Catalyst: Combining Co-training and Active Learning for Lifelong Classification

Maxim A. Ryndin
Ivannikov Institute for System Programming of the RAS
Moscow, Russia
ORCID: 0000-0002-7504-3975

Denis Y. Turdakov
Ivannikov Institute for System Programming of the RAS
Lomonosov Moscow State University
Moscow, Russia
ORCID: 0000-0001-8745-0984

Sergey D. Kuznetsov
Ivannikov Institute for System Programming of the RAS
Lomonosov Moscow State University
Moscow Institute of Physics and Technology (State University)
National Research University Higher School of Economics
Plekhanov Russian University of Economics
Moscow, Russia
ORCID: 0000-0002-8257-028X

***Abstract*—Modern supervised algorithms assume that the dataset used for training has the same distributions as the data to be processed. However, the real data is permanently changing. This leads to the gradual degradation of supervised machine learning algorithms in production systems and increases the cost of the maintaining.**

To solve this problem, we are focusing on domain adaptation of machine learning algorithms in lifelong manner. We assume that real unlabelled data come in continuously. For this setting we propose a method for detecting changes in data distributions, as well as updating supervised algorithms.

The idea behind the method is to process a portion of the data and create a new labelled dataset for training a supervised model. The trained model becomes a part of the ensemble used for selecting a strategy to deal with new examples: assign the label automatically using co-training or manually with the aid of active learning. This method is independent of the specific architecture of the model and could be used with any modern supervised algorithms, including artificial neural networks.

Our research also confirms two findings. First, adding small portion of data with reliable labels to a self-labelled dataset improves model's performance, even if this amount is small to build a model from scratch. It is also shown that accumulating domain knowledge by continuously adding new trained models to ensemble used for labelling, reduces the amount of labelled data required while maintaining the high performance of the adapted model.

***Index Terms*—co-training, active learning, lifelong learning, feature drift, concept drift**

I. Introduction

The current dominant machine learning paradigm assumes that training and testing are performed on data obtained from the same distribution. However, in the real world, distributions of features and labels can change over time: slowly due to natural evolution of a source (for example, new slang in social networks) or drastically upon the occurrence of big events. These processes are known as *feature and concept drifts* [1]. If developers do not take this factor into account in industrial applications, it will lead to a gradual degradation of the entire system.

For example, such model degradation was observed during SentiRuEval2016 [2] competition devoted to the assessment of sentiment of Russian-language tweets about banks. Public and private parts of the dataset were collected with time gap of several months during Russian financial crisis. Events at that time led to language changes in target domain and feature drift happened. As a result, all participants obtained high gap in performance on training and test parts of the dataset.

Such situation is often found in production systems that use supervised techniques for natural language processing. The main reason is that language is constantly evolving and drift processes are natural for new texts. Usually the problem is solved by retraining the model on new data. But first someone has to detect the problem and then obtain labelled data for training. It is worth mentioning that data labelling is the most expensive and time consuming stage.

The main idea for solution of the discussed problem is to use unlabeled data for continuous correction of the model throughout its operation. In this work we study *lifelong machine learning setup* [3], which means that unlabeled data is received by portions one after the other, for example, every day. However, it is very expensive or even not possible to assign labels manually to unlabeled examples every time they come. Therefore, we propose to combine several techniques that allow to significantly reduce a cost of the supervised model support in production environment.

We use *domain adaptation* [4] techniques that automatically improve a model using unlabeled data. Intuition dictates that if the distribution of data changes slowly, we can gradually adapt our model to these changes without human intervention. In particular, *co-training* technique may be used. It applies two or more independent models to the same data and automatically assigns labels to the examples if classifiers are confident of it. However, domain adaptation algorithms can not cope with concept drift and fast feature changes. Thus, in any case, we have to manually label the data for complex cases. To reduce the number of examples for manual markup, we use *active learning* technique [5]. Active learning tries to find examples that can improve the models in the best way.

Main contribution of this paper is a method for identifying good examples for automatic and manual labelling in order to maintain model performance while reducing the cost of manual labelling. In contrast to existing works, this method could also, in a low cost, generate large training datasets that could be used to train neural networks. Neural networks are currently the most popular supervised models because of their high accuracy and generalisation performance. However, a lot of data is usually required for such models to start perform well. The evaluation of the proposed method shows that it helps to solve this problem as well. We show that the performance of recurrent neural network trained on data obtained using our method reaches nearly in-domain level whereas the number of manually labelled examples turns out to be small.

II. Related Work

Domain adaptation problem is often formulated as follows: given labelled data from source domain (X_s, y_s) and unlabelled data from target domain X_t, to build a model predicting y_t better than a model using only X_s and y_s. For the problem to be non-trivial X_s and X_t have different distributions, when y_s and y_t are supposed to have the same. Several techniques for neural networks show promising results such as domain adversarial training [6], self-training [7] and co-training [8]. Main idea of self-training and co-training is expanding training dataset with self-labelled examples. Only examples in which the model is confident enough are added to training dataset. If X_t and X_s have "close" distributions, a lot of extra training examples will be labelled and classification surface will automatically move to real, that can be learnt from (X_t, y_t). There are several limits of this approach – it can not fight concept drift and fast feature drift.

The main idea of active learning algorithms [9] is to let the trained model find itself best examples for training. Model retrains several times and on each iteration a little number of training examples for next iteration is chosen. It's hard to apply this technique to neural networks training because

- neural networks often need huge training dataset to rich peak performance,
- retraining neural network several dozens times can be very time-consuming,
- there are not that many theoretical understanding of best sampling strategies for non-linear models.

One of the most frequently used strategies to sample examples for labelling is uncertainty sampling both in co-training and active learning. Key difference is that co-training algorithms search for examples that model is most confident about [10] when active learning algorithms on the contrary prefer the least confident [11].

Machine learning paradigm that is focused on accumulating knowledge over time to keep models performing well is known as lifelong [3] learning. It is worth mentioning that lifelong learning often uses techniques similar to multitask [12] and transfer [13] learning paradigms, but it pursues other goals. Works focused on lifelong classification problems often uses "classical" machine learning algorithms. In particular, Naïve Bayesian Classification [14] is widely studied and used in lifelong literature. It is proved that the performance of such models is lower than the performance of nonlinear models, and from a point of view of practice, the issue of developing lifelong methods for nonlinear models is more important, which, however, is little studied.

We propose an algorithm that combines ideas of self-labelling "easy" examples and finding and human labelling most useful "hard" examples. We suggest to use this algorithm in lifelong manner, so let us first describe the model of the environment.

III. Environment Model

We assume that we get new data at discrete time points t_i. At the start point t_0, we have training data (X_0, y_0). In next time points $t_i, i > 0$ we get only data X_i without labels and distribution of X_i can differ from distribution of X_{i-1}. Time points are distributed in a way that $|X_i| \approx |X_0|,\ \forall i > 0$, that is, the data come in batches of the approximately same size.

On each step, we can build a new model M_i^A using algorithm A and all previously available data:

$$M_i^A = A(\{X_\tau\}_{\tau=0}^{i}, y_0). \qquad (1)$$

The "ideal" model for i^{th} time stamp is $M_i' = A(X_i, y_i)$ – one that can be built if we know all the labels. Out goal is to build such algorithm A that

$$perf(M_0) < pref(M_i^A) \leq perf(M_i'), \qquad (2)$$

where $perf$ is one of quality metrics.

IV. Proposed Method

A. Algorithm

On the step t_0, we get labelled data (X_0, y_0) and build M_0 using selected machine learning algorithm. The next timestamp we already have a model M_0, but its performance is not ideal due to domain shift. However, we can use this model to generate training data for another model, that fits new data distribution better. For this purpose we combine co-training and active learning approaches.

On the each step i the method

- selects examples $\hat{X}_i \subset X_i$ for labelling using previous models,
- chooses between automatic and manual labelling.

The method uses combination of confidence and agreement of previous models for decision making. This allows to process all data in one pass through X_i and train M_i only once, which is more time efficient than active learning iterative training.

After labelling $\hat{X}_i$ we get a training set $(\hat{X}_i, \hat{y}_i)$ that is used to build a new model M_i. This model has new information of

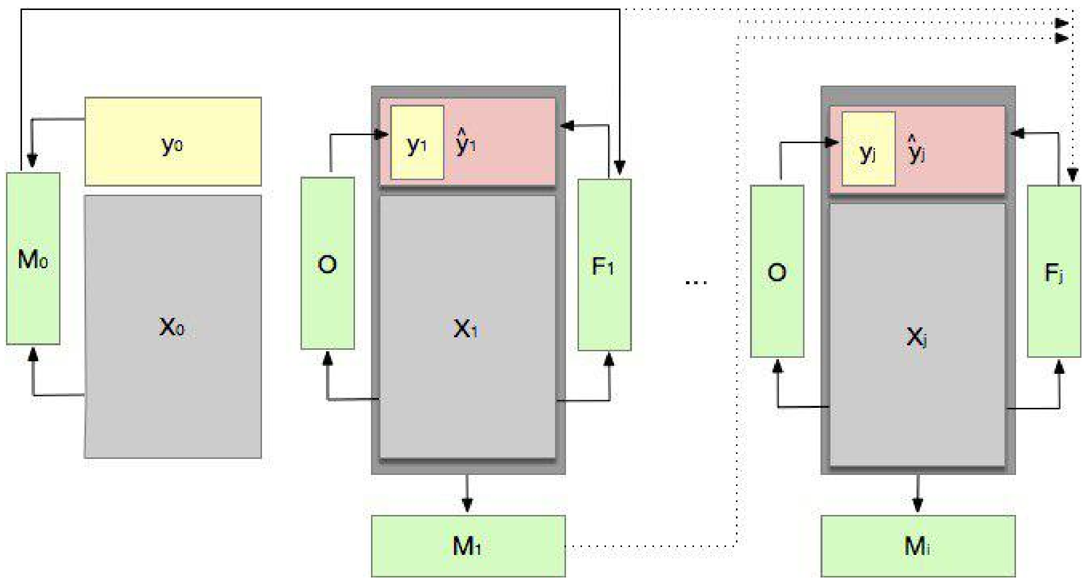

Fig. 1. Scheme of the method

current data distribution (X_i, y_i) and is useful to self-label examples on next step $i+1$. We suggest to add models built on each step to group of models that perform co-training labelling.

We divide all examples from X_i into three groups:

1) *High confident group* – we are likely to trust self-labelling for examples in this group, but these examples seem to be not specific for current data distribution.
2) *Low confident group* – our model does not seem to see such examples before, they are more likely to need to be labelled manually.
3) *Medium confident group* – examples in this group seems to be specific for current data distribution and we should decide if we can trust self-labelling.

It's natural to assume that in the most cases domain distributions are changing slowly. Therefore we will usually have much more examples in high confident group than in low confident one. But in the case of fast domain shift we will see the opposite situation. To balance the number of "hard" and "easy" examples to be labelled, each example is added to training set with probability opposite to confidence. Probabilistic approach of forming training set can be treated as forgetting mechanism that allows us to build models that are more specific to current distribution of data.

B. *Formal Definitions and Sampling Strategy*

Let's make some definitions and describe algorithm formally. We assume labelling expert to have fixed labelling price and make no mistakes. We will call him O.

We define set of fallible oracles F. On each i^{th} step, this set contains all previously built models: $F_i = \{M_\tau\}_{\tau=0}^{i-1}$. This oracles are free, but they can make mistakes.

We assume that output of our classifiers is probabilistic and we can think the probability of a mistake to be proportional to confidence. For simplicity, we will focus on binary classification problem, further reasoning can be naturally generalised for multiclass classification.

$p^j = \{p_0^j \dots p_{i-1}^j\} = \{M_0(x^j) \dots M_{i-1}(x^j)\}$ is a probability of example $x^j \in X_i$ belongs to the class "1". We define mean confidence as a distance from class decision boundary (0.5 in binary case):

$$C^j = \mathbb{E}(|p_k^j - 0.5|). \tag{3}$$

Vector of classes predicted by each fallible oracle is $\tilde{y}^j = \{\tilde{y}_0^j \dots \tilde{y}_{i-1}^j\} = \{\mathbb{1}(p_k^j > 0.5)\}_{0 \le k \le i-1}$

We suggest using not only confidence but also agreement as additional metric to make decision how to label given example. We define agreement as

$$A^j = \frac{A_0^j - A_1^j}{A_0^j + A_1^j}, \tag{4}$$

where A_0^j, A_1^j are number of fallible oracles voted for class "0" or "1" respectively: $A_l^j = \sum_{\tilde{y}_k^j \in \tilde{y}^j} \mathbb{1}(\tilde{y}_k^j = l),\ l \in \{1, 0\}$. Note that sign of A^j is used to distinguish the main class, therefore we don't use the modulo.

We will also need mean confidence for each class

$$C_l^j = \mathbb{E}_{k|\tilde{y}_k^j = l}(|p_k^j - 0.5|),\ l \in \{1, 0\} \tag{5}$$

and dominating class mean confidence

$$C_{dom}^j = C_0^j, \text{ if } A^j > 0,\ C_1^j \text{ otherwise}. \tag{6}$$

$\theta_1, \theta_2, \theta_3$ are algorithms hyperparameters, defining agreement and confidence thresholds which are used in Algorithms 1, 2, 3 correspondingly.

On i^{th} step, we form training set $(\hat{X}_i, \hat{y}_i)$, $\hat{X} \subset X_i$ using following "Sample and Label" Algorithm 1.

Algorithm 1: "Sample and Label"

```
1:  $\hat{X}$=[], $\hat{y}$=[]
2:  for $x^j \leftarrow X_i$ do
3:      calculate $p^j, \tilde{y}^j$
4:      calculate $C^j, C_0^j, C_1^j, C_{dom}^j, A^j$
5:      if $Bernoulli(C^j) == 0$ then
6:          $\hat{X} = \hat{X} \bigcup x^j$
7:          if $|A^j| \leq \theta_1$ then
8:              $\hat{y} = \hat{y} \bigcup L_{low}(x^j, O, A^j, C_0^j, C_1^j)$
9:          else
10:             $\hat{y} = \hat{y} \bigcup L_{high}(x^j, O, A^j, C_{dom}^j)$
11:         end if
12:     end if
13: end for
14: return $\hat{X}, \hat{y}$
```

Sampling by confidence is performed in line 5. We prefer specific examples for labelling. The lower the confidence, the higher the chances to get into the training set.

Then algorithm selects labelling strategy using agreement between previous models in lines 7-11. We propose two labelling strategies for low and high agreements.

After N iterations of the Algorithm 1 we will have N models. Each of them potentially has information of different data distribution. By accumulating models we accumulate more and more knowledge about possible changes in data. Therefore, chances to label more examples automatically increase.

If the ensemble of models has low agreement for a given example, then this example is more likely to be specific for current time point and models guess randomly. In this situation our rules to decide which way to label example should be more strict. These rules are presented in the Algorithm 2.

Algorithm 2: "L_{low}: Low Agreement Labelling"

```
1: if $A^j \geq 0$ and $C_0^j \geq \theta_2$ and $C_1^j < \theta_2$ then
2:     return 0
3: else if $A^j \leq 0$ and $C_0^j < \theta_2$ and $C_1^j \geq \theta_2$ then
4:     return 1
5: else
6:     return $O(x^j)$
7: end if
```

Algorithm 2 labels an example automatically if it was specific for part of time points and non-specific for other (lines 1-4). In this case, the mean confidence will be high in one group of the models and low in the other. Otherwise we label example manually (line 6).

If agreement is high then example seems to be less domain-specific and we can make rules less strict. This rules are represented in the Algorithm 3.

The algorithm 3 does not require dominant and non-dominant classes to have different level of confidence.

Also we use different confidence thresholds to have more control and intuitively $\theta_2 > \theta_3$.

Algorithm 3: "L_{high}: High Agreement Labelling"

```
1: if $C_{dom}^j \geq \theta_3$ then
2:     if $A^j > 0$ then
3:         return 0
4:     else
5:         return 1
6:     end if
7: else
8:     return $O(x^j)$
9: end if
```

As mentioned earlier this algorithm can be generalised for multiclass classification problem. Without deep details one need make following changes to algorithm:

- Single agreement value should be replaced with set of one-vs-all agreements value for each class, except one.
- Agreement modulo in algorithm 1 should be replaced with agreement's vector norm.
- Algorithms 2 and 3 receive additional branches for each class.

V. Evaluation

A. Dataset

Amazon review data[1] is a classical textual dataset used for evaluation of domain adaptation algorithms. Researches commonly use small preprocessed subset but it has several disadvantages. It is not publicly available on official webcite, it is very small for modern deep models and it has pre-extracted features, which makes unable to use modern language model features. That's why we decided to use the whole dataset for our experiments following common patterns in preprocessing.

We choose popular domains – Books (B, 8898041 reviews), Electronics (E, 1689188 reviews), Kitchen (K, 551682 reviews).

Task was transformed to binary classification task by following rule: if the label is greater than 3, example gains label "1", it gains label "0" otherwise (following the scheme from domain adaptation literature [15]).

Mean sentence length was calculated and it turned to be equal about 100 words long. All sentences were cut to 100 words. After vectorization (described in next subsection) shorter sequences were padded with zero vectors to have 100 elements.

To emulate our environment model we "put" one of domain in time point. We use data from one of domains to train M_0 and two other to test our algorithm in a sequence.

B. Model

As mentioned we developed the method aiming to use it with non-linear models. Our method only requires output of classifier to have probabilistic output.

[1]https://jmcauley.ucsd.edu/data/amazon/

TABLE I
AMAZON REVIEW, SINGLE FALLIBLE ORACLE

Source	Target	Accuracy, proposed method	Accuracy, self-training	Accuracy, active learning	Budget	Accuracy, no adaptation (lower bound)	Accuracy, indomain (upper bound)
B	E	**90.7** ± 0.2	88.4 ± 0.3	63.2 ± 0.4	0.064 ± 0.002	88.6 ± 0.1	91.2 ± 0.2
	K	**89.9** ± 0.4	86.6 ± 0.2	78.3 ± 0.2	0.056 ± 0.001	86.5 ± 0.2	91.3 ± 0.2
E	B	**90.1** ± 0.2	86.1 ± 0.2	84.5 ± 0.1	0.048 ± 0.001	87.0 ± 0.1	90.9 ± 0.1
	K	**90.2** ± 0.3	88.9 ± 0.3	78.1 ± 0.2	0.055 ± 0.002	89.1 ± 0.1	91.3 ± 0.2
K	E	**91.1** ± 0.4	88.5 ± 0.3	63.4 ± 0.3	0.066 ± 0.003	88.9 ± 0.3	91.2 ± 0.2
	B	**89.3** ± 0.2	86.3 ± 0.3	84.7 ± 0.1	0.049 ± 0.001	84.8 ± 0.1	90.9 ± 0.1

TABLE II
AMAZON REVIEW, SEVERAL FALLIBLE ORACLE

Sources	Target	Accuracy, proposed method	Accuracy, co-training	Accuracy, active learning	Budget
E, B	K	**89.6** ± 0.1	88.9 ± 0.2	72.7 ± 0.3	$0.023 \pm\ 0.001$
K, B	E	**90.1** ± 0.2	88.8 ± 0.3	59.8 ± 0.3	$0.034 \pm\ 0.002$
E, K	B	**88.7** ± 0.2	88.1 ± 0.4	76.5 ± 0.2	$0.025 \pm\ 0.002$

To keep balance between model power and training speed, we decided to use LSTM [16] neural network over language model vectors. We use triangular learning rate schedule described in [17] for training.

We use fasttext [18], [19] language model because text in chosen dataset is not formal and brief analysis showed presence of spelling errors . It has available pretrained[2] models and does not suffer from out-of-vocabulary problem. Fasttext feature vector is 300-dimensional.

As a classifier over fasstext features, we use a network with two hidden LSTM layers with ReLU activations and one output neuron with sigmoid activation for classification.

C. Experiments

We compare the proposed algorithm with pure co-training and active learning, which form the basis of our approach. Accuracy is used as a quality metrics.

$$Accuracy = \frac{\sum True\ positive + \sum True\ negative}{\sum Total\ population}$$

Due to the probabilistic nature of the proposed algorithm, it can give slightly different results on each run. Namely, it has following sources of randomness:

- neural network weight initialisation seed;
- seed for Bernoulli random variable;
- train and validation split.

We round performance results over 15 iterations to neutralise random effects and report mean and variance.

First we evaluate developed algorithm in domain adaptation manner: we use only model trained on source domain as an oracle for self-labelling target domain. Results are presented in Table 1. Accuracy of the proposed method exceeds pure co-training and active learning approaches. Note that co-training algorithm simplifies to self-training in such setup (case of one oracle).

[2]https://fasttext.cc/docs/en/english-vectors.html

To compare with active learning, we first find budget that our algorithm needs to achieve peak performance and then evaluate active learning algorithm with given budget. We use least confidence sampling strategy: sequentially select and assign a label to examples, for which the model has the lowest confidence. The proposed algorithm selects slightly different number of examples to be labelled manually because of randomness. We freeze this number on each iteration and report average active learning performance over set of this budgets.

We also show accuracy of the model applied to target domain without adaptation and trained on in-domain dataset as lower and upper bounds of expected performance.

The percentage of examples that remained for markup (either manually or automatically) after sampling was between 35% and 40% for all experiments. Rate of wrong labelled examples was $4\% \pm 1\%$ among all examples selected to be labelled by a model.

Secondly we evaluate proposed algorithm in lifelong manner. Results are presented in Table 2. We train the model on the labelled data from the first source domain. Then we build training set for the second source domain using first model as fallible oracle and train second model on this new dataset. On the third step we assign labels for all data in target domain using both models combined with our method and report the results. We show only one of two possible sequences of source domains because reordering produces similar results up to variance.

In this set of experiments we compare proposed algorithm with co-training and active sampling with the same budget. For co-training we use models trained on each domain independently and having in domain performance. It shows rather good results, slightly lower than our method.

For active learning we use the same scheme as in the first set of experiments. But it shows the worse result due to lower budget required to achieve peak performance for our algorithm. This budget has decreased two times on average while sampling and wrong labelling ration remains approximately the same.

VI. Conclusion and Discussion

We proposed the method, which

- combines ideas of co-training and active learning to efficiently adapt to environment changes;
- accumulates knowledge of previous time points to decrease labelling cost;
- can be used for lifelong classification.

Experiments show that idea of adding small number of validly labelled examples to self-labelled data improves performance of a model even if the amount of such examples is not enough to build a model from scratch. The second series of experiments confirms the hypothesis that the accumulating of domain knowledge using previously built models reduces the cost for training of the following models without significant drop in the performance.

The proposed method provides the basis for building an automated classification system. To improve the method, several important problems need to be solved.

First, current method stores every new model and after each iteration our algorithm gets slower and more resource-consuming. For real lifelong usage we need to develop a mechanisms to clean fallible oracle set or algorithm that will decide if it is worth adding given model to this set.

Second, the proposed algorithm uses several heuristics for sampling. These heuristics can be improved by utilising data-driven approach, for instance, using ensemble learning techniques.

VII. Funding

This work was supported by RFBR Grant 20-07-00561 A.

References

[1] A. Tsymbal, “The problem of concept drift: definitions and related work,” *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[2] N. Loukachevitch and Y. V. Rubtsova, “Sentirueval-2016: overcoming time gap and data sparsity in tweet sentiment analysis,” in *Computational Linguistics and Intellectual Technologies*, pp. 416–426, 2016.

[3] Z. Chen and B. Liu, “Lifelong machine learning,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–207, 2018.

[4] G. Wilson and D. J. Cook, “A survey of unsupervised deep domain adaptation,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 5, pp. 1–46, 2020.

[5] B. Settles, “Active learning literature survey,” tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2009.

[6] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[7] Y. He and D. Zhou, “Self-training from labeled features for sentiment analysis,” *Information Processing & Management*, vol. 47, no. 4, pp. 606–616, 2011.

[8] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, 1998.

[9] R. Gilyazev and D. Y. Turdakov, “Active learning and crowdsourcing: A survey of optimization methods for data labeling,” *Programming and Computer Software*, vol. 44, no. 6, pp. 476–491, 2018.

[10] P. Kang, D. Kim, and S. Cho, “Semi-supervised support vector regression based on self-training with label uncertainty: An application to virtual metrology in semiconductor manufacturing,” *Expert Systems with Applications*, vol. 51, pp. 85–106, 2016.

[11] J. Zhu, H. Wang, B. K. Tsou, and M. Ma, “Active learning with sampling by uncertainty and density for data annotations,” *IEEE Transactions on audio, speech, and language processing*, vol. 18, no. 6, pp. 1323–1331, 2009.

[12] Y. Zhang and Q. Yang, “A survey on multi-task learning,” *arXiv preprint arXiv:1707.08114*, 2017.

[13] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[14] Z. Chen, N. Ma, and B. Liu, “Lifelong learning for sentiment classification,” *arXiv preprint arXiv:1801.02808*, 2018.

[15] M. Chen, Z. Xu, K. Weinberger, and F. Sha, “Marginalized denoising autoencoders for domain adaptation,” *arXiv preprint arXiv:1206.4683*, 2012.

[16] S. Hochreiter and J. Schmidhuber, “Lstm can solve hard long time lag problems,” in *Advances in neural information processing systems*, pp. 473–479, 1997.

[17] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” *arXiv preprint arXiv:1801.06146*, 2018.

[18] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[19] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” *arXiv preprint arXiv:1607.01759*, 2016.