FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 49 (2024) No. 4
ISSN 0867-6356
DOI: 10.2478/fcds-2024-0020 e-ISSN 2300-3405

sciendo

Leveraging Unseen Features along with their PLM-based Representation to Handle Negative Covariate Shift Problem in Text Classification

Nesar Ahmad Wasi, Muhammad Abulaish *

Abstract. This paper presents a novel approach to address the problem of negative covariate shift by using unseen features. Covariate shift occurs when there is a drift between the data observed during the training and testing phase of a machine learning model. Covariate shift typically transpires in the negative class as a consequence of the swift evolution of topics discussed therein, which is driven by the characteristics of online social media. Because there is a shift in data, it signals that the data is changing, and it includes features that the trained model did not see during the training phase. We refer to such features as *unseen features*. To the best of our knowledge, we are the first to use unseen features to address negative covariate shift problem. The proposed approach is compared to three baselines and one state-of-the-art method. The experimental results obtained from a multi-domain sentiment dataset show that the proposed approach outperforms the baselines and state-of-the-art approaches by a significant margin in terms of various performance evaluation metrics.

Keywords: Machine Learning, Covariate Shift, Unseen Features, Text Classification, Word Embedding

1. Introduction

In conventional machine learning, it is presumed that the training and testing data samples are drawn from the same data distribution. However, in practice, and especially in online social media analytics, there is a shift in data due to the diversity of topics discussed. In online social media analytics, the user usually has sufficient knowledge of the topic or product it is looking for, which makes the distribution related to that class well-represented and, in turn, mitigates the issue of covariate shift. However, due to insufficient knowledge or diversity in negative class data, the covariate shift problem happens in negative class [7].

*Department of Computer Science, South Asian University, New Delhi, India, nesarahmadwasi17@gmail.com, abulaish@sau.ac.in

In literature, to handle the problem of covariate shift, a bias of the training data based on testing data is computed. Further training data is weighted based on the computed bias to train a classifier. Such approaches need the testing data to be present during the training phase, which is a significant disadvantage as, in real life, we usually do not have access to the testing data at the time of training. In negative covariate shift, as the shift happens in one of the classes, some approaches drop the class of data affected by covariate shift; further, a one-class classifier is used to train the classifier. Based on the literature and our experimental results, it is suggested that one-class classifiers perform poorly in such cases. Compared to the above strategies, in our proposed approach, we do not need the testing data to be present during training data. Additionally, we also do not drop the negative class. These are the two advantages of the proposed approach over literature.

During the deployment phase, when the machine learning model faces a data sample that is comprised of features that the model may not have seen during the training phase, such samples are arbitrarily assigned to either of the binary classes. We term features unknown to the trained classifier as *unseen features*. The *unseen features* can not be handled immediately, as such features do not carry specific knowledge associated with them. In order to handle such features, we train the classifier in a way that is able to handle the unseen features in the deployment phase. For this purpose, we use a pre-trained language model that helps to deal with *unseen features*. Since a language model is trained on humongous data, similar features are closely situated. However, those features that are dissimilar are situated apart from each other. We use this analogy present in language models to handle *unseen features*.

We have performed extensive experiments on a multi-domain sentiment dataset to evaluate the effectiveness of utilizing unseen features to deal with the problem of negative covariate shift. The proposed approach is compared with three baselines and one state-of-the-art approach. The experimental results suggest the effectiveness of the proposed approach over baselines and state-of-the-art approaches. In short, the key contributions of the proposed work can be summarized as follows:

- Introducing a word embedding-based approach for dealing with unseen features encountered during the deployment phase of a machine learning model.
- Presenting an unseen features-enriched method to improving a machine learning model to deal with the negative covariate shift problem.
- Presenting an extensive empirical evaluation and comparative analysis on a multi-domain sentiment dataset to evaluate the effectiveness of the proposed approach.

For reproducibility, the source codes of the proposed approach and all baselines approaches are uploaded at GitHub[1] repository. The rest of the paper is organized as follows. Section 2, presents a concise summary of the related works on negative covariate shift. Section 3, explains the pre-trained language model-based approach to handle the problem of negative covariate shift. Section 4, discusses dataset, experimental setup, evaluation metrics, and comparative analysis results. Finally, Section 5 concludes the paper and outlines the future directions of research.

[1] https://github.com/nesar-a-wasi/UFB-L

2. Related Works

The assumption that the train and test data must be drawn from the same distribution is often violated in real-life scenarios due to the non-stationary environments. Covariate shift is a type of the sample selection bias problem [9]. In [22], authors proposed an approach in which the kernel density of the train and test data distribution are first computed. Further, the computed ratio is used as a weighting scheme in the log-likelihood function to improve the predictive inference. Similar to [22], [23] proposed an estimator to compute the generalization error for the square loss function. In [24], a train and test data density-independent approach is proposed to estimate the importance (i.e., the ratio of train and test data distribution) using Kullback-Leibler divergence.

In [1], an approach to find all model parameters through an optimization process is proposed that can mitigate the problem of covariate shift. For optimization, they have utilized Newton gradient descent. Further, it is used with logistic regression to handle covariate shift. In [31], the authors proposed a Bayesian approach to handle the covariate shift problem in which ensembles are utilized to perform adaptation during test time in order to approximate inference using a regularized entropy minimization. Our proposed work is different from the aforementioned approaches because, in our work, we do not utilize the train and test data ratio during model training.

The covariate shift problem is also addressed using the concept of Learning from Positive and Unlabeled data (PU learning) [30, 13]. In PU learning, the machine learning model is trained on positive and unlabeled data; however, no labeled negative data is present during the training phase. In [21], the authors proposed an approach in which they studied the task of covariate shift under the PU learning framework. In [21], it is shown that an importance-weight estimation, i.e., a ratio of train and test distributions, is necessary to address the covariate shift problem using PU data. In [8], the issue of the arbitrary shift in the positive class of PU learning is discussed under the assumption that data instances belonging to the negative class must be sampled from a single data distribution. The arbitrary shift means covariate shift in data of the positive class seen during the training and testing phase. Our proposed approach differs from PU learning due to the nature of the problem we address because we address the covariate shift problem in the negative class.

One-class classification [15, 25] is a natural fit due to the nature of the problem we address. In one-class classification, data associated with one of the classes are dropped during the training phase, and the learning happens in another class. Since text classification problems are represented in high-dimensional feature space, one-class classification approaches are unsuitable for this task [12]. The experimental results reported by [7] and our results also confirm that the one-class classification approach performs poorly.

The problem addressed in [7] is similar to the one we want to address. In [7], the document-vector is transformed from the traditional document-based representation, i.e., TF-IDF, to the centrality-based representation, which mitigates the problem of negative covariate shift. Proposed approach in this paper is different from the approach proposed by [7], as we convert the feature space to the pre-trained language model-based representation. Our approach's key novelty is utilizing unseen features during the deployment phase[27]. In [28], the concept of unseen features is utilized in a lifelong machine learning framework. We use the concept of unseen features to handle covariate shift.

3. Proposed Approach

In a machine learning approach, let x and y be the input and output distributions, respectively. The joint distribution over $x \times y$ is denoted as $p(x, y)$, called data distribution. The shift can be due to non-stationary environments or sample selection bias. In real-life scenarios, the environment in which machine learning models are deployed is rapidly changed, and the environment is constantly evolving, which results in a non-stationary environment. Due to privacy reasons, the data distribution may not be well generalized and representative of each class, which makes way for the problem of sample selection bias.

The shift in distribution can happen in either of the input and output distributions of x and y or simultaneously. Let $p_{tr}(x, y)$ denote the probability distribution of the training data over input variable x and output variable y, and $p_{te}(x, y)$ denotes the probability distribution of the testing data over input x and output y variables. If the shift is in both x and y distributions, i.e., $p_{tr}(x, y) \neq p_{te}(x, y)$, it is called a joint shift [16]. In joint shift, the testing data is shifted from training data in both input and output variables, i.e., data corresponding to those classes that the model has not seen during training is assigned to the classifier [6]. If the change is in output variables y only, i.e., $p_{tr}(y) \neq p_{te}(y)$, it is called a class-prior shift. In the class-prior shift, the $p_{tr}(x|y) = p_{te}(x|y)$ is not changed; however, due to noisy class label during training time, the class-prior shift happens [26]. If the change is in input variable x only, i.e., $p_{tr}(x) \neq p_{te}(x)$, it is called a covariate shift. The covariate shift is the most prevalent case of distribution shift, as with the change in time, the data used to describe an entity alters. If the change in input variable x is associated with the input data of the negative class, it is called a negative covariate shift [7].

In a supervised machine learning approach, when a machine learning model is deployed to the application area, the problem of covariate shift usually happens in the negative class. We usually have significant knowledge associated with the positive class. Therefore, the positive class data distribution is usually well represented, and the covariate shift problem does not happen in the positive class. However, the data domains fed to the classifier as the negative class can not be predicted fully. That is the reason behind the usual covariate shift problem in the negative class. In a supervised machine learning setting, a labeled dataset $D = \{(d_i, y_i), 1 \leq i \leq n\}$ (where each $d_i \in \mathbb{R}^l$ is a data sample, $y_i \in \{-1, 1\}$ is a class label associated to data samples, and n is the number of data samples) is passed through a classification algorithm to train a classifier, which is later on used to predict the class label of an unseen sample. To feed a document d_k to the classifier, it must first be translated to its feature vector representation, which we perform using a pre-trained language model.

3.1. Pre-trained Language Model Feature Representation

The main motive behind selecting a pre-trained language model (PLM) over conventional feature representation schemes, such as term frequency-inverse document frequency (tf-idf), bag-of-words, one-hot encoding, etc., is the issue of fixed feature vector size consisting of those features that are seen during the training phase. When the machine learning model is deployed, a fixed-size feature vector of a document is required to fed into the model for prediction purpose. In the deployment phase, when machine learning model encounters new

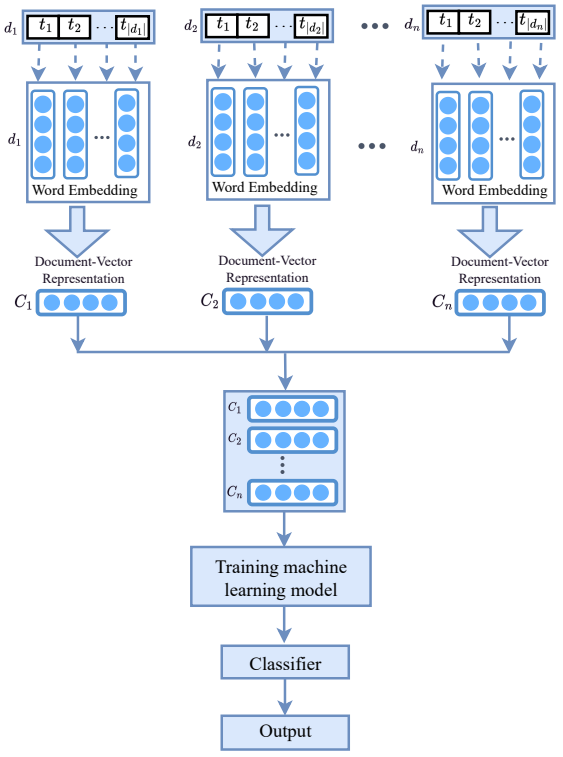

Figure 1. Work-flow of the proposed approach during training phase

(unseen) features, they can not be accommodated directly. As a result, such features are discarded during the preprocessing stage of the input document. The reason for discarding unseen features is the absence of knowledge (parametric weight) associated with them; therefore, unseen features are ineffective during deployment. Due to the aforesaid reasons, we have used word embedding associated with the PLM as the feature representation scheme. Word embedding is a dense vector representation of words trained using huge amounts of data that preserves the contextual semantics of the words. Due to this, the PLM consists of a broader vocabulary of words that makes it easy to accommodate even the words that were not seen during the training phase of a machine learning model. Word embedding is also a fixed-size vector representation of data. However, because it transforms words to its dense vector representation, it can significantly mitigate the problem of unseen features. Some notable examples of PLM's are: Word2Vec [14], GloVe [18], BERT [5], FastText [11], ELMO [19], XLNet [29], and GPT [20].

3.2. Why Utilizing Unseen Features are Important?

When a data sample x_i is assigned for classification. It first undergoes the tokenization process. The chance of correctly classifying to its label increases if it consists of more features that the model has seen during the training phase. We denote such features as *learned features*. However, if the input data sample x_i consists of more features that the model has not seen before, it is arbitrarily classified to one of the classification labels. We denote the newly seen features as *unseen features*. In the latter case, *unseen features* are discarded during the pre-processing phase, the removal of which leads to the scarcity of features, which further is responsible for the arbitrary classification of data sample x_i. Because there is a shift in input sample x_i, it is denoted that a covariate shift problem has happened. It is important to note that better handling of the *unseen features* is critical for a reliable classification model.

3.3. Training Time Document-Vector Representation

The following steps are followed to convert each document $d \in D^t$ during the training phase into their document-vector representation.

- Step 1: Each document $d \in D^t$ is going through a tokenization step to convert document d to its associated tokenized form, i.e., $d = \{t_1, t_2, ..., t_{|d|}\}$. The stop words, URLs, hashtags, and single-lettered words are removed. A library T of tokens present in the training data D^t is constructed.

- Step 2: In order to encode document d, each token $t \in T$ is looked in the word embedding library $\mathbb{L}$, and those tokens present in the word embedding library are assigned their associated word vectors. There may also be some tokens that are not present in the word embedding library, such tokens are assigned a zero vector. It can be formally presented using equation (1).

$$W_i = \begin{cases} \mathbb{E}(t)_{1\times p}, & \text{if } t \in \mathbb{L} \\ \{0\}_{1\times p}, & \text{otherwise} \end{cases} \tag{1}$$

  Dimension of each word vector W_i in equation (1) is $1 \times p$, where p is the size of the word embedding vector. The term $\mathbb{E}(t)$ is the embedding vector associated with the token t.

- Step 3: To get the final document-vector representation of the document d, the average of word vector W_i associated with tokens $\{t_1, t_2, ..., t_{|d|}\} \in d$ is computed using equation (2).

$$C = \frac{\sum_{j=1}^{|d|} W_j}{|d|} \tag{2}$$

  The computed vector C is the center of all word vectors constituting document d. In equation (2), $|d|$ denotes the size of document d in terms of number of tokens. Further, C is assigned to the document d as its document vector representation. The

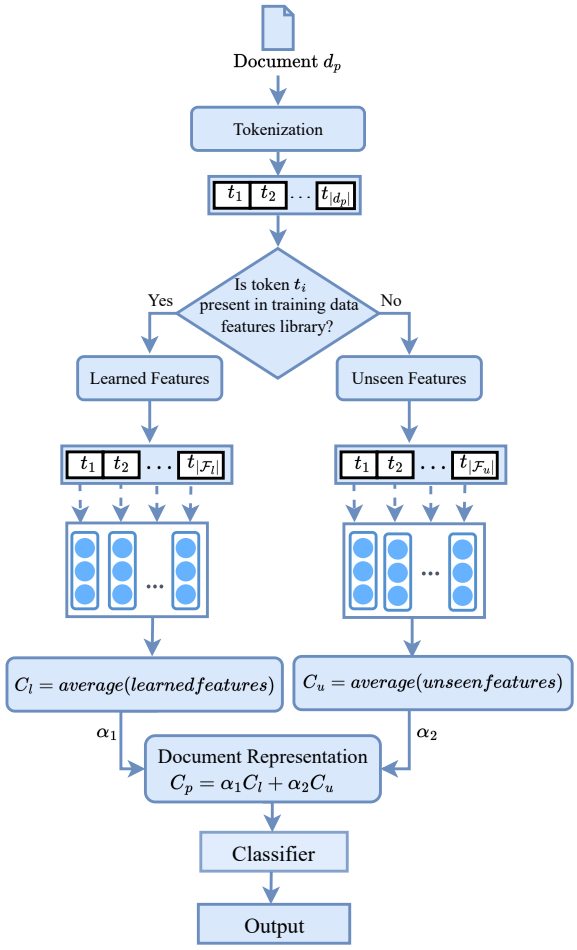

Figure 2. Work-flow of the proposed approach during deployment phase

computed center vector C is assigned to the training model to train the classification model described in equation (3).

$$\min_{w,b} \frac{||w||^2}{2} + \theta \sum_{n=1}^{N} \max(0, 1 - y_n(w^T C_n + b))^2 \tag{3}$$

Equation (3) is the squared hinge loss formulation of the Support Vector Machine (*SVM*), which is the classifier we use as a machine learning model. In equation (3), θ is a hyper-parameter that indicates the trade-off between maximizing the margin between support vectors and reducing the rate of misclassification. A higher θ means correctly classifying all training samples with complex decision surface. Using equation (3) we get the optimized $\hat{w}$ and $\hat{b}$. The complete workflow of the proposed approach during the training phase is presented in Figure 1.

3.4. Testing Time Document-Vector Representation

For a given document d_p during the testing or deployment phase, it undergoes the following process. Given document d_p, it is first converted to its tokenized form $d_p = \{t_1, t_2, ..., t_{|d_p|}\}$. Further, each token $t_i \in d_p$ is looked at in the training data vocabulary T. If found in the vocabulary T, it is added to the learned features list $\mathcal{F}_l$; otherwise, it is added to the list of unseen features $\mathcal{F}_u$. Equation (5) converts document d_p into two mutually exclusive lists $\mathcal{F}_l$ and $\mathcal{F}_u$.

$$d_p = \mathcal{F}_l \cup \mathcal{F}_u \tag{4}$$

$$d_p = \begin{cases} \mathcal{F}_l \leftarrow t_i, & \text{if } t_i \in \mathbb{T} \\ \mathcal{F}_u \leftarrow t_i, & \text{otherwise} \end{cases} \tag{5}$$

Each feature present in $\mathcal{F}_l$ is looked at in the word embedding vocabulary, and its associated word embedding is assigned. All word embedding associated with the learned features $\mathcal{F}_l$ are concatenated, and its associated average is computed. It is formally denoted using equation (6).

$$W_{\mathcal{F}_l} = \mathbb{E}(t)_{1\times p}, \quad \text{if } t \in \mathcal{F}_l \tag{6}$$

To get the document-vector representation of the learned features $\mathcal{F}_l$, the average of word vectors $W_{\mathcal{F}_l}$ associated to tokens $\{\mathcal{F}_{l_1}, \mathcal{F}_{l_2}, ..., \mathcal{F}_{l_{|\mathcal{F}_l|}}\} \in W_{\mathcal{F}_l}$ is computed using equation (7).

$$C_{\mathcal{F}_l} = \frac{\sum_{j=1}^{|\mathcal{F}_l|} W_{\mathcal{F}_l}^j}{|\mathcal{F}_l|} \tag{7}$$

To encode the unseen features $\mathcal{F}_u$ to its document-vector representation, each unseen feature $\mathcal{F}_{u_i}$ is looked at in the word embedding library; if it exists in the library, it will be assigned corresponding word embedding else, it will be assigned a zero vector. It is formally denoted using equation (8).

$$W_{\mathcal{F}_u} = \begin{cases} \mathbb{E}(t)_{1\times p}, & \text{if } t \in \mathcal{F}_u \\ \{0\}_{1\times p}, & \text{otherwise} \end{cases} \tag{8}$$

The document-vector representation of the unseen features $\mathcal{F}_u$ is denoted using equation (8).

$$C_{\mathcal{F}_u} = \frac{\sum_{j=1}^{|\mathcal{F}_u|} W_{\mathcal{F}_u}^j}{|\mathcal{F}_u|} \tag{9}$$

The resulting document-vector representation for the document d_p is the linear combination of the document-vector representations of learned features $\mathcal{F}_l$ and unseen features $\mathcal{F}_u$. It is denoted using equation (10), where $\alpha_i \geq 0$ and $\alpha_1 + \alpha_2 = 1$.

$$C_p = \alpha_1 C_{\mathcal{F}_l} + \alpha_2 C_{\mathcal{F}_u} \tag{10}$$

In equation (10), α_1 denotes the ratio of learned features by the total number of features present in the document d_p, and α_2 denotes the ratio of unseen features by the total number of features present in the document d_p.

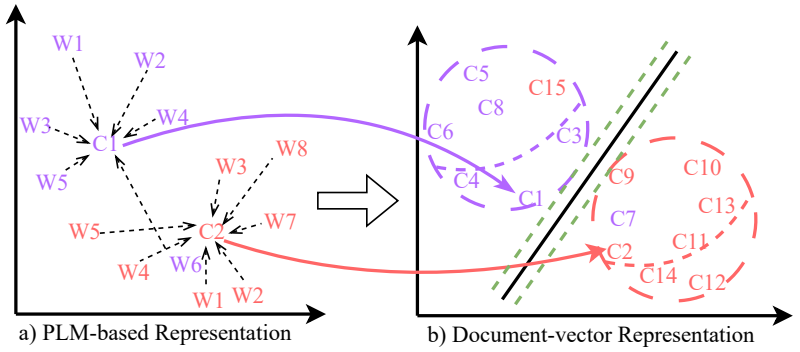

Figure 3. Conversion of textual features from word embedding to document-vector representation. W_i denotes words in a document. C_i denotes the center of the document i. The right-hand side diagram is the converted representation of documents based on each document's centers. All centers of each document are in $l-$dimensional space of word embedding, making the shape of a sphere surrounding associated data samples belong to each class.

The final computed document-vector representation C_p for the document d_p is assigned to the sign function to get a prediction, i.e., $y\prime_p = \textbf{sign}(\hat{w}^T C_p + \hat{b})$. It is denoted using equation (11).

$$y\prime_p = \textbf{sign}(\hat{w}^T C_p + \hat{b}) = \begin{cases} 1, & \hat{w}^T C_p + \hat{b} \geq 0 \\ -1, & \hat{w}^T C_p + \hat{b} < 0 \end{cases} \tag{11}$$

The complete workflow of the proposed approach during the deployment phase is presented in Figure 2.

3.5. Why Proposed Approach Works?

In section 3.3, each document is converted to its document-vector representation. Due to the nature of PLM, similar features are situated near compared to those that are different from each other. Because we use PLM, the newly seen feature in the pre-processing phase is assigned the associated word embedding. Because of the document-vector representation procedure, a document is represented by its center vector, which pulls the unseen features to the learned features that are similar in embedding space. In Figure 3, the textural features in the word embedding representation of a document are represented by the center vector C. A document belonging to a particular class will pull its constituted features to its center. The more a document pulls features to its center, the more critical it is to that particular class. By incorporating unseen features, the number of features increases, which can be influential during predictions. This makes the unseen features enhanced approach robust compared to those that discard or drop the unseen features.

4. Experimental Setup and Results

In this section, we discuss the dataset used for experiments. We also discuss evaluation metrics used to evaluate the effectiveness of the proposed approach compared to baselines and state-of-the-art approaches. Further, we report the performance evaluation results and finally discuss comparative analysis.

Table 1. Statistics of the dataset

	Domain	**# Total instances**
DS01	Book	975,194
DS02	Electronics	23,009
DS03	Video	36,180
DS04	DVD	124,438
DS05	Kitchen and housewares	19,856
DS06	Apparel	9,252
DS07	Beauty	2,884
DS08	Baby	4,256
DS09	Music	174,180
DS10	Grocery	2,632
DS11	Magazines	4,191
DS12	Gourmet food	1,575
DS13	Cell phones and service	1,023
DS14	Health and personal care	7,225
DS15	Jewelry and watches	1,981
DS16	Toys and games	13,147
DS17	Camera and photo	7,408
DS18	Automotive	736
DS19	Sports and outdoors	5,728
DS20	Outdoor living	1,599
DS21	Software	2,390
DS22	Computer and video games	2,771
DS23	Musical instruments	332
DS24	Office products	431
DS25	Tools and hardware	112

4.1. Datasets and Evaluation Metrics

In order to perform experiments and to effectively address the issue of covariate shift, we require to have a dataset that contains data from multiple domains or classes in which data from new domains that are not seen during the training phase can be introduced to show the

problem of covariate shift. For this purpose, we use the multi-domain sentiment dataset[2] which is collected by [2] from Amazon.com. This dataset consists of data from 25 different types of domains. Each domain has three types of reviews, i.e., positive, negative, and unlabeled. Because we are not performing sentiment classification, we are not concerned about each product review's label. We essentially perform a topic (product) classification task, i.e., given a product review, the machine learning model predicts the product type to which the review belongs.

From each domain, 1000 product reviews are selected through a random selection process to perform experiments. In some domains, there are fewer reviews. In those domains, we select 70% of the reviews for training the positive class, and the rest, 30%, are used as testing data. The complete statistic of the dataset is presented in Table 1. To perform experiments, each product serves as a positive class, resulting in 25 test experiments. In each experiment, apart from one positive class, data of 10 products are combined to make the negative train class. During the testing phase, the testing set contains data from one positive class domain and data from 10 domains comprising the negative class. Apart from the data of the above domains, data from 10 other domains that the model has not seen during training are assigned to the classifier as testing data during the deployment phase. This makes the covariate shift problem. During all experiments, the testing data will always have data from 21 different domains. We make it a total of 21 domains because domains such as DS18, DS23, DS24, and DS25 have fewer data instances.

Each review undergoes a preprocessing step during the training and deployment phase. We have used the `Natural Language Toolkit (NLTK)` library for preprocessing. In the preprocessing step, each review text is tokenized into words. Moreover, it also includes the removal of punctuation, stop words, non-alphabetic words, and one-lettered words. Further, each review is case-folded to normalize the text of the review. To prepare the document-vector representation of a review, each token present in the review is looked at in the vocabulary of word embedding. We have used the 100 dimensional `GloVe` word embedding. Those words that are not present in the vocabulary of word embedding are assigned a zero vector. Further, the average of all embedding vectors associated with tokens present in a review is computed to represent a review in its document-vector representation. The document-vector is used to train the *SVM* classifier during the training phase. We have used the `scikit-learn` implementation of *SVM* to train the classifier. In the deployment phase, the document-vector is assigned to the *SVM* classifier for prediction.

In order to evaluate the effectiveness of proposed and baseline approaches, we use standard evaluation metrics – *precision*, *recall*, *f-score*, and *accuracy*. The evaluation metrics are defined in equations 12, 13, 14, and 15, respectively. TP indicates the number of positive reviews correctly predicted as positive. FP indicates the number of negative reviews incorrectly predicted as positive. TN indicates the number of negative reviews correctly predicted as negative. FN indicates the number of positive reviews incorrectly predicted as negative.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{12}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{13}$$

[2]https://www.cs.jhu.edu/ mdredze/datasets/sentiment/

Table 2. Precision: Performance evaluation results of the baselines and proposed approach over multi-domain dataset

Domain	SVM	O-SVM	E-SVM	CBS-L	UFB-L
DS01	71.90	50.00	52.57	55.77	75.59
DS02	54.84	40.88	46.47	47.59	55.00
DS03	78.30	49.21	53.25	51.92	73.98
DS04	69.53	60.33	50.59	63.31	71.09
DS05	55.56	35.56	48.62	49.43	55.33
DS06	67.54	41.84	61.94	58.04	79.38
DS07	78.76	55.56	57.69	67.63	68.15
DS08	64.10	44.03	50.94	61.42	67.24
DS09	84.00	51.49	48.31	57.69	89.62
DS10	73.77	46.36	57.62	65.73	71.64
DS11	81.42	50.35	52.10	55.29	81.67
DS12	75.89	39.44	55.00	57.34	73.64
DS13	67.24	46.15	58.22	63.70	70.34
DS14	55.32	38.46	43.43	49.12	55.07
DS15	70.80	48.53	63.83	61.07	74.31
DS16	55.10	44.78	52.26	52.63	71.56
DS17	70.77	48.98	55.92	58.94	71.43
DS18	63.89	40.15	48.77	54.09	55.56
DS19	55.04	43.45	44.32	51.15	53.44
DS20	60.45	44.74	51.83	57.79	62.04
DS21	75.25	48.12	54.25	63.77	74.14
DS22	97.30	67.24	47.40	94.44	80.00
DS23	65.09	47.37	53.72	59.09	69.03
DS24	71.68	40.33	54.10	62.10	70.69
DS25	67.16	30.97	82.05	58.97	68.89

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{15}$$

4.2. Performance and Comparative Analysis

In order to evaluate the performance of the proposed unseen features-based (*UFB-L*) approach, we compare it with three baselines and one state-of-the-art approach, i.e., Support Vector Machine (*SVM*), One-Class SVM (*O-SVM*), Embedding-based SVM (*E-SVM*), and Fei & Liu [7]'s approach (*CBS-L*), respectively. The implementation details and a short description of each of the aforementioned approach is presented as follows:

- Support Vector Machine (*SVM*): In this baseline, we used the `LIBSVM` [3] implementation of *SVM*. We used linear kernel, as it performs best with the linear kernel as

Table 3. Recall: Performance evaluation results of the baselines and proposed approach over multi-domain dataset

Domain	SVM	O-SVM	E-SVM	CBS-L	UFB-L
DS01	87.00	69.00	92.00	87.00	96.00
DS02	68.00	56.00	79.00	79.00	77.00
DS03	83.00	62.00	90.00	81.00	91.00
DS04	89.00	73.00	86.00	88.00	91.00
DS05	75.00	48.00	88.00	86.00	83.00
DS06	77.00	59.00	83.00	83.00	77.00
DS07	89.00	75.00	90.00	94.00	92.00
DS08	75.00	59.00	81.00	78.00	78.00
DS09	84.00	69.00	86.00	90.00	95.00
DS10	90.00	70.00	87.00	94.00	96.00
DS11	92.00	72.00	87.00	94.00	98.00
DS12	85.00	56.00	77.00	82.00	95.00
DS13	78.00	60.00	85.00	93.00	83.00
DS14	78.00	60.00	86.00	84.00	76.00
DS15	80.00	66.00	90.00	91.00	81.00
DS16	81.00	60.00	81.00	80.00	78.00
DS17	92.00	72.00	85.00	89.00	95.00
DS18	92.00	55.00	79.00	86.00	80.00
DS19	71.00	63.00	82.00	89.00	70.00
DS20	81.00	68.00	85.00	89.00	85.00
DS21	76.00	64.00	83.00	88.00	86.00
DS22	64.29	78.00	82.00	60.71	85.71
DS23	69.00	72.00	65.00	65.00	78.00
DS24	81.00	73.00	66.00	77.00	82.00
DS25	90.00	62.50	57.14	92.00	93.00

reported by [10, 4]. The *TF-IDF* features representation with no feature selection is used to represent each review.

- One-Class SVM (*O-SVM*): In this baseline, we have used the `scikit-learn` implementation of *O-SVM* [17]. We have used the Gaussian kernel as it performs the best with *O-SVM*, as stated by [12]. We have used *TF-IDF* features representation with no feature selection scheme to represent each review.

- Embedding-based SVM (*E-SVM*): In this baseline, we used word embedding as a features representation scheme and then applied traditional *SVM* for prediction. We have used linear kernel. The difference between this baseline and the proposed approach lies in incorporating the unseen features that appear for the first time during the deployment phase.

- Center-Based Similarity Space Learning (*CBS-L*) [7]: We have performed experiments over multiple versions of the *CBS-L*; however, during our experiments, it performed the best when we used uni-gram features representation without features selection. In

Table 4. F-score: Performance evaluation results of the baselines and proposed approach over multi-domain dataset

Domain	SVM	O-SVM	E-SVM	CBS-L	UFB-L
DS01	78.73	57.98	66.91	67.97	84.58
DS02	60.71	47.26	58.52	59.40	64.17
DS03	80.58	54.87	66.91	63.28	81.61
DS04	78.07	66.06	63.70	73.64	79.82
DS05	63.83	40.85	62.63	62.77	66.40
DS06	71.96	48.96	70.94	68.31	78.17
DS07	83.57	63.83	70.31	78.66	78.30
DS08	69.12	50.43	62.55	68.72	72.22
DS09	84.00	58.97	61.87	70.31	92.23
DS10	81.08	55.78	69.32	77.37	82.05
DS11	86.38	59.26	65.17	69.63	89.09
DS12	80.19	46.28	64.17	67.49	82.97
DS13	72.22	52.17	69.11	75.61	76.15
DS14	64.73	46.88	57.72	61.99	63.87
DS15	75.12	55.93	74.69	73.09	77.51
DS16	65.59	51.28	63.53	63.49	74.64
DS17	80.00	58.30	67.46	70.92	81.55
DS18	75.41	46.41	60.31	66.41	65.57
DS19	62.01	51.43	57.54	64.96	60.61
DS20	69.23	53.97	64.39	70.08	71.73
DS21	75.62	54.94	65.61	73.95	79.63
DS22	76.92	72.22	60.07	71.88	82.76
DS23	76.06	57.14	58.82	68.75	73.24
DS24	66.99	51.96	59.46	61.90	75.93
DS25	77.42	41.42	67.37	73.91	79.15

order to transform uni-gram feature space to centrality-based space, we used uni-gram features with no feature selection. In this approach, we have used the Gaussian kernel as proposed by the authors.

In Tables 2 to 5, we report the *precision*, *recall*, *f-score*, and *accuracy*, respectively, of the baseline approaches and proposed approach. In Figures 4 to 7, we display the visualization of the results reported using various evolution metrics such as *precision*, *recall*, *f-score*, and *accuracy*. From Tables 2 to 5 and Figures 4 to 7, it can be observed that the proposed approach have outperformed the baseline approaches and the-state-of-the-art approach in almost all evaluation metrics.

In Figure 8 and Figure 9, we show the *accuracy* and *f-score* of the DS01 domain over the inclusion of the different number of domains in the negative class during the training phase. It can be seen that as we introduce more domains during training of the negative class, apart from *E-SVM*, the performance of all methods improves. The reason is that the classifier have seen these domains during the training phase. As the number of domains increases in training negative class, the problem of covariate shift alleviates, and the setting of the problem

Table 5. Accuracy: Performance evaluation results of the baselines and proposed approach over multi-domain dataset

Domain	SVM	O-SVM	E-SVM	CBS-L	UFB-L
DS01	84.84	67.74	70.65	73.55	88.71
DS02	71.61	59.68	63.87	65.16	72.26
DS03	87.10	67.10	71.29	69.68	86.77
DS04	78.07	75.81	68.39	79.68	85.16
DS05	72.58	55.16	66.13	67.10	72.90
DS06	80.65	60.32	78.06	75.16	86.13
DS07	88.71	72.58	75.48	83.55	83.55
DS08	78.39	62.58	68.71	77.10	80.65
DS09	89.68	69.03	65.81	75.48	94.84
DS10	86.45	64.19	75.16	82.26	86.45
DS11	90.65	68.06	70.00	73.55	92.26
DS12	86.45	58.06	72.26	74.52	87.42
DS13	80.65	64.52	75.48	80.65	83.23
DS14	72.58	56.13	59.35	66.77	72.26
DS15	82.90	66.45	80.32	78.39	84.84
DS16	72.58	63.23	70.00	70.32	82.90
DS17	85.16	66.77	73.55	76.45	86.13
DS18	80.65	59.03	66.45	71.94	72.90
DS19	71.94	61.61	60.97	69.03	70.65
DS20	76.77	62.58	69.68	75.48	78.39
DS21	84.19	66.13	71.94	80.00	85.81
DS22	82.58	80.65	64.84	76.77	92.48
DS23	83.55	65.16	70.65	77.42	81.61
DS24	78.06	56.45	70.97	74.19	83.23
DS25	92.11	62.78	88.35	90.98	84.19

evolves to the classical supervised classification. Due to the space limit, we have only shown the above results for the DS01 domain.

4.3. Deciding the Values of α_1 and α_2 for Document-Vector Representation

In order to decide the optimum value for α_1 and α_2 for the final document-vector representation C_p, we performed extensive experiments on different combinations of the values for α_1 and α_2. In equation (10), α_1 denotes the hyper-parameter associated with learned features $\mathcal{F}_l$, and α_2 denotes the hyper-parameter associated with unseen features $\mathcal{F}_u$. We have considered four different settings as follows:

- Setting-1 (*ST-1*): In this setting, we have assigned $\alpha_1 = 0.25$ and $\alpha_2 = 0.75$. As we have assigned the α_2 more weight than the α_1, this setting will focus more on the unseen features.

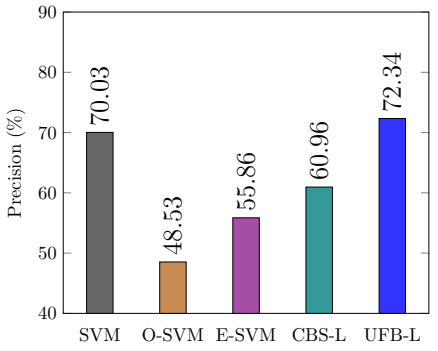

Figure 4. Average *precision* of proposed approach and all baselines over all domains of multi-domain sentiment dataset

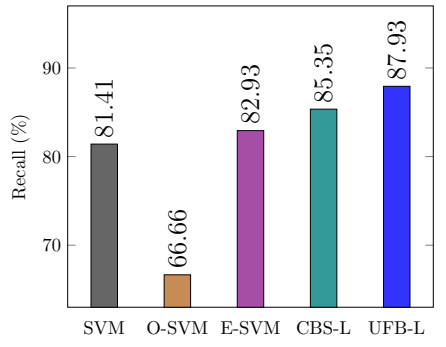

Figure 5. Average *recall* of proposed approach and all baselines over all domains of multi-domain sentiment dataset

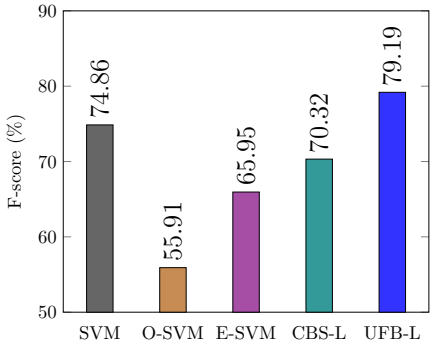

Figure 6. Average *f-score* of proposed approach and all baselines over all domains of multi-domain sentiment dataset

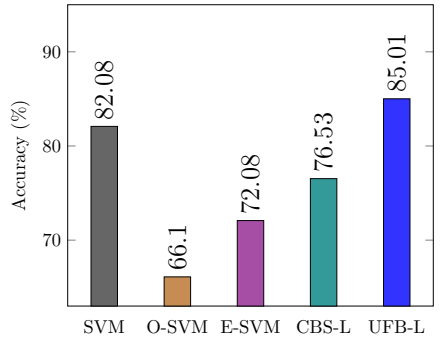

Figure 7. Average *accuracy* of proposed approach and all baselines over all domains of multi-domain sentiment dataset

- Setting-2 (*ST-2*): In this setting, we have assigned $\alpha_1 = 0.5$ and $\alpha_2 = 0.5$. We assigned equal weight to α_1 and α_2. In the final document-vector representation C_p, the contribution of both learned features $\mathcal{F}_l$ and unseen features $\mathcal{F}_u$ are equal in the making of the final document-vector representation C_p.

- Setting-3 (*ST-3*): In this setting, we have assigned $\alpha_1 = 0.75$ and $\alpha_2 = 0.25$. We have assigned more weight to the α_1 compared to the α_2; the focus in this setting will be more on those features that the machine learning model has already learned during the training phase compared to the unseen features $\mathcal{F}_u$.

- Setting-4 (*ST-4*): In this setting, we have assigned $\alpha_1 = \frac{|\mathcal{F}_l|}{|d_p|}$ and $\alpha_2 = \frac{|\mathcal{F}_u|}{|d_p|}$. In this setting, α_1 is the ratio of learned features by the total number of features that constitute document d_p, and α_2 is the ratio of unseen features by the total number of features that are present in document d_p.

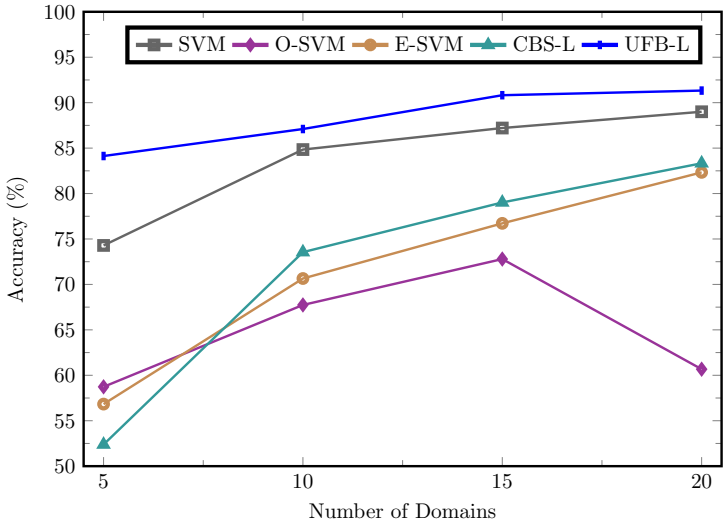

Figure 8. Accuracy scores of the proposed approach with baselines and state-of-the-art approach using different numbers of in-train domains as the negative class for the DS01 domain

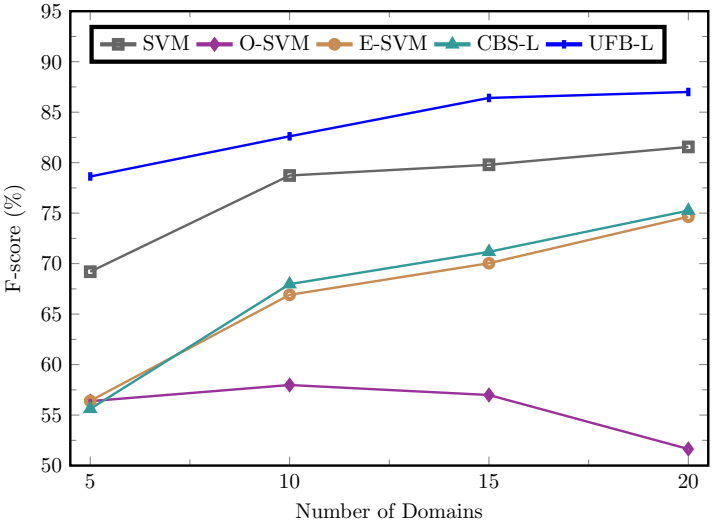

Figure 9. F-score of the proposed approach with baselines and state-of-the-art approach using different numbers of in-train domains as the negative class for the DS01 domain

In Figures 10 to 13, we show the performance of the proposed approach in different settings to decide the values of α_1 and α_2. It can be noted that, in *ST-1*, *ST-2*, and *ST-3*, as we have assigned more weight to the learned features, the performance of the proposed approach has been improved in different performance evolution metrics. It is intuitive, as the machine

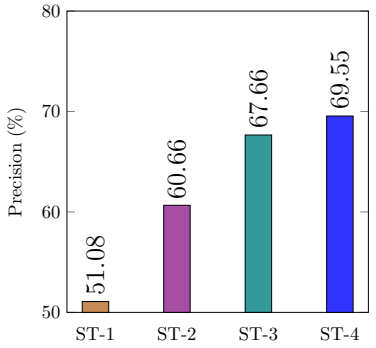

Figure 10. Average *precision* of the proposed approach for different values of α_1 and α_2 over all domains of multi-domain sentiment dataset

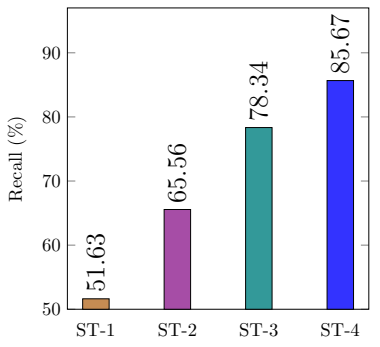

Figure 11. Average *recall* of the proposed approach for different values of α_1 and α_2 over all domains of multi-domain sentiment dataset

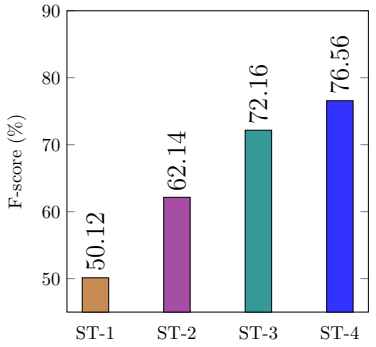

Figure 12. Average *f-score* of the proposed approach for different values of α_1 and α_2 over all domains of multi-domain sentiment dataset

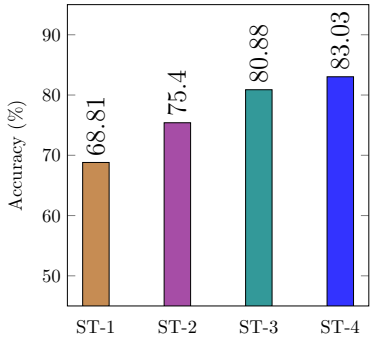

Figure 13. Average *accuracy* of the proposed approach for different values of α_1 and α_2 over all domains of multi-domain sentiment dataset

learning model learned those features during training. The proposed approach performs best when assigned values to the α_1 and α_2 are based on *ST-4*. In *ST-4*, the values of α_1 and α_2 are dynamically allotted based on the number of learned and unseen features present in document d_p. Therefore, we have selected *ST-4* to assign values of α_1 and α_2 in the proposed approach.

4.4. Limitations

The issue of covariate shift is widespread in social media analytics because of its dynamic nature. As the data changes over time, it may include new features that the machine learning model has not encountered during its training phase; a more dynamic environment results

in a greater shift in covariates. In order to address this problem of covariate shift, we employ the notion of unseen features by utilizing a pre-trained language model (PLM) at the deployment phase. But PLM-based methods have drawbacks. One such drawback is the out-of-vocabulary word problem, which in our case is handled by assigning a zero-value vector. However, we believe that it can be handled more efficiently. Another issue with the PLM-based method is that it needs to be updated periodically to account for the way social media is changing and the terms that are becoming more popular.

5. Conclusion

In this paper, we have presented a text classification approach based on pre-trained language models that effectively addresses the negative covariate shift problem by utilizing unseen features. Covariate shift occurs when there is a discrepancy between the distributions of the training and testing data. A distribution shift refers to the presence of new data that was not previously encountered by the model during the training phase. The newly encountered data generally includes new features for which the classifier may not have a parametric weight. As a result, these features are eliminated in literature. Such features are referred to as unseen features, and we used them in the proposed approach. We have employed a pre-trained language model to handle unseen features, which converts each document into its document-vector form. The proposed method is compared to one state-of-the-art approach and three baseline approaches. The empirical results show that the proposed method is effective in efficiently managing the negative covariate shift, outperforming both the state-of-the-art and baseline approaches. The research of negative covariate shift in deep learning-based systems is a worthwhile undertaking as an immediate future work, since there is limited literature available on the subject of negative covariate shift. It will be interesting to see whether it considerably improves real-world application performance.

References

[1] Bickel S., Brückner M., and Scheffer T. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(9), 2009.

[2] Blitzer J., Dredze M., and Pereira F. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 440–447. ACL, 2007.

[3] Chang C.-C. and Lin C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] Colas F. and Brazdil P. Comparison of svm and some older classification algorithms in text classification tasks. In *Artificial Intelligence in Theory and Practice*, pages 169–178. Springer US, 2006.

[5] Devlin J., Chang M., Lee K., and Toutanova K. 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, NAACL-HLT, Minneapolis, MN, USA, June 2-7*, pages 4171–4186. ACL, 2019.

[6] Fang T., Lu N., Niu G., and Sugiyama M. Rethinking importance weighting for deep learning under distribution shift. In *In Proceedings of the Advances in Neural Information Processing Systems*, volume 33, pages 11996–12007. Curran Associates, Inc., 2020.

[7] Fei G. and Liu B. Social media text classification under negative covariate shift. In *Proceedings of the 20th Conference on Empirical Methods in Natural Language Processing*, pages 2347–2356, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics.

[8] Hammoudeh Z. and Lowd D. Learning from positive and unlabeled data with arbitrary positive shift. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 13088–13099. Curran Associates Inc., 2020.

[9] Heckman J. J. Sample selection bias as a specification error. *Econometrica*, 47(1):153–161, 1979.

[10] Joachims T. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, page 137–142. Springer-Verlag, 1998.

[11] Joulin A., Grave E., Bojanowski P., Douze M., Jégou H., and Mikolov T. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651, 2016.

[12] Khan S. S. and Madden M. G. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

[13] Liu B., Dai Y., Li X., Lee W., and Yu P. Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 179–186, 2003.

[14] Mikolov T., Chen K., Corrado G., and Dean J. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations, ICLR, Scottsdale, Arizona, USA, May 2-4, Workshop Track Proceedings*, 2013.

[15] Minter T. Single-class classification. In *In Symposium on Machine Processing of Remotely Sensed Data*, page 54, 1975.

[16] Nguyen T., Lyu B., Ishwar P., Scheutz M., and Aeron S. Joint covariate-alignment and concept-alignment: A framework for domain generalization. In *In Proceedings of the 32nd IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2022.

[17] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., V.Dubourg, Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., and Duchesnay E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] Pennington J., Socher R., and Manning C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, October 25-29, 2014, Doha, Qatar*, pages 1532–1543. ACL, 2014.

[19] Peters M. E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., and Zettlemoyer L. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[20] Radford A., Narasimhan K., Salimans T., and Sutskever I. Improving language understanding by generative pre-training. *Open AI*, 2018.

[21] Sakai T. and Shimizu N. Covariate shift adaptation on learning from positive and unlabeled data. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2019.

[22] Shimodaira H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[23] Sugiyama M. and Müller K.-R. Input-dependent estimation of generalization error under covariate shift. *Statistics & Risk Modeling with Applications in Finance and Insurance*, 23(4):249–279, 2005.

[24] Sugiyama M., Nakajima S., Kashima H., Buenau P., and Kawanabe M. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of the 21st Annual Conference on Advances in Neural Information Processing Systems*, volume 20, pages 1433–1440. Curran Associates, Inc., 2007.

[25] Tax D. M. and Duin R. P. Support vector domain description. *Pattern Recognition Letters*, 20(11):1191–1199, 1999.

[26] Tian J., Hsu Y.-C., Shen Y., Jin H., and Kira Z. Exploring covariate and concept shift for out-of-distribution detection. In *In Proceedings of the NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[27] Wasi N. A. and Abulaish M. An unseen features enhanced text classification approach. In *In the Proceddings of the International Joint Conference on Neural Networks (IJCNN)*, page 8, Queensland, Australia, June 2023.

[28] Wasi N. A. and Abulaish M. An unseen features-enriched lifelong machine learning framework. In *In the Proceddings of the International Conference on Computational Science and Its Applications – ICCSA*, pages 471–481, Athen, Greece, 2023. Springer Nature Switzerland.

[29] Yang Z., Dai Z., Yang Y., Carbonell J., Salakhutdinov R. R., and Le Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the 32th Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[30] Yu H., Han J., and Chang K. C.-C. Pebl: Positive example based learning for web page classification using svm. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 239–248, New York, NY, USA, 2002. Association for Computing Machinery.

[31] Zhou A. and Levine S. Bayesian adaptation for covariate shift. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems 2021, NeurIPS*, pages 914–927, 2021.

Received 17.09.2023, Accepted 17.06.2024