NLP tasks needs further improvements.

5.3 Effect of Error Regularization

In this part, we show that our simple regularization trick improves selective prediction performance. In

Model	Reg.	bSST5				bMNLI-(m/mm)			
		Acc(↑)	Acc*(↑)	AUC(↓)	RPP(↓)	Acc(↑)	Acc*(↑)	AUC(↓)	RPP(↓)
BERT base	none	71.7	74.0	174.7	5.34	63.9/64.2	70.6/70.9	1645.4/1630.7	4.74/4.72
	curr.	72.0	73.8	173.5	5.35	63.8/64.2	**71.1**/71.4	**1562.2/1562.1**	4.41/4.45
	hist.	72.6	**74.7**	**157.4**	5.17	63.8/64.1	70.7/**71.6**	1630.5/1583.2	4.62/4.51
BERT large	none	73.2	73.7	158.4	5.58	64.8/64.8	72.9/72.5	1861.0/1852.3	5.18/5.16
	curr.	73.3	**74.2**	**137.1**	4.82	64.5/65.1	72.7/73.2	**1476.8**/1629.7	4.91/4.68
	hist.	73.5	73.7	148.8	4.53	65.0/64.8	**73.1/73.2**	1695.9/**1460.6**	4.14/4.11
ALBERT base	none	72.3	**73.5**	172.4	5.61	64.0/64.3	71.6/72.4	1579.1/1534.4	4.44/4.34
	curr.	72.4	73.2	168.0	5.32	63.8/64.2	**72.9**/72.3	**1563.8**/1550.9	4.45/4.32
	hist.	72.5	73.2	**161.0**	5.63	63.9/64.4	71.6/**73.5**	1601.8/**1496.3**	4.20/4.10

Table 4: Selective prediction performance of different models and regularization methods (Reg.) on two datasets with the *no-answer* label. All metrics except AUC are in percentages.

Table 3, we report the accuracy, AUC, and RPP for each model, paired with three different regularizers: no regularization (none), current error regularizer (curr.), and history error regularizer (hist.), as described in Section 4.

We first see that applying error regularization (either current or history) does not harm model accuracy. There are minor fluctuations, but generally speaking, error regularization has no negative effect on the models' effectiveness.

We can also see that error regularization improves models' selective prediction performance, reducing AUC and RPP. As we mention in the previous section, AUC is a comprehensive metric for both the classifier f and the confidence estimator $\tilde{g}$. We therefore focus on this metric in this section, and we bold the lowest AUC in Table 3. We see that error regularization consistently achieve the lowest AUC values, and on average, the best scores are approximately 10% lower than the scores without regularization. This shows that error regularization produces confidence estimators that give better confidence rankings.

The two regularization methods, current error and history error, are similar in quality, with neither outperforming the other across all models and datasets. Therefore, we can conclude only that the error regularization trick improves selective prediction, but the best specific method varies. We leave this exploration for future work.

5.4 The No-Answer Problem

In this section, we conduct experiments to see how selective classifiers perform on datasets that either allow abstention or, equivalently, provide the *no-answer* label. This no-answer problem occurs whenever a trained classifier encounters an example whose label is unseen in training, which is common in practice. For example, in the setting of ultrafine entity typing with more than 10,000 labels (Choi et al., 2018), it is unsurprising to encounter examples with unseen types. Ideally, in this case, the classifier should choose the no-answer label. This setting is important yet often neglected, and there exist few classification datasets with the no-answer label. We therefore build our own datasets, binarized MNLI and SST-5 (bMNLI and bSST-5), to evaluate different models in this setting (Table 2).

The MNLI dataset is for sentence entailment classification. Given a pair of sentences, the goal is to predict the relationship between them, among three labels: entailment, contradiction, and neutral. The SST-5 dataset is for fine-grained sentence sentiment classification. Given a sentence, the goal is to predict the sentiment of it, among five labels: strongly positive, mildly positive, strongly negative, mildly negative, and neutral. To convert the original MNLI and SST-5 datasets into our binarized versions bMNLI and bSST-5, we modify the following: for SST-5, we merge strongly and mildly positive/negative into one positive/negative class; for MNLI, we simply regard entailment as positive and contradictory as negative. We then remove all neutral instances from the training set but keep those in the development and test sets. This way, neutral instances in the development and test sets should be classified as no-answer by the model. A good model is expected to assign neutral examples in the development and test sets with *low confidence scores*, thereby predicting the *no-answer* label for them.

We report results for these two datasets with

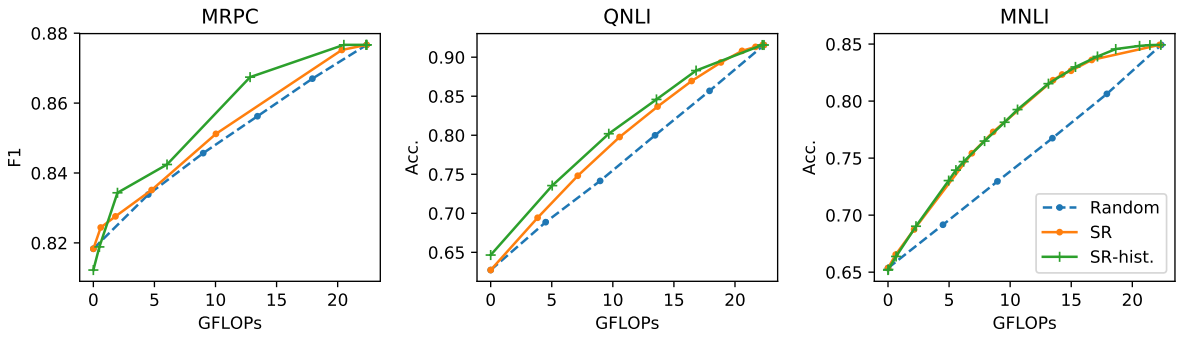

Figure 4: Accuracy–efficiency trade-offs by using classifier cascades. All examples are first evaluated by LSTM, and then we compare three ways of choosing examples to send to the more sophisticated model (BERT-base): random selection (Random), SR without regularization (SR), and SR with history error regularization (SR-hist.). The legend applies to all sub-plots.

the no-answer label in Table 4. Accuracy (Acc), AUC, and RPP have the same meaning from the previous sections. We also consider a new metric specifically for the no-answer setting, *augmented accuracy* (Acc*), which is calculated as follows: (1) we make a number of attempts by searching a threshold α from 0.7 to 1.0 in increments of 0.01; (2) for each attempt, we regard all examples with predicted confidence lower than α as neutral, and then calculate the accuracy; (3) among all attempts, we take the highest accuracy as Acc*. Choosing the optimal α requires knowing the ground-truth answers in advance and is not practical in reality.[4] Instead, Acc* indicates how well a model recognizes examples whose label is likely unseen in the training set.

We first see that Acc* is consistently higher than Acc in all cases. This is unsurprising, but it demonstrates that unseen samples indeed have lower confidence and shows that introducing the abstention option is beneficial in the no-answer scenario. Also, we observe that error regularization improves the models' selective prediction performance, producing lower AUC/RPP and higher Acc* in most cases. This further demonstrates the effectiveness of the simple error regularization trick.

Secondly, we can see that the improvement of Acc* over Acc is larger in bMNLI than in bSST-5. The reason is that in bMNLI, neutral examples constitute about a third of the entire development set, while in bSST-5 they constitute only a fifth. The improvement is positively correlated with the proportion of neutral examples, since they are assigned lower confidence scores and provide the potential for abstention-based improvements.

5.5 Classifier Cascades

In this section, we show how confidence estimation and abstention can be used for accuracy–efficiency trade-offs. We use classifier cascades: we first use a less accurate classifier for prediction, abstain on examples with low confidence, then send them to more accurate but more costly classifiers. Here we choose LSTM and BERT-base to constitute the cascade, but one can also choose other models and more levels of classifiers.

We first use an LSTM for all examples' inference, and then send "difficult" ones to BERT-base. Since the computational cost of LSTM is negligible[5] compared to BERT-base, the key to efficiency here is correctly picking the "difficult" examples.

In Figure 4, we show the results of accuracy/F1 score versus average FLOPs[6] per inference example. Each curve represents a method to choose difficult examples: The blue curves are obtained by randomly selecting examples, as a simple baseline. The orange and green curves are obtained by using SR of LSTM as the indicator of example difficulty; the orange curves represent the LSTM trained with no regularization while the green curves are with history error regularization. Different points on the curves are chosen by varying the proportion of examples sent to the more accurate model, BERT-

[4] Alternatively, one may use a validation set to choose the optimal α. In our experiments, however, we use the development set for evaluation, since the labels of the test set itself are not publicly available. Holding out a part of the training set for validation is left for future exploration.

[5] BERT-base's cost is $\sim 10^5$ times larger than LSTM here.

[6] We use the torchprofile toolkit to measure multiply–accumulate operations (MACs), and then double the number to obtain floating point operations (FLOPs).

base. A curve with a larger area under it indicates a better accuracy–efficiency trade-off.

We can see that the blue curves are basically linear interpolations between the LSTM (the lower-left dot) and BERT-base (the upper-right dot), and this is expected for random selection. Orange and green curves are concave, indicating that using SR for confidence estimation is, unsurprisingly, more effective than random selection. Between these two, the green curves (history error regularization) have larger areas under themselves than orange ones (no regularization), i.e., green curves have better accuracy given the same FLOPs. This demonstrates the effectiveness of error regularization for better confidence estimation.

6 Conclusion

In this paper, we introduce the problem of selective prediction for NLP. We provide theoretical background and evaluation metrics for the problem, and also propose a simple error regularization method that improves selective prediction performance for NLP models. We conduct experiments to compare different models under the selective prediction setting, demonstrate the effectiveness of the proposed regularization trick, and study two scenarios where selective prediction and the error regularization method can be helpful.

We summarize interesting experimental observations as follows:

1. Recent sophisticated NLP models not only improve accuracy over simple models, but also provide better selective prediction results (better confidence estimation).

2. MC-dropout, despite having a solid theoretical foundation, has difficulties matching the effectiveness of simple SR in practice.

3. The simple error regularization helps models lower their AUC and RPP, i.e., models trained with it produce better confidence estimators.

4. Selective prediction can be applied to scenarios where estimating example difficulties is necessary. In these cases, our proposed error regularization trick can also be helpful, such as providing better accuracy–efficiency trade-offs.

Future Work (1) Despite the effectiveness of the proposed error regularization trick, we are not certain on the best way for computing the error (current or history); it is important to unify them into one method that consistently does well. (2) We have only covered a selection of NLP tasks, and there are still other unexplored categories: token-level classification such as named entity recognition and question answering, sequence generation such as summarization and translation, and so on; it would be interesting to extend selective prediction to these problems. (3) There exists another setting for selective prediction where abstention induces a fixed cost (Bartlett and Wegkamp, 2008) and the goal is to minimize the overall cost instead of AUC; it would also be interesting to investigate this setting for NLP applications.

Acknowledgements

We thank anonymous reviewers for their constructive suggestions. This research is supported in part by the Canada First Research Excellence Fund and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Peter L. Bartlett and Marten H. Wegkamp. 2008. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(59).

Mark Carlebach, Ria Cheruvu, Brandon Walker, Cesar Ilharco Magalhaes, and Sylvain Jaume. 2020. News aggregation with diverse viewpoint identification using neural embeddings and semantic understanding models. In *Proceedings of the 7th Workshop on Argument Mining*, pages 59–66, Online. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96, Melbourne, Australia. Association for Computational Linguistics.

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

David Cohn, Zoubin Ghahramani, and Michael Jordan. 1995. Active learning with statistical models. In *Advances in Neural Information Processing Systems*, volume 7. MIT Press.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 743–753, Melbourne, Australia. Association for Computational Linguistics.

Ran El-Yaniv and Yair Wiener. 2010. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53).

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA. PMLR.

Yonatan Geifman and Ran El-Yaniv. 2017. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Yonatan Geifman and Ran El-Yaniv. 2019. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2151–2159. PMLR.

Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. 2019. Bias-reduced uncertainty estimation for deep neural classifiers. In *International Conference on Learning Representations*.

Charles J. Geyer. 1992. Practical markov chain monte carlo. *Statistical science*, pages 473–483.

Alex Graves. 2011. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*.

Geoffrey E. Hinton and Drew Van Camp. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational learning theory*, pages 5–13.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Heinrich Jiang, Been Kim, Melody Guan, and Maya Gupta. 2018. To trust or not to trust a classifier. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

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.

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. 2018. Trainable calibration measures for neural networks from kernel mean embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2805–2814. PMLR.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, pages 6402–6413. Curran Associates, Inc.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Shiyu Liang, Yixuan Li, and R. Srikant. 2018. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang,

Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Bernhard Schölkopf, Robert C. Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 2000. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651, Online. Association for Computational Linguistics.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. 2018. Deep active learning for named entity recognition. In *International Conference on Learning Representations*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 2019. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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.

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.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. 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. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 91–104, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Xuchao Zhang, Fanglan Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2019. Mitigating uncertainty in document classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3126–3136, Minneapolis, Minnesota. Association for Computational Linguistics.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. BERT loses patience: Fast and robust inference with early exit. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc.

Model	Confidence Estimator	SST-5 Acc	SST-5 AUC	SST-5 RPP	MNLI-(m/mm) Acc	MNLI-(m/mm) AUC	MNLI-(m/mm) RPP
LSTM	SR	37.6	624.8	10.1	65.4/64.3	1984.0/1990.0	6.8/6.6
	PD		646.5	10.8		2001.8/2013.8	6.9/6.7
	MC		701.2	11.9		3554.1/3548.1	11.7/11.7
BERT base	SR	51.6	439.9	10.1	84.9/84.6	514.8/491.7	2.6/2.4
	PD		438.9	10.2		517.8/494.0	2.6/2.4
	MC		493.5	10.9		639.0/677.5	3.4/3.5
BERT large	SR	53.3	430.5	10.4	86.4/86.0	486.1/470.0	2.5/2.4
	PD		434.2	10.5		489.2/473.0	2.5/2.4
	MC		474.8	11.2		482.0/510.2	2.5/2.6
ALBERT base	SR	50.2	474.2	10.5	84.7/85.4	469.3/453.5	2.3/2.3
	PD		481.3	10.8		473.1/456.6	2.3/2.3
	MC		524.4	11.8		921.1/878.3	5.0/4.7

Table 5: Adding PD as confidence estimation to the comparison between different confidence estimators on SST-5 and MNLI.

A Detailed Experiment Settings

The LSTM is randomly initialized without pre-training. For models that require pre-trained, we use the following ones provided by the Huggingface Transformer Library (Wolf et al., 2020).

- BERT-BASE-UNCASED
- BERT-LARGE-UNCASED
- ALBERT-BASE-V2

All these models are trained/fine-tuned for 3 epochs without early-stopping or checkpoint selection. Learning rate is 2×10^{-5}. A batch size of 32 is used for training/fine-tuning. The maximum input sequence length is 128. Choices for the regularization hyperparameter λ from Equation 9 are shown in Table 6.

The numbers of parameters for the two models BERT and ALBERT can be found in the paper by Lan et al. (2020).

The LSTM used in the paper is a two-layer bi-directional LSTM, with a hidden size of 200. On top of it there is a max-pooling layer and a fully-connected layer.

Model	curr.	hist.
LSTM	0.5	0.5
BERT-base	0.05	0.05
BERT-large	0.1	0.1
ALBERT-base	0.01	0.05

Table 6: Choices for λ for different models and regularization methods.

B PD Confidence Estimator

Probability difference (PD), the difference between probabilities of the top two classes, can also be used as confidence estimation. Among the four datasets used in the paper, MRPC and QNLI are binary classification, and therefore PD's results are identical to softmax response (SR). SST-5 and MNLI have more than two classes, and therefore PD's results are different from SR's. We show them in Table 5.

We can see that the results of PD are very similar to those of SR. Of course, MNLI and SST-5 have only three/five labels respectively, and for datasets with far more labels, PD will possibly show its difference from SR.