

# Protoformer: Embedding Prototypes for Transformers

Ashkan Farhangi\*, Ning Sui, Nan Hua, Haiyan Bai, Arthur Huang, and Zhishan Guo

University of Central Florida, Orlando FL, USA  
ashkan.farhangi@ucf.edu, zhishan.guo@ucf.edu

**Abstract.** Transformers have been widely applied in text classification. Unfortunately, real-world data contain anomalies and noisy labels that cause challenges for state-of-art Transformers. This paper proposes Protoformer, a novel self-learning framework for Transformers that can leverage problematic samples for text classification. Protoformer features a selection mechanism for embedding samples that allows us to efficiently extract and utilize anomalies prototypes and difficult class prototypes. We demonstrated such capabilities on datasets with diverse textual structures (e.g., Twitter, IMDB, ArXiv). We also applied the framework to several models. The results indicate that Protoformer can improve current Transformers in various empirical settings.

**Keywords:** Text Classification · Twitter Analysis · Class Prototype

## 1 Introduction

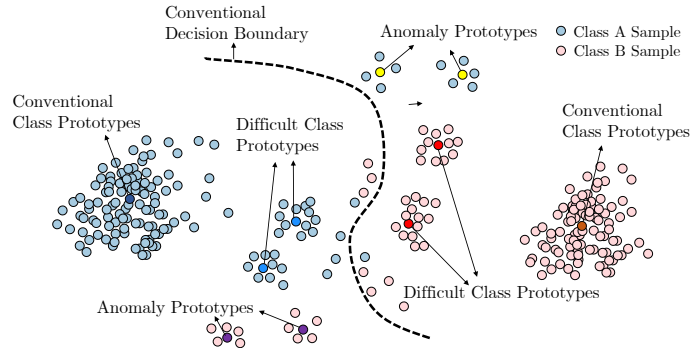
For real-world textual datasets, anomalies are known as samples that depart from the standard samples. Such anomalies tend to have scattered textual distributions, which can cause performance drops for state-of-art Transformer models [13]. Moreover, models that rely on supervised learning can suffer from incorrect convergence when provided with noisy labeled data gathered from Internet [14]. Hence, there is a need to automatically detect the anomalies and adjust noisy labels to make the model more robust to complex noisy datasets.

As human annotations can be highly time-and-cost inefficient, it is more common that noisy labels are gathered from the Internet. For instance, Twitter has been increasingly adopted to understand human behavior [3]. However, such data tend to complex and often contain noisy labels. This can make the standard supervised learning objective lead to incorrect convergence [4].

One of the applications of this study is to classify college students' academic major choices based on their historical Tweets. When students follow a certain college's official account, it might indicate that the student belongs to that major. However, there are uncertainties about the correctness of the labels. Therefore, the supervised model's results can become untrustworthy.

---

\* Corresponding author.



**Fig. 1.** Distribution of embeddings for real world data samples is often scattered. Although conventional class prototypes are easier to select, difficult class prototypes and anomaly prototypes require a more careful approach in selection and play a critical role in improving the decision boundary.

There are some prior works on prototype embeddings. CleanNet [7] proposes providing extra supervision for the training. Subsequently, SMP [5] proposes using multiple prototypes to capture embeddings with high density without extra human supervision. However, both approaches do not provide a solution for troublesome embeddings that are scattered and are often minorities, as shown in Figure 1. To alleviate this issue, we select prototypes through their contextual embeddings in a way to not only cover the difficult-to-classify samples but also represent minority samples of the dataset (i.e., anomalies).

We propose Protoformer framework that selects and leverages multiple embedding prototypes to enable Transformer’s specialization ability to classify noisy labeled data populated with anomalies. Specifically, we improve the generalization ability of Transformers for problematic samples of a class through *difficult class prototypes* and their specialization ability for minority samples of a class through *anomaly prototypes*. We show that the representations of both prototypes are necessary to improve the model’s performance. Protoformer leverages these prototypes in a self-learning procedure to further improve the robustness of textual classification. To our best knowledge, this is the first study that extracts and leverages anomaly prototypes for Transformers.

In summary, the contributions are threefold:

- We propose a novel framework that learns to leverage harder to classify and anomaly samples. This acts as a solution for classifying datasets with complex samples crawled from the Internet.
- The framework contains a label adjustment procedure and thus is robust to noise. This makes the framework suitable for noisy Internet data and can be used to promote a more robust Transformer model. Leveraging the similarity in the embedding space and a ranking metric, we can identify questionable labels and provide a certain level of adjustment. This mitigates the potential negative impact on the training.

- We evaluate the framework based on multiple datasets with both clean and noisy labels. Results show that our model improves the testing accuracy from 95.7% to 96.8% on the IMDB movie review dataset. For a self-gathered Twitter dataset with noisier labels, the classification accuracy improved with a greater margin (from 56.7% to 81.3%).

## 2 Problem Formulation

Given a sample text as  $\mathbf{x}_i$ ,  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  represents all the  $N$  samples of the dataset, while  $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_N\}$  indicates the corresponding noisy labels from the Internet. The noisy label  $\hat{y}_i \in \{0, 1\}^{\bar{c}}$  is a binary vector format with only one non-zero element, indicating the class label of  $\mathbf{x}_i$ , where  $\bar{c}$  is the total number of classes. A Transformer model  $\mathcal{F}_W$  can be used as a classification model to produce an estimated label  $\mathcal{F}_W(\mathbf{x}_i) \in [0, 1]^{\bar{c}}$ , where  $W$  represents the parameters. The optimization strategy is based on the cross-entropy loss function:

$$\mathcal{L}(\mathcal{F}_W(\mathbf{x}_i), \tilde{y}_i) = - \sum_{j=1}^{\bar{c}} \tilde{y}_{i,j} \log(\mathcal{F}_W(\mathbf{x}_i)_j), \quad (1)$$

In addition, labels from the internet are often noisy. Hence, as detailed in Section 3.4, the labels can be adjusted according to the similarities of the class prototypes, resulting in adjusted labels  $\tilde{y}_i \in [0, 1]^{\bar{c}}$ —it is a probability distribution, and thus  $\sum_{j=1}^{\bar{c}} \tilde{y}_{i,j} = 1$ . Even when we have sufficient confidence in the original labels, we can use it as a complementary supervision.

Specifically, for each batch with  $m$  samples, we would pursue the following optimization problem:

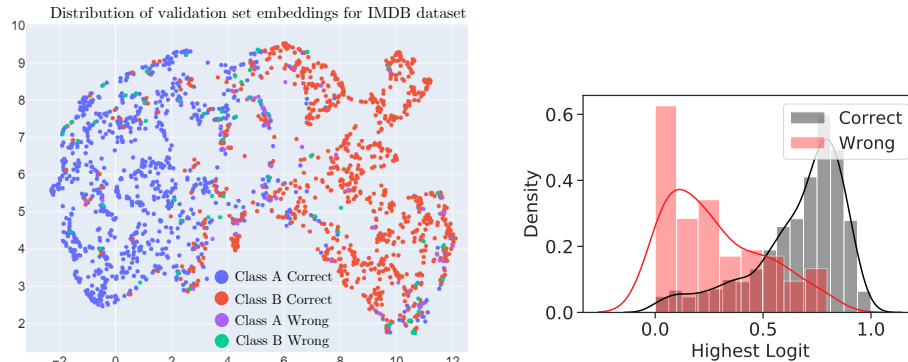
$$W^* = \operatorname{argmin}_W \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\mathcal{F}_W(\mathbf{x}_i), \tilde{y}_i) \quad (2)$$

## 3 Design of Protoformer

This section provides the details of Protoformer. Specifically, we describe a procedure for extracting the difficult class prototypes (Section 3.1). Subsequently, we describe a procedure for extracting anomaly prototypes (Section 3.2). Both types of prototypes are then used in a multi-objective self-learning training process that optimizes the network parameters for robust text classification (Section 3.3). In order to handle noisy labeled data, we adjust the noisy labels through a label adjustment procedure that uses the prototype similarities (Section 3.4).

### 3.1 Difficult Class Prototypes

Difficult class prototypes act as the representatives for the problematic samples of the dataset. For example, Figure 2 showcases the fine-tuned embeddings of



**Fig. 2.** **Left:** distribution of the embedding for IMDB dataset. Presence of anomalies and problematic samples cause misclassification. **Right:** Distribution of the highest output logits (scaled 0-1) for the same model. The higher values of the largest logits can represent the confidence of the network’s classification.

a benchmark dataset gathered from the Internet (i.e., IMDB). Although the majority of samples of each class are located closely together, there are anomaly samples that are scattered and often far from the majority. Unfortunately, these harder-to-classify samples are not the target focus of the state-of-art models in text classification. Moreover, traditional clustering methods (e.g., K-means) are not designed to capture or cluster such samples that are scattered and distributed throughout the embedding space.

Intuitively, these problematic samples can cause the greatest error. For instance, Figure 2 also shows the classification error of the fine-tuned BERT [2] model where the majority of the classification error stems from harder-to-classify samples (over 51%). Such error arises when the highest classification logit values are still low and in between classes, which indicates the indecisiveness of the Transformer. Following [5], we define the similarity of the extracted embeddings through **pairwise similarity score** (i.e., cosine distance) of any two inputs  $\mathbf{x}_i$  and  $\mathbf{x}_j$  as:

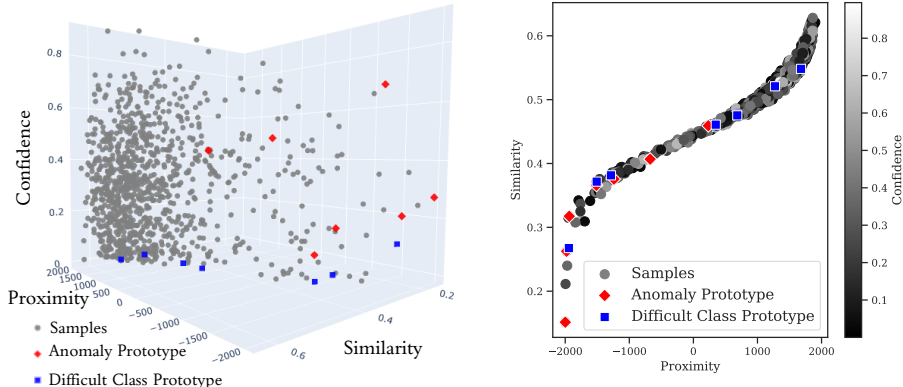
$$s_{ij} = \frac{\mathbf{e}(\mathbf{x}_i)^T \cdot \mathbf{e}(\mathbf{x}_j)}{\|\mathbf{e}(\mathbf{x}_i)\|_2 \|\mathbf{e}(\mathbf{x}_j)\|_2}, \quad (3)$$

where  $\mathbf{e}(\mathbf{x})$  is the embedding vector of sample  $\mathbf{x}$ , extracted from the first layer of the Transformer<sup>1</sup>.

To determine the closeness of embeddings, we also define the **proximity** metric  $p$  for each embedding as:

$$p_i = \sum_{j=1}^m \text{sign}(s_{ij} - s_c), \quad (4)$$

<sup>1</sup> For large-scale datasets, one can randomly choose a limited number (e.g.,  $q$ ) of samples per class to develop a triangular similarity matrix  $S^{q \times q}$  which can enhance the computational efficiency.



**Fig. 3.** Selected embedding prototypes of a single class of Twitter dataset. Difficult class prototypes have higher proximity, while anomaly prototypes suffer from low proximity due to their complex nature.

where  $sign(x)$  is a sign function<sup>2</sup> and  $s_c$  is an arbitrary value from the similarity matrix (default as 20-percentile). Intuitively, a higher proximity indicates that the textual embeddings have more similar embeddings around them and are ‘closer’ to every other sample in the embedding space.

Following [12], problematic samples cause low confidence in output logits of the model. Hence, we define the **confidence** metric  $c$  as:

$$c_i = \left| \overbrace{\max_{\hat{c}^1} \mathcal{F}_W(\mathbf{x}_i)_{\hat{c}^1}}^{\text{largest logit}} - \overbrace{\max_{\hat{c}^2} \mathcal{F}_W(\mathbf{x}_i)_{\hat{c}^2}}^{\text{second largest logit}} \right| \quad (5)$$

where logits are scaled (0-1 range) and are taken from the output before the softmax layer after a preliminary training stage. Intuitively, when the confidence is low (near zero), the model indecisiveness is the highest.

We can now represent the embeddings in a three-dimensional space as shown in Figure 3 (similarity-proximity-confidence). The difficult class prototype selection follows three general rules: (i) it should prioritize low confidence samples (i) it should be ‘far’ enough from existing prototypes (if any), (iii) it should have high ‘proximity’ when possible. To this end, the first prototype with the lowest confidence, highest proximity, and highest similarity is chosen. Then, the subsequent difficult class prototypes are chosen in a logsparse [8] manner for every round with an exponential selection step of sample size  $(\log_2(N))$ . Note that the samples are selected based on the low confidence, then high proximity but should have the lowest average similarity with the previously selected prototypes to be distinctive from each other. This strategy ensures us that the difficult class prototype are well represent problematic samples of the dataset.

<sup>2</sup>  $sign(x) = 1$  for  $x > 0$ ,  $sign(x) = 0$  for  $x = 0$ , and  $sign(x) = -1$  otherwise.

Next, at a certain round ( $t$ ), a prototype set  $\mathbf{X}_c = \{\mathbf{x}_c^{(1)}, \dots, \mathbf{x}_c^{(t)}\}$  is already formed for the  $c$ -th class,  $c = 1, \dots, \bar{c}$ . Given any text  $\mathbf{x}_i$ , we can calculate the average cosine similarity between sample  $\mathbf{x}_i$  and the selected prototype embeddings as:

$$s_{i,(c)}^c = \frac{1}{t} \sum_{j=1}^t s_{i,c^{(j)}}, \quad (6)$$

where  $s_{i,(c)}^c$  is the average similarity of difficult class embeddings in the  $j$ th iteration for the  $c$ -th class. This average similarity can then be used as a complementary supervision:

$$z_i^c = \operatorname{argmax}_c \{s_{i,(c)}^c | c = 1, \dots, \bar{c}\}. \quad (7)$$

As shown in Figure 3, difficult prototypes are chosen with low confidence levels, where they have the least similarity among the previously selected prototypes. During this process, we ensure that the subsequent prototypes stay far enough from existing prototypes so that there are limited redundant representations of the similar samples.

### 3.2 Anomaly Prototypes

Anomaly prototypes are the selected sample prototypes that represent the scattered and shattered minority samples of a dataset. Such samples are often harder to detect and tend to deviate from normal samples.

Given that the remaining classification error can be caused by such anomalies, it’s important to not only capture such anomalies robustly but also leverage them for the optimization objectives of Transformers.

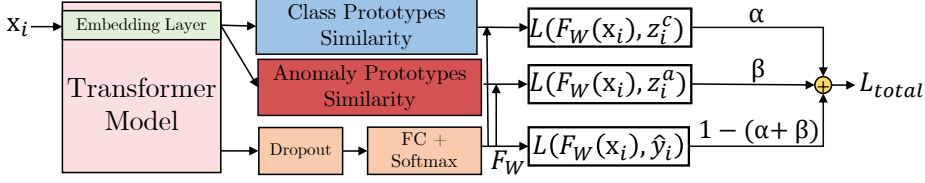
So far, difficult class prototypes can cover the problematic samples as they are detected by having high proximity and similarity. However, a certain portion of prototypes may be located ‘far’ from the difficult class prototypes and often represent the minority members of a class, as indicated by the red dots in the in Figure 3. Such prototypes represent the minority of samples as they have a lower density.

The prototype with the least proximity  $p_{min}$  is selected in the first round. This ensures us that the elected prototype is representative of the minority samples. We then select the subsequent ones in the same logsparse manner as before, ensuring that the prototypes have the least similarity. The similarity score is calculated in a similar manner to Equation (6) while including the anomaly prototypes in the summation.

Figure 2 also illustrates the process, where gray dots represent all sample embeddings, and red dots indicate the embeddings of selected anomaly prototypes.

### 3.3 Multi-Objective Self-Learning

Transformers used in text classification often rely on a single source of supervision which is the given labels. However, such design choice limits the Trans-



**Fig. 4.** Protoformer leverages the embedding space to derive the difficult class and anomaly prototypes. The network is trained jointly on Transformer and similarity of embedding prototypes. The total loss is dependent on the  $\alpha$  and  $\beta$  values which are estimated in the training phase.

former’s ability to perform well when the datasets are noisy labeled. Moreover, anomaly samples appear less in training compared to samples with high similarity. Note that majority of self-learning objectives for Transformers are to provide the greatest level of classification accuracy for all samples regardless of whether they are in the majority or minority. Intuitively, such a self-learning objective does not guarantee that the model suits well for minority classes due to their lower occurrence. In order to incorporate our prototypes during the training and test stage, we introduce a multi-objective self-learning mechanism to Protoformer.

As shown in Figure 4, the similarities of embedding prototypes are used as self-supervision to train the Protoformer  $\mathcal{F}_W$  after its fine-tuning state. The self-supervision is provided by the class prototype as below:

$$\mathcal{L}_{\text{proto}} = \frac{1}{m} \sum_{i=1}^m (\alpha \cdot \mathcal{L}(\mathcal{F}_W(\mathbf{x}_i), z_i^c) + \beta \cdot \mathcal{L}(\mathcal{F}_W(\mathbf{x}_i), z_i^a)), \quad (8)$$

where the weight factors  $\alpha, \beta \in [0, 1)$  and  $\alpha + \beta < 1$  indicate the concentration of Transformer on the similarities of self-supervision of difficult class prototypes  $z_i^c$  and anomaly prototypes  $z_i^a$ . Hence, the overall loss is calculated by minimizing the classification loss based on three components:

$$\mathcal{L}_{\text{total}} = (1 - (\alpha + \beta)) \cdot \frac{1}{m} \sum_{i=1}^m (\mathcal{L}(\mathcal{F}_W(\mathbf{x}_i), \hat{y}_i) + \mathcal{L}_{\text{proto}}), \quad (9)$$

To this end, when the network’s predictions are in between classes, the network can improve its training by the self-supervision provided by the similarity of difficult class prototype  $z_i^c$  and anomaly prototype  $z_i^a$ . Hence, we continue the training procedure iteratively until convergence:  $W^{(t+1)} \leftarrow W^{(t)} - \xi \nabla(\mathcal{L}_{\text{total}})$ , where the gradient descent vector  $\nabla(\mathcal{L}_{\text{total}})$  holds the partial derivatives of weights and biases of the total loss function, and  $\xi$  is the learning rate. We use a fully connected layer over the final hidden state corresponding to the output token of the Transformer (i.e., [CLS] token). The softmax activation function is then applied to the hidden layer to provide classification. It is important to note that this procedure can also be implemented solely during the test stage,

which can make the calculation timing complexity of Protoformer similar to the fine-tuning process.

### 3.4 Noisy Labels Enhancement

To mitigate the effect of noisy labels throughout the datasets, we are enhancing the labels through the similarities of embedding prototypes. This allows Protoformer to be robust toward datasets when the labels are not fully trustworthy. Consequently, when the labels are wrong, the training procedure of Transformers provides suboptimal weights, which makes the classification results untrustworthy.

Specifically, we can obtain the adjusted label of the a noisy labeled sample through maximum similarity to the difficult class prototype:

$$\tilde{y}_i = \operatorname{argmax}_c \{s_{i,(c)} | c = 1, \dots, \bar{c}\}, \quad (10)$$

where  $s_{i,(c)}$  is the cosine similarity defined in Equation (6) and the enhanced labels  $\tilde{y}$  can be used as a replacement for the noisy labels. Thus, the overall loss is calculated in a similar manner as Equation (9), while we are replacing the original noisy labels with the adjusted label.

## 4 Experiments

In this section, we provide descriptions for the datasets. We also describe the experimental settings and evaluation results. Lastly, we provide an analysis section that further discusses the effectiveness of Protoformer components.

### 4.1 Benchmark Datasets & Baselines

We have experimented with three challenging real-world datasets<sup>3</sup>. The brief discussion for each dataset is as follows:

**Table 1.** Summary statistics of the evaluation dataset.

Dataset	Twitter-Uni	IMDb	Arxiv-10
# Examples	25,000	25,000	100,000
# Train	20,000	20,000	80,000
# Validation	2,500	2,500	10,000
# Test	2,500	2,500	5,000
# Classes	8	2	10

**Twitter-Uni**<sup>3</sup>. We crawled over 12 million historical Tweets of 25,000 Twitter profiles from 8 U.S. college followers. As an example, the college of engineering holds near 3000 followers, which are labeled as engineering. Note that most existing benchmark Twitter datasets fail to hold high-quality labels that are

<sup>3</sup> Self-gathered datasets are accessible at <https://github.com/ashfarhangi/Protoformer>



provided by the original Twitter users. To alleviate this issue, we extracted a set of students that stated their major in their Twitter bio. This set can serve as ground truth of the clean labels. We made this challenging new dataset available online, which can be used for future text classification or noisy label correction studies.

**ArXiv-10<sup>3</sup>.** We also crawled the abstracts and titles of 100 thousand ArXiv scientific papers on ten research categories that include subcategories of computer science, physics, and math. The dataset is downsampled to contain exactly 10 thousand samples per category.

**IMDB.** The third dataset is the benchmark IMDB movie reviews [10]. The dataset is widely used as the sentiment classification task. It contains 25 thousand samples per sentiment (positive or negative). Both IMDB and ArXiv-10 datasets are originally labeled by the authors. It is however good to note that the labels are still susceptible to noisy labels.

The **baseline** methods for comparison include:

- SVM [11], supervised learning with a linear separator to maximize the margin between classes, with the fine-tuned embeddings derived from the Transformers.
- HAN [6], a hierarchical attention network for textual classification with word and sentence-level attention mechanisms.
- DocBERT [1], a document Transformer model with an LSTM architecture rather than a fully connected layer.
- RoBERTa [9], a Transformer with an improved pretraining procedure. Specifically, showing improvement by removing the next sentence prediction pretraining objective.

**Table 2.** Hyperparameters of the Protoformer used for each dataset.

Parameter	Twitter-Uni	IMDb	Arxiv-10
Batch size	32	64	32
Learning rate	$5 \times 10^{-5}$	$3 \times 10^{-5}$	$5 \times 10^{-5}$
Weight decay	$5 \times 10^{-5}$	$1 \times 10^{-5}$	$1 \times 10^{-4}$
Preliminary training epochs	5	3	2
Fine-tuning epochs	20	10	10
Training time	1:49h	1:32h	1:45h
Transformer	DistilBERT	BERT	RoBERTa

## 4.2 Experimental Settings

To showcase the generalization ability of our framework, we selected a unique Transformer for each dataset (Table 2). The hyperparameters are based on the highest Macro-F1 score obtained on the validation set for all models (following the standard 80-10-10 split). We used a grid search approach to explore the

**Table 3.** Evaluation of the Protoformer and baseline methods.

Model	Twitter			IMDb			ArXiv		
	Ma-F1	Recall	Acc	Ma-F1	Recall	Acc	Ma-F1	Recall	Acc
SVM [11]	0.384	0.361	0.391	0.744	0.733	0.748	0.691	0.654	0.708
HAN [15]	0.412	0.392	0.425	0.894	0.882	0.896	0.732	0.696	0.746
DocBERT [1]	0.521	0.506	0.534	0.932	0.921	0.936	0.752	0.727	0.764
RoBERTa [9]	0.555	0.531	0.567	0.952	0.941	0.957	0.769	0.732	0.779
Protoformer	<b>0.802</b>	<b>0.784</b>	<b>0.813</b>	<b>0.964</b>	<b>0.952</b>	<b>0.968</b>	<b>0.784</b>	<b>0.744</b>	<b>0.794</b>

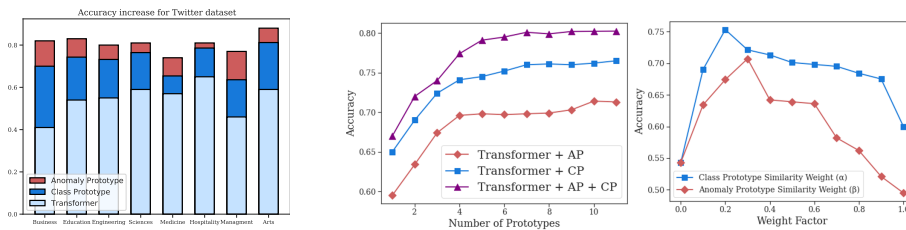
hyperparameters: size of fully connected layer  $H^D \in \{256, 512, 768, 1024\}$  and dropout  $\delta \in \{0.0, 0.1, \dots, 0.9\}$ . The experiments are conducted using PyTorch on a cloud workstation using Nvidia Tesla A100 GPU.

### 4.3 Experimental Results

For a less noisy labeled datasets such as IMDB and Arxiv, the evaluated methods performed comparatively. Note that the majority of the classification error appears when the network does not show confidence in its classification, as was previously shown for the IMDB dataset in Figure 5. The Protoformer is also able to provide a competitive accuracy for cleaner datasets such as IMDb and ArXiv-10. Among the baselines, the performance of RoBERTa [9] is favorable compared to others. This is partly due to the different pretraining objectives from DocBERT. As shown in Table 3, Protoformer resulted in the highest margin of accuracy for a noisy dataset, improving the Macro-F1 score from 55.5% to 80.2% for the Twitter-Uni dataset. We observed that this dataset provides the greatest difficulties for baseline methods where the models often misclassify problematic samples. To this end, we report a detailed accuracy breakdown for the Twitter dataset in Figure 5. The fine-tuning process for Transformers such as DocBERT, RoBERTa results in suboptimal classification. Leveraging the selected prototypes, Protoformer was able to improve its classification accuracy on the harder and more complex samples (e.g., management students that are similar to other classes). To this end, the fine-tuning process alone does not result in adequate accuracy due to the noise of the dataset. The combination of both embedding prototypes allows the Transformer to have a solution for anomalies and problematic samples of the dataset and further improves its generalization ability through difficult class prototypes.

### 4.4 Analysis

In this section, we provide an extensive analysis of the performance of Protoformer, as well as the role of each type of prototype on the overall performance. Hence, we limited the number of prototypes per class for the Twitter dataset and reported the changes. The results in Figure 5 show that a single prototype is not sufficient to provide competitive accuracy even with the help of a fine-tuned



**Fig. 5.** **Left:** Accuracy increase from the initial (light blue), class prototype (blue) and class anomalies (red), for Twitter dataset. **Right:** Influence of anomaly labeling of hurricanes for Collier county gross hotel sales revenue. **Middle:** Number of anomaly prototypes (AP) and difficult class prototypes (CP) per class for Twitter dataset. Higher number of prototypes resulted in marginal improvement while the combination of both category of prototypes gives us the optimal accuracy. **Right:** Testing accuracy with respect to the weight factors ( $\alpha$  and  $\beta$ ) ranging from 0 to 1.

Transformer. However, as the number of prototypes increased, we observed improvements in the accuracy of the Protoformer. The prototype selection procedure previously discussed ensures that there are multiple prototypes for every proximity metric, and the calculation of them is computationally expensive even for the large-scale dataset. Moreover, the weight factors are reported separately to showcase the effect of their self-supervision for the Twitter dataset. The results show that relying on the noisy labels ( $\alpha$  and  $\beta = 0$ ) during training would be suboptimal and perform poorly on confirmed test data. Moreover, the accuracy would be optimal when weight factors sum to 0.5 (i.e.,  $\alpha=0.2$ ,  $\beta=0.3$ ).

## 5 Conclusion

In this work, we developed a novel Transformer framework, Protoformer, that leverages the embedding prototypes of the dataset to enhance its generalization and specialization abilities. It also includes a procedure for handling noisy labels. Various experiments are conducted to demonstrate the effectiveness of Protoformer over state-of-art topic and sentiment classification methods. For future work, we are interested in applying Protoformer for the image recognition tasks. We also like to explore the use of Protoformer on spherical and hyperbolic embedding space.

### Acknowledgement

Our work has been supported by the US National Science Foundation under grants No. 2028481, 1937833, and 1850851.

## References

1. Adhikari, A., Ram, A., Tang, R., Lin, J.: Docbert: Bert for document classification. arXiv preprint arXiv:1904.08398 (2019)

2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
3. Fiok, K., Karwowski, W., Gutierrez, E., Saeidi, M., Aljuaid, A.M., Davahli, M.R., Taiar, R., Marek, T., Sawyer, B.D.: A study of the effects of the covid-19 pandemic on the experience of back pain reported on twitter® in the united states: A natural language processing approach. *International Journal of Environmental Research and Public Health* **18**(9), 4543 (2021)
4. Garg, S., Vu, T., Moschitti, A.: Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 7780–7788 (2020)
5. Han, J., Luo, P., Wang, X.: Deep self-learning from noisy labels. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5138–5147 (2019)
6. Krishnan, R., Shalit, U., Sontag, D.: Structured inference networks for nonlinear state space models. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31 (2017)
7. Lee, K.H., He, X., Zhang, L., Yang, L.: Cleannet: Transfer learning for scalable image classifier training with label noise. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5447–5456 (2018)
8. Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.X., Yan, X.: Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems* **32** (2019)
9. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
10. Maas, A., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies. pp. 142–150 (2011)
11. Meyer, D., Leisch, F., Hornik, K.: The support vector machine under test. *Neurocomputing* **55**(1-2), 169–186 (2003)
12. Pleiss, G., Zhang, T., Elenberg, E., Weinberger, K.Q.: Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems* **33**, 17044–17056 (2020)
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. *Advances in neural information processing systems* **30** (2017)
14. Wei, H., Feng, L., Chen, X., An, B.: Combating noisy labels by agreement: A joint training method with co-regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13726–13735 (2020)
15. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies. pp. 1480–1489 (2016)