Exploring Generalizability of Fine-Tuned Models for Fake News Detection

Abhijit Suprem
School of Computer Science
Georgia Institute of Technology
Atlanta, USA
asuprem@gatech.edu

Sanjyot Vaidya
School of Computer Science
Georgia Institute of Technology
Atlanta, USA

Calton Pu
School of Computer Science
Georgia Institute of Technology
Atlanta, USA

***Abstract*—The Covid-19 pandemic has caused a dramatic and parallel rise in dangerous misinformation, denoted an 'infodemic' by the CDC and WHO. Misinformation tied to the Covid-19 infodemic changes continuously; this can lead to performance degradation of fine-tuned models due to concept drift. Degredation can be mitigated if models generalize well-enough to capture some cyclical aspects of drifted data. In this paper, we explore generalizability of pre-trained and fine-tuned fake news detectors across 9 fake news datasets. We show that existing models often overfit on their training dataset and have poor performance on unseen data. However, on some subsets of unseen data that overlap with training data, models have higher accuracy. Based on this observation, we also present KMeansProxy, a fast and effective method based on K-Means clustering for quickly identifying these overlapping subsets of unseen data. KMeans-Proxy improves generalizability on unseen fake news datasets by 0.1-0.2 f1-points across datasets. We present both our generalizability experiments as well as KMeans-Proxy to further research in tackling the fake news problem.**

I. Introduction

The rapid spread of the Covid-19 virus has led to a parallel surge in misinformation and disinformation [1] This surge of false information, coined an 'infodemic' by the CDC [2] can be life-threatening, destabilizing, and potentially dangerous [3]. The infodemic is multimodal, meaning associated fake news can take forms of social media posts, tweets, articles, blogs, commentary, misrepresented titles and headlines, videos, and audio content.

Current Research. There is significant progress on developing domain-specific automated misinformation detection and classification tools [4], [5], [6], [7], [8], [9]. Such tools analyze labeled datasets in aforementioned modalities to classify fake news. Recent approaches focus on transformer-based classifiers and language modelers [9], [7].

Such fake news detectors are specific to the datasets they are trained with [6], [9]. More recently, there is a focus on addressing generalizability concerns in these models [9], [10], [11], [12], [13], [14]. For example, [9] explores the impact of generalization of 15 transformer models on 5 fake news datasets. The results show there is a generalizability gap in fake news detection: a fine-tuned model trained on one fake news dataset performs poorly on other unseen, but related, fake news datasets.

Generalization Study. In this paper, we study the generalizability and fine-tuning tradeoff and present our findings for furthering the research interest. We study several fake news detection architectures across 9 fake news text datasets of different modalities. We find that fine-tuned models often have reduced accuracy on any unseen dataset. However, when paired with a reject option to abstain from low-confidence predictions, fine-tuned models perform significantly better. These abstention results can then be labeled with active learning, crowdsourcing, weak label integration, or a variety of other methods present in literature.

KMeans-Proxy. Through observations on our generalizability results, we present a simple 'reject option' [15], [16], [17] for fake news detectors, called KMeans-Proxy. KMeans-Proxy is based on KMeans clustering, and is inspired by research into proxy losses [18], [19] and foundation models [20], [21], [22]. It is written as a PyTorch layer and requires only a few lines of code to implement for most feature extractors. We show in our results that KMeans-Proxy improves generalization on fake news datasets by 0.1 to 0.2 f1 points across several experiments.

Contributions. In summary, our contributions are:

1. Extensive set of experiments across 9 fake news datasets on the generalizability/fine-tuning trade off.
2. KMeans-Proxy, a simple reject-option for feature extractors. KMeans-Proxy uses cluster proxies from ProxyNCA to estimate embedding cluster centers of the training data. During prediction, KMeans-Proxy provides a reject option based on label difference between sample prediction and nearest training data cluster center.

We will release our code for running experiments and for KMeans-Proxy.

IEEE 8th International Conference on Collaboration and Internet Computing (CIC) | 978-1-6654-7300-2/22/$31.00 ©2022 IEEE | DOI

978-1-6654-7300-2/22/$31.00 ©2022
IEEE
DOI 10.1109/CIC56439.2022.00022

II. Related Work

A. Generalizability and Fine-Tuning

Since 2002, there are increasing numbers of Covid-19 fake news datasets and associated models for these datasets [4], [8]. Recently, there is increasing interest in gauging the effectiveness of each of these fine-tuned models on related, but unseen datasets [9]. The authors of [9] conduct a generalization study over 15 model architectures over 5 datasets, and find that fine-tuning offers little advantage in classification accuracy. There is also an abundance of research in unsupervised domain adaptation to recover accuracy under changing domains or concept drift [23], [24], [25], [26].

B. Concept Drift

Concept drift occurs when testing or prediction data exhibits distribution shift [24], either in the data domain, or in the label domain [26]. Data domain shift can include introduction of new vocabularies, disappearance of existing words, and word polysemy [27]. Label domain shift occurs when the label space itself changes for the same type of data [25], [28]. For example, when new types of misinformation are detected, then the boundary between misinformation and true information must be adjusted [28]. We show label shift in Figure 1, where subsets of true and fake news occupy the same embedding space across datasets due to fine-grained differences.

C. Reject Options

One drawback of classification models is that they provide a prediction for every data point [15], regardless of confidence. Reject options perform external or internal diagnosing. This can help detect either low confidence due to low coverage or divergence from training data distributions due to concept drift. Several approaches are covered in a recent survey [15]. We present a reject option that uses recent findings in [21] and [29] on the topology of the embedding space: (i) find that local smoothness of the label space is indicative of local accuracy and coverage [29], and (ii) local label shift, where nearby samples have different labels, is a good predictor of local smoothness [21]. Our reject option, described in Section III-C is a clustering approach that calculates cluster centers in the feature extractor embedding space for the training data. Then, during prediction, a model can provide a prediction as well the label for the nearest training data cluster center. Flipped, or different, labels can indicate reduced local smoothness, confidence, and coverage, leading to a reject decision.

D. Motivation

It is well known that fine-tuned models suffer performance degradation over time due to data domain shift [28], [30], [31]. Usually, this performance degradation is detected, and a new model is trained on new labeled data. Recently, the velocity and size of new data makes obtaining labeled data quickly and at scale, very expensive [32]. Updating models during data domain shift requires relying on weak labels, authoritative sources, and hierarchical models [32], [33], [34]. In such cases, a team of prediction models is pruned and updated with new training data [28]. However, we still need predictions in the period when data domain shift is occurring, and new models have not been trained. Our work, as well as recent research in generalizability [9], weak labeling [33], foundation models [21], and rapid fake news detection [32] falls in this period. Our generalizability experiments in Section III show that finetuned models, while having lower performance on unseen data, do have better accuracy on some subsets. Our KMeans-Proxy solution finds these subsets where fine-tuned models have higher accuracy.

TABLE I: Datasets used for our experiments. If splits were not available, we used a random class-balanced split. For Tweet datasets, sample counts are after rehydration, which removed some samples due to missing tweets.

Dataset	Training	Testing	Type
k title [35]	31k	9K	Article Titles
coaid [4]	5K	1K	News summary
c19 text [36]	2.5K	0.5K	Articles
cq [37]	12.5K	2K	Tweets
miscov [38]	4K	0.6K	Headlines
k text [35]	31k	9K	Articles
rumor [39]	4.5K	1K	Social Posts
cov fn [13]	4K	2K	Tweets
c19 title [36]	2.5K	0.5K	Article Titles

III. Generalization Experiments

We transformer-based text feature extractors for fake news classification generalizability with several experiments. We cover the datasets, architectures, and experiments below.

Datasets. We used 9 fake news datasets, consisting of blog articles, news headlines, news content, tweets, social media posts, and article headlines. We have described our dataset below in Table I.

Where possible, we have used the provided training and validation sets; otherwise, we performed a random, classbalanced 70-30 split for training and testing. For [37] and [13] datasets, we performed tweet rehydration, which removed some samples due to missing tweets. We show example of label shift due to label overlap in Figure 1. Here, samples from each dataset are passed through a pre-trained BERT classifier. The BERT embeddings are then reduced to 50 components with PCA then to 2 components with tSNE. There are several regions with label overlaps, where samples with positive and negative labels occupy similar spaces.

Architectures. We use BERT and AlBERT architectures for our experiments [40], [41]. Each transformer architecture converts input tokens to a classification feature vector. We used pretrained architectures as starting points; our selections include the BERT [40], AlBERT [41], and COVID-Twitter-BERT [42].

Experiments. We performed 3 experiments to evaluate generalizability of covid fake news detectors. In each case, our starting point is a pre-trained foundation model,

described in previous section. We then conduct the following experiments:

1. Static-Backbone. We freeze the pre-trained feature extractor backbone, and train only the classifier head. This is analogous to using a static foundation model.
2. Static-Embedding. We fine-tune the transformer part of the pre-trained feature extractor along with the classifier head together with a single optimizer, and freeze the embedding module
3. Fine-Tuned Backbone. We fine-tune the entire feature extractor backbone along with the classifier head.

Evaluation. We average results across multiple runs of each transformer architecture. To show results in limited space, we have provided complete evaluation results for backbones using Covid-Twitter-BERT. To test generalizability, we train each model on a single dataset, and evaluate on the test-sets of the remaining, unseen datasets as well as its own testing dataset. Our results are presented as a confusion matrix. All approaches are trained for 5 epochs with an AdamW optimizer, with a learning rate of 1e-4, with a batch size of 64.

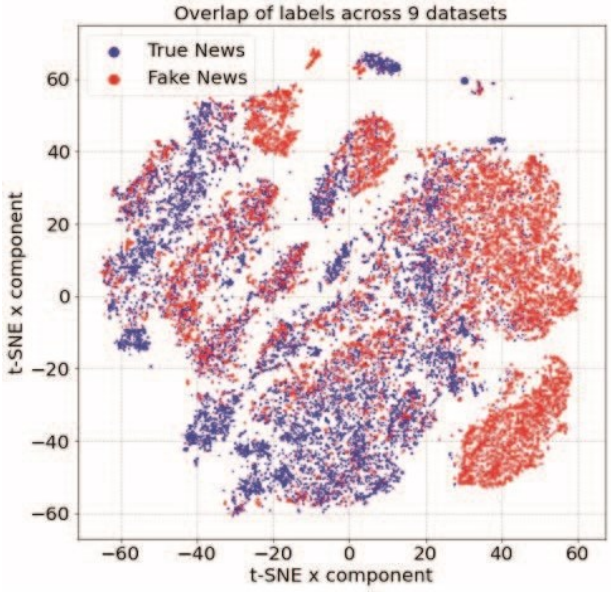

Fig. 1: Overlap of label embeddings: We show label embeddings for all 9 datasets here. There are several regions where true news (blue) overlap with fake news (red) across different datasets. This can make models for one dataset perform worse on unseen, but related text-based fake news datasets.

Static Backbone

	cov_fn	k_short	coaid	cq	k_long	rumor	c19_text	miscov	c19_title
cov_fn	**0.79**	0.45	0.57	0.48	0.48	0.76	0.39	0.53	0.61
k_short	0.49	**0.90**	0.49	0.52	0.79	0.49	0.51	0.39	0.77
coaid	0.18	0.41	**0.85**	0.18	0.27	0.17	0.83	0.81	0.31
cq	0.47	0.58	0.42	**0.59**	0.59	0.47	0.44	0.42	0.53
k_long	0.44	0.54	0.47	0.53	**0.91**	0.59	0.52	0.42	0.66
rumor	0.69	0.54	0.34	0.66	0.65	**0.70**	0.33	0.36	0.65
c19_text	0.16	0.68	0.43	0.56	0.66	0.17	**0.88**	0.71	0.46
miscov	0.51	0.45	0.54	0.44	0.45	0.52	0.50	**0.55**	0.45
c19_title	0.72	0.63	0.49	0.57	0.68	0.74	0.35	0.28	**0.85**

Fig. 2: Confusion Matrix for Static Backbone

A. Generalizability Results

We show generalizability results using the COVID-Twitter backbone in for static-backbone training in Figure 2, staticembedding training in Figure 3, and fine-tuned backbone training in Figure 4.

Static backbone vs Fine-Tuning. The confusion matrices show that upon fine-tuning, each model increases accuracy on its corresponding test dataset. For example, accuracy on 'k _short' increases from 0.90 to 0.97 between the static backbone and the fine-tuned backbone. However, this finetuning comes at the cost of generalization in some cases: across

Static Embedding

	cov_fn	k_short	coaid	cq	k_long	rumor	c19_text	miscov	c19_title
cov_fn	**0.88**	0.41	0.44	0.48	0.47	0.48	0.44	0.52	0.57
k_short	0.52	**0.96**	0.57	0.52	0.87	0.52	0.59	0.48	0.49
coaid	0.21	0.77	**0.95**	0.17	0.33	0.17	0.76	0.83	0.91
cq	0.50	0.56	0.59	**0.59**	0.59	0.59	0.51	0.41	0.45
k_long	0.53	0.76	0.51	0.52	**0.97**	0.52	0.60	0.48	0.49
rumor	0.69	0.45	0.31	0.66	0.65	**0.66**	0.47	0.34	0.48
c19_text	0.51	0.76	0.64	0.57	0.64	0.57	**0.97**	0.43	0.68
miscov	0.49	0.47	0.45	0.45	0.45	0.45	0.48	**0.55**	0.50
c19_title	0.59	0.55	0.75	0.57	0.67	0.57	0.58	0.43	**0.98**

Fig. 3: Confusion Matrix for Static-Embedding Backbone

Fine-Tuned Backbone

	cov_fn	k_short	coaid	cq	k_long	rumor	c19_text	miscov	c19_title
cov_fn	**0.96**	0.43	0.51	0.44	0.47	0.75	0.40	0.52	0.64
k_short	0.52	**0.97**	0.56	0.57	0.86	0.52	0.50	0.48	0.52
coaid	0.20	0.53	**0.97**	0.48	0.43	0.40	0.76	0.83	0.84
cq	0.57	0.57	0.55	**0.54**	0.59	0.42	0.55	0.41	0.46
k_long	0.53	0.71	0.58	0.40	**0.98**	0.43	0.54	0.48	0.54
rumor	0.67	0.53	0.55	0.52	0.62	**0.83**	0.48	0.34	0.60
c19_text	0.57	0.72	0.79	0.44	0.67	0.58	**0.98**	0.43	0.43
miscov	0.46	0.45	0.47	0.53	0.44	0.54	0.46	**0.55**	0.54
c19_title	0.57	0.55	0.77	0.55	0.62	0.81	0.62	0.43	**0.95**

Fig. 4: Confusion Matrix for FIne-Tuned Backbone

several datasets, model accuracy on unseen data suffers in the fine-tuned backbone experiments. For example, a model trained on 'cov fn' achieves f1 of 0.72 when tested on 'c19 _title' in the static-backbone experiment in Figure 2. On the finetuned experiment, the same trained model achieves f1 of 0.57, approximately a 20% drop. SImilarly, a model trained on 'miscov' and tested on 'c19 text' achieves f1 of 0.71 with static backbone, versus f1 of 0.43 with fine-tuned backbone. This indicates once a model is fine-tuned on a specific covid dataset, it loses some generalization information compared to the staticbackbone version. However, this is not consistent. In some cases, generalization accuracy increases: 'rumor' performs better on 'cov _fn' after fine-tuning, with f1 of 0.52 on the static backbone, versus an f1 of 0.75 after fine-tuning. Furthermore, 'rumor' achieves f1 of 0.67 when tested on 'c19 _title' on the static backbone, and f1 of 0.81

on the fine-tuned backbone (conversely, it performs *worse* on 'coaid', with f1 dropping from 0.83 to 0.40).

Static Embedding. A middle ground between complete finetuning and a fully static foundation feature extractor is to freeze the embedding layer and fine-tune the transformer layer of the backbone. Recent work finds freezing the embedding layer during training can reduce computation costs while achieving 90% of the accuracy of a fully fine-tuned model [43], [44], [45], [46], [47]. We find similar results, where freezing the embedding layer achieves accuracy similar to the corresponding fine-tuned model on the testing dataset, shown in Figure 3 However, on unseen data, accuracy drop has higher variance. It is not immediately clear what impact embedding freezing has on unseen data accuracy. For this, we must explore the actual overlap between datasets.

B. Data Overlap and Accuracy

We have seen that there is label overlap between datasets in Figure 1. We have also seen accuracy variances on unseen data: rather than a linear drop across unseen data, some models perform better and some perform worse after fine-tuning. These can be explained by directly measuring dataset overlap.

O-Metric. To compute overlap, we use the O-metric to calculate point-proximity overlap from [48]. The O-metric computes overlap between 2 sets of points in n-dimensional space using a distance-metric. We find overlap as follows: given two datasets A and B, we compute the fraction of points in each dataset where the nearest neighbor is not from the same dataset. So, for each point $x \in A$, first we obtain:

$$O_A(x) = w_A(x)/b_A(x) \text{ where } w_A() \text{ is the}$$

distance to the nearest point to x in A,

and $b_A()$ is the distance to the nearest point to x in B. Then we can compute the ratio $p_A = |O_A > 1|/|A|$ to find overlap of B in A. p_A is bounded in $[0,1]$; as p_A approaches 1, this indicates most points in A are closer to a point in B than in A. The O-metric is bidirectional in computing overlap and includes both p_A and p_B:

$$O = \frac{p_A + 2p_B}{}$$

Since we are interested in evaluating generalization, where we want to see only the overlap of unseen data on training data, we use a directional O-metric. That is, we let $O = p_A$ for the final overlap value in a context where A is the training dataset and B is the unseen dataset. We compute the overlap value between each dataset pair using cosine similarity on the embeddings of each data point. So, for each model, we compute embeddings of every sample across all 9 datasets, then compute the directional O-metric overlap of each dataset on the model's training dataset [ALGO???].

Accuracy and Overlap.

Generalization and Overlap. Clearly, higher overlap between evaluation data and training data is indicative of accuracy. During testing, however, it may be difficult to evaluate this overlap due to computational constraints [21]. Further, evaluation data changes continuously, so the overlap may itself change due to concept drift. Recent research has shown the importance of exploring a model's feature space to identify embedding clusters [21], [29]. These embedding clusters signify regions of the data space a model has captured. Metrics such as probabilistic Lipschitzness show that accuracy on embedding clusters can be bounded using the smoothness, or gradient, in the embedding space [29]. Further, LIGER [21] shows that nondeterministic label regions, i.e. where labels overlap, indicate non-smoothness. We extend these findings to present KMeans-Proxy - a plug-and-play pytorch layer.

C. KMeans-Proxy

Intuitively, if we can store the coverage of a model's embedding space, then for any sample point, we can check if it falls inside the coverage. Further, we can also check if a model's prediction on the sample matches the prediction for the coverage. We can pair this with a coverage radius, e.g. by computing r that constitutes coverage of all points in a cluster that are 1 standard deviation away from the cluster center with respect to a distance metric, such as the l2 norm. Then, if the predictions do not match or a point falls outside the single standard deviation coverage radius, this is a strong abstention/reject signal.

Implementation. We can capture the coverage of the embedding space by using embedding proxies. Proxies are common in cluster and proxy NCA losses [18], [19]. We adapt them as the KMeans cluster centroids by acting as proxies for the cluster centers. This allows our approach to extend to online or continuous learning domains as well. The findings in [21] suggest increased partitioning of the embedding space can yield better local region coverage. So, KMeans-Proxy is initialized with 2 parameters: the number of classes c, and a proxy factor k. Then, we then obtain $k \cdot c$ centers, with k proxies for each class, so that each cluster is a smaller, more representative local region.

```
self.proxy = KMeansProxy(proxy_factor=
    4, classes=3, dimensions=768)
```

During model training, KMeans-Proxy performs minibatch online KMeans clustering to obtain the embedding space proxies for cluster centers. Online clustering converges asymptotically, per [49].

```
def forward(x):  # Proxy forward function
    if self.training:
        self.update_proxies(x)
        return x, None, None
    return x, self.nearest_proxy(x), self
        .nearest_proxy_label(x)
```

During prediction, a model using KMeans-Proxy can predection, as well as nearest proxy label and nearest proxy. A meta abstention policy can review for label flipping, or coverage radius.

```
def forward(x):          # Model forward function
    x = self.featureextractor(x) x, proxy,
    proxy label = self.proxy(x
        )
    label =          self.classifier(x)
    return    label,    proxy,    proxy    label
```

D. KMeans-Proxy Results

We now show results from using KMeans-Proxy as a reject option in Table II and Table III. With rejection, we can increase generalization accuracy on unseen data. This is a faster approach than domain adaptation, since proxies are updated during training. Further, it is a plug-and-play solution, allowing for faster iteration on overall model design.

With KMeans-Proxy, we are able to improve generalization performance across the board. Here, we compare models trained on 'coaid' and 'rumor' in Table II and Table III,

TABLE II: Generalization improvement with KMP for model trained on 'coaid'.

Trained on 'coaid'				
Testing Dataset	Approach			
	SB	SE	FT	FT+KMP
cov fn	0.57	0.44	0.51	0.53
k short	0.49	0.57	0.56	0.57
coaid	0.85	0.95	0.97	0.98
cq	0.42	0.59	0.55	0.57
k long	0.47	0.51	0.58	0.55
rumor	0.34	0.31	0.55	0.68
c19 text	0.43	0.64	0.79	0.94
miscov	0.54	0.45	0.47	0.57
c19 title	0.49	0.75	0.77	0.90

TABLE III: Generalization improvement with KMP for model trained on 'rumor'

Trained on 'rumor'				
Testing Dataset	Approach			
	SB	SE	FT	FT+KMP
cov fn	0.76	0.48	0.75	0.77
k short	0.49	0.52	0.52	0.54
coaid	0.17	0.17	0.40	0.76
cq	0.47	0.59	0.42	0.58
k long	0.59	0.52	0.43	0.53
rumor	0.70	0.66	0.83	0.86
c19 text	0.17	0.57	0.58	0.73
miscov	0.52	0.45	0.54	0.54
c19 title	0.74	0.57	0.81	0.76

respectively. Models are compared across static backbone (SB), static embedding (SE), fine-tuned (FT), and fine-tuned with KMeans-Proxy (FT+KMP).

For models trained on 'coaid' (in Table II) and tested on all datasets, incorporating KMeans-Proxy improves generalization performance. In each case, FT+KMP is either the best or the runner-up model by at most 0.05 f1 f1 points. We see similar result for models trained on 'rumor' in Table III, where KMeans-Proxy is either the best model or runner up for every testing dataset.

Choice of Proxy Factor. Increasing the proxy factor leads to better generalization performance. Table IV shows performance of a model trained on 'c19 text' that has poor generalization without KMeans-Proxy (see Figure 4). As we increase the proxy factor, we gain better generalization across testing datasets. We test with different proxy factors and compare generalization performance of each model. We find that increasing the proxy factor leads to small, but measurable increase in accuracy.

TABLE IV: Impact of changing proxy factor: increasing the proxy factor increases accuracy, since more proxies allow tighter bounds on local coverage estimates.

Testing Dataset	Trained on c19 text					
	FT	k=1	k=2	k=3	k=5	k=10
cov fn	0.44	0.50	0.54	0.58	0.58	0.58
k short	0.59	0.70	0.73	0.73	0.73	0.75
coaid	0.76	0.88	0.84	0.89	0.89	0.90
cq	0.51	0.56	0.58	0.58	0.60	0.63
k long	0.60	0.70	0.72	0.73	0.73	0.73
rumor	0.47	0.45	0.51	0.58	0.58	0.67
c19 text	0.97	0.98	0.99	0.99	0.99	0.99
miscov	0.48	0.45	0.44	0.53	0.58	0.58
c19 title	0.58	0.61	0.64	0.69	0.68	0.73

E. Discussion

There are several observations we can make from our generalization studies and KMeans-Proxy experiments. Generalization. For fake news detection, fine-tuned models must be used carefully to take advntage of learned parameters. As we showed in the confusion matrices, fine-tuning improves performance only on subsets of unseen data. These subsets are regions of the data space where the unseen data overlaps with training data. On completely new regions of the data space, fine-tuned models make mistakes. These mistakes are because of label overlap.

We must make a distinction between label and data overlap. Data overlap means a model has coverage on the unseen data, and can make predictions with higher confidence. Label coverage, as we showed in Fig, indicates where different labels occur close to each other in the embedding space. Both can coincide: unseen data points can have both data *and* label overlap. For these points, fine-tuned models that have better captured a local region with training data are better poised to provide high-confidence labels.

KMeans-Proxy. KMeans-Proxy allows us to identify these regions. With KMeans-Proxy, we partition the data space into clusters representing model coverage and labels. Our inclusion of the proxy factor k, where we create k clusters for each class label, allows fine-grained partitioning of the embedding space. This means we can better capture local characteristics of the embedding space [21]. In our experiments, we focus on 2 such characteristics: (i) whether the label for an unseen point matches the label for nearest proxy, and (ii) whether this unseen point is within one standard-deviation radius of the proxy. In our experiments, we show that using these provides improvements in generalizing to unseen data points.

Clearly, there is significant progress to be made in capturing local characteristics. For example, when using an ensemble of fine-tuned models, local smoothness [29], [21] can be computed for each non-abstaining model to rank them on coverage. There may also be advantages in using dynamic proxy allocation. If prior class balance is known, then we could use a class-specific proxy factor.

IV. Conclusion

In this paper, we have presented generalizability experiments and KMeans-Proxy. We perform generalization studies across 9 fake news datasets using several transformer-based fake news detector models. Our generalizability experiments show that fine-tuned models generalize well to unseen data when there is overlap between unseen and training data. On unseen data that does not overlap, fine-tuned models make mistakes due to poor coverage, label flipping, and concept drift.

Using our observations and recent research into local embedding regions, we develop and present KMeans-Proxy, a simple online KMeans clusterer paired with a proxy factor. With KMeans-Proxy, we partition the embedding space into local regions and use local characteristics to create a reject option for models. We show that KMeans-Proxy improves generalization accuracy for fine-tuned models across all 9 fake news datasets. We welcome future research in this area to better explore the generalizability and fine-tuning tradeoff.

Acknowledgment

This research has been partially funded by National Science Foundation by CISE/CNS (1550379, 2026945, 2039653), SaTC (1564097), SBE/HNDS (2024320) programs, and gifts, grants, or contracts from Fujitsu, HP, and Georgia Tech Foundation through the John P. Imlay, Jr. Chair endowment. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding agencies and companies mentioned above.

References

[1] A. M. Enders, J. E. Uscinski, C. Klofstad, and J. Stoler, "The different forms of covid-19 misinformation and their consequences," *The Harvard Kennedy School Misinformation Review*, 2020.

[2] ——, "The different forms of covid-19 misinformation and their consequences," *The Harvard Kennedy School Misinformation Review*, 2020.

[3] E. K. Quinn, S. S. Fazel, and C. E. Peters, "The instagram infodemic: cobranding of conspiracy theories, coronavirus disease 2019 and authorityquestioning beliefs," *Cyberpsychology, Behavior, and Social Networking*, vol. 24, no. 8, pp. 573–577, 2021.

[4] L. Cui and D. Lee, "Coaid: Covid-19 healthcare misinformation dataset," *arXiv preprint arXiv:2006.00885*, 2020.

[5] M. Weinzierl, S. Hopfer, and S. M. Harabagiu, "Misinformation adoption or rejection in the era of covid-19," in *Proceedings of the International AAAI Conference on Web and Social Media (ICWSM), AAAI Press*, 2021.

[6] T. Hossain, R. L. Logan IV, A. Ugarte, Y. Matsubara, S. Young, and S. Singh, "Covidlies: Detecting covid-19 misinformation on social media," 2020.

[7] J. C. M. Serrano, O. Papakyriakopoulos, and S. Hegelich, "Nlp-based feature extraction for the detection of covid-19 misinformation videos on youtube," in *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, 2020.

[8] Z. Kou, L. Shang, Y. Zhang, C. Youn, and D. Wang, "Fakesens: A social sensing approach to covid-19 misinformation detection on social media," in *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2021, pp. 140–147.

[9] J. P. Wahle, N. Ashok, T. Ruas, N. Meuschke, T. Ghosal, and B. Gipp, "Testing the generalization of neural language models for covid-19 misinformation detection," in *International Conference on Information*. Springer, 2022, pp. 381–392.

[10] Y. Bang, E. Ishii, S. Cahyawijaya, Z. Ji, and P. Fung, "Model generalization on covid-19 fake news detection," in *International Workshop on Combating Hostile Posts in Regional Languages during Emergency Situation*. Springer, 2021, pp. 128–140.

[11] R. K. Kaliyar, A. Goswami, and P. Narang, "Mcnnet: generalizing fake news detection with a multichannel convolutional neural network using a novel covid-19 dataset," in *8th ACM IKDD CODS and 26th COMAD*, 2021, pp. 437–437.

[12] R. Walambe, A. Srivastava, B. Yagnik, M. Hasan, Z. Saiyed, G. Joshi, and K. Kotecha, "Explainable misinformation detection across multiple social media platforms," *arXiv preprint arXiv:2203.11724*, 2022.

[13] S. D. Das, A. Basak, and S. Dutta, "A heuristic-driven uncertainty based ensemble framework for fake news detection in tweets and news articles," *Neurocomputing*, 2021.

[14] B. Chen, B. Chen, D. Gao, Q. Chen, C. Huo, X. Meng, W. Ren, and Y. Zhou, "Transformer-based language model fine-tuning methods for covid-19 fake news detection," in *International Workshop on Combating Hostile Posts in Regional Languages during Emergency Situation*. Springer, 2021, pp. 83–92.

[15] K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, and J. Davis, "Machine learning with a reject option: A survey," *arXiv preprint arXiv:2107.11277*, 2021.

[16] J. Brinkrolf and B. Hammer, "Interpretable machine learning with reject option," *at-Automatisierungstechnik*, vol. 66, no. 4, pp. 283–290, 2018.

[17] Y. Geifman and R. El-Yaniv, "Selectivenet: A deep neural network with an integrated reject option," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2151–2159.

[18] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 360–368. [19] E. Wern Teh, T. DeVries, and G. W. Taylor, "Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis," *arXiv e-prints*, pp. arXiv–2004, 2020.

[20] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[21] M. F. Chen, D. Y. Fu, D. Adila, M. Zhang, F. Sala, K. Fatahalian, and C. Re, "Shoring up the foundations: Fusing model embeddings and weak´ supervision," *arXiv preprint arXiv:2203.13270*, 2022.

[22] L. Orr, K. Goel, and C. Re, "Data management opportunities for´ foundation models," in *12th Annual Conference on Innovative Data Systems Research*, 2021.

[23] A. Suprem, J. Arulraj, C. Pu, and J. Ferreira, "Odin: automated drift detection and recovery in video analytics," *arXiv preprint arXiv:2009.05440*, 2020.

[24] J. Gama, I. Zliobaitˇ e, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A´ survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[25] W. M. Kouw and M. Loog, "An introduction to domain adaptation and transfer learning," *arXiv preprint arXiv:1812.11806*, 2018.

[26] I. Zliobaitˇ e, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

[27] A. Suprem, A. Musaev, and C. Pu, "Concept drift adaptive physical event detection for social media streams," in *World Congress on Services*. Springer, 2019, pp. 92–105.

[28] C. Pu, A. Suprem, R. A. Lima, A. Musaev, D. Wang, D. Irani, S. Webb, and J. E. Ferreira, "Beyond artificial reality: Finding and monitoring live events from social sensors," *ACM Transactions on Internet Technology (TOIT)*, vol. 20, no. 1, pp. 1–21, 2020.

[29] R. Urner and S. Ben-David, "Probabilistic lipschitzness a niceness assumption for deterministic labels," in *Learning Faster from Easy DataWorkshop NIPS*, vol. 2, 2013, p. 1.

[30] S. Yang, M. Santillana, and S. C. Kou, "Accurate estimation of influenza epidemics using google search data via argo," *Proceedings of the National Academy of Sciences*, vol. 112, no. 47, pp. 14473–14478, 2015.

[31] D. Lazer, R. Kennedy, G. King, and A. Vespignani, "Google flu trends still appears sick: An evaluation of the 2013-2014 flu season," *Available at SSRN 2408560*, 2014.

[32] Y. Li, K. Lee, N. Kordzadeh, B. Faber, C. Fiddes, E. Chen, and K. Shu, "Multi-source domain adaptation with weak supervision for early fake news detection," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 668–676.

[33] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Re, "Snorkel:´ Rapid training data creation with weak supervision," in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 11, no. 3. NIH Public Access, 2017, p. 269.

[34] S. Ruhling Cachay, B. Boecking, and A. Dubrawski, "End-to-end weak¨ supervision," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[35] S. Patel, "Covid-19 Fake News Dataset (Kaggle)," 2021. [Online]. Available: https://www.kaggle.com/code/therealsampat/fake-news-detection/

[36] I. Agarwal, "Covid 19 Fake News Dataset," 6 2020. [Online]. Available: https://figshare.com/articles/dataset/COVID19FN Dataset csv/12489293

[37] E. C. Mutlu, T. Oghaz, J. Jasser, E. Tutunculer, A. Rajabi, A. Tayebi, O. Ozmen, and I. Garibay, "A stance data set on polarized conversations on twitter about the efficacy of hydroxychloroquine as a treatment for covid-19," *Data in Brief*, p. 106401, 2020.

[38] S. A. Memon and K. M. Carley, "Characterizing covid-19 misinformation communities using a novel twitter dataset," *arXiv preprint arXiv:2008.00791*, 2020.

[39] M. Cheng, S. Wang, X. Yan, T. Yang, W. Wang, Z. Huang, X. Xiao, S. Nazarian, and P. Bogdan, "A covid-19 rumor dataset," *Frontiers in Psychology*, vol. 12, p. 1566, 2021.

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[41] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[42] M. Muller, M. Salath¨ e, and P. E. Kummervold, "Covid-twitter-bert: A´ natural language processing model to analyse covid-19 content on twitter," *arXiv preprint arXiv:2005.07503*, 2020.

[43] J. Lee, R. Tang, and J. Lin, "What would elsa do? freezing layers during transformer fine-tuning," 2019. [Online]. Available: https://arxiv.org/abs/1911.03090

[44] E. Hulburd, "Exploring bert parameter efficiency on the stanford question answering dataset v2.0," 2020. [Online]. Available: https://arxiv.org/abs/2002.10670

[45] T. Moon, P. Awasthy, J. Ni, and R. Florian, "Towards lingua franca named entity recognition with bert," 2019. [Online]. Available: https://arxiv.org/abs/1912.01389

[46] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Lightmbert: A simple yet effective method for multilingual bert distillation," 2021. [Online]. Available: https://arxiv.org/abs/2103.06418

[47] A. Merchant, E. Rahimtoroghi, E. Pavlick, and I. Tenney, "What happens to bert embeddings during fine-tuning?" 2020. [Online]. Available: https://arxiv.org/abs/2004.14448

[48] M. Cardillo and D. L. Warren, "Analysing patterns of spatial and niche overlap among species at multiple resolutions," *Global Ecology and Biogeography*, vol. 25, no. 8, pp. 951–963, 2016.

[49] G. So, G. Mahajan, and S. Dasgupta, "Convergence of online k-means," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 8534–8569.