Forward and Backward Knowledge Transfer for Sentiment Classification

Hao Wang HWANG@MY.SWJTU.EDU.CN
School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China

Bing Liu LIUB@UIC.EDU
Shuai Wang SHUAIWANGHK@GMAIL.COM
Nianzu Ma JACOBMA005@GMAIL.COM
Department of Computer Science, University of Illinois at Chicago, Chicago, USA

Yan Yang YYANG@SWJTU.EDU.CN
School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China

Editors: Wee Sun Lee and Taiji Suzuki

Abstract

This paper studies the problem of learning a sequence of sentiment classification tasks. The learned knowledge from each task is retained and later used to help future or subsequent task learning. This learning paradigm is called *lifelong learning*. However, existing lifelong learning methods either only transfer knowledge forward to help future learning and do not go back to improve the model of a previous task or require the training data of the previous task to retrain its model to exploit backward/reverse knowledge transfer. This paper studies reverse knowledge transfer of lifelong learning. It aims to improve the model of a previous task by leveraging future knowledge without retraining using its training data, which is challenging now. In this work, this is done by exploiting a key characteristic of the generative model of naïve Bayes. That is, it is possible to improve the naïve Bayesian classifier for a task by improving its model parameters directly using the retained knowledge from other tasks. Experimental results show that the proposed method markedly outperforms existing lifelong learning baselines.

Keywords: Lifelong learning, Sentiment classification, Naïve Bayes, Transfer learning

1. Introduction

Sentiment classification classifies an opinion document (e.g., a product review) as expressing a positive or negative sentiment Pang et al. (2008); Liu (2012). Traditional sentiment classification methods learn from the training data of one task and the learned model is also tested in the same task. In this paper, we are interested in learning a sequence of sentiment classification tasks. This learning setting is the same as the current lifelong learning setting. Lifelong learning is a continual learning (also a continuous learning) process where the learner learns a sequence (possibly never ending) of tasks; after each task is learned, its knowledge is retained and later used to help future task learning Thrun (1998); Silver et al. (2013); Ruvolo and Eaton (2013); Chen and Liu (2014); Chen et al. (2015); Nguyen et al. (2018); Mitchell et al. (2018); Parisi et al. (2019); Anthes (2019). Lifelong learning has been introduced extensively in a book, see Chen and Liu (2018). In this paper, we study

© 2019 H. Wang, B. Liu, S. Wang, N. Ma & Y. Yang.

lifelong learning for sentiment classification. We will discuss why lifelong learning works for sentiment classification in the next section. However, existing approaches mainly focus on helping future task learning by leveraging the knowledge learned from past tasks, which we call *forward knowledge transfer*. If the model of a past task needs to be improved, lifelong learning can also use the knowledge learned in future tasks to help, but the past task's training data are required for retraining.

In this paper, we not only achieve forward knowledge transfer to improve any future task learning, but also achieve *backward/reverse knowledge transfer* to improve the model of any past task *without retraining using the past task training data.* This *enhanced lifelong learning setting* is natural as we humans seem to learn in a similar way. We can use some newly learned knowledge to help improve a previous task without re-learning from the past experiences or data, which are often forgotten. In addition, it would be promising to improve the model of a past task after the model has been learned previously. Following the work in Chen et al. (2015), we treat the classification in each domain (e.g., a type of product) as a learning task. Thus, we use the terms *domain* and *task* interchangeably throughout the paper. Formally, we study the following problem.

Problem Statement: At any point in time, the learner has learned n tasks. Each task T_i has its training data D_i. The learned knowledge of each task is retained in a knowledge base. When faced with a new task T_{n+1} with its training data D_{n+1}, the previously learned knowledge in the knowledge base is leveraged to help learn T_{n+1}. After T_{n+1} is learned, its knowledge is also incorporated into the knowledge base. The knowledge in the knowledge base can also be used to improve the model of each past task T_i ($1 \leq i \leq n$) without retraining using its past training data D_i.

Given the above problem, we have the following questions: (1) what forms of knowledge should be retained from the past learning tasks? (2) how to obtain the knowledge? (3) how to use the knowledge to help future task learning and also to improve the model of any past task without using its past training data for retraining (non-availability of the past training data is a common assumption of continual learning Chen and Liu (2018))?

We answer the above three questions with naïve Bayes by exploiting its *generative model* parameters. We call the proposed method Lifelong Naïve Bayes (LNB). The idea is that the prior knowledge can be mined from the generative model parameters of each past task and used in learning the *target task*, which can be a new or a past task. *These deal with the first two questions.* Another *crucial point* is that the generative model for each task is independent of other tasks and the generative model parameters are available in training the target task model. Thus no retraining is needed when going back to improve the model of any past task. *This deals with the last question.* We will discuss the details of the proposed method in the subsequent sections. To our knowledge, this work is the first to study improving the models of past tasks with no retraining using their past training data.

2. Related Work

2.1. Why Lifelong Learning for Sentiment Classification?

As noted in Chen et al. (2015), lifelong learning works for sentiment classification because the training data of a domain may not be fully representative of this domain due to *sample selection bias* Heckman (1979); Zadrozny (2004). The problem is particularly acute when the training data is small and imbalanced. For example, the word "nice" indicates a positive sentiment, but in the training data of a particular domain, it only appeared once and it is in a negative review because in a negative review, the reviewer can still be positive about some aspects. Furthermore, a positive or negative word may not even appear in the training data of a domain but appear in its test data of this domain. In these cases, the knowledge from other domains can be helpful because most sentiment words have the same polarity across domains. However, as we will see in the experiment section, the result of a classifier built by simply combining the training data from all domains is not satisfactory because each domain often also has some words with domain specific sentiment polarities, e.g., "quiet" is positive for cars but negative for earphones. This problem is called *domain-specific sentiment* problem Li and Zong (2008). The combined data can cause the domain specific sentiment of a word being overshadowed by a different sentiment from other domains.

2.2. Lifelong Learning

Since the proposed LNB is a lifelong learning model, most related work to ours is lifelong learning Ruvolo and Eaton (2013); Chen and Liu (2014); Pentina and Lampert (2015); Chen et al. (2015); Fei et al. (2016); Isele et al. (2016); Xia et al. (2017); Aljundi et al. (2017); Clingerman and Eaton (2017); Shu et al. (2017); Kirkpatrick et al. (2017); Xu et al. (2018); Sun et al. (2018); Chaudhry et al. (2019); Lv et al. (2019). However, these approaches only achieve forward knowledge transfer to improve future task learning and do not achieve backward knowledge transfer to improve the model of any past task without retraining using the past task training data. Regarding sentiment classification, Chen et al. (2015) proposed the first lifelong sentiment classification method based on optimization considering the past knowledge. However, it needs retraining using the past domain data to improve the model. Xia et al. (2017) presented two lifelong learning methods based on voting of individual task classifiers for sentiment classification. The first method votes with equal weight for each task classifier, which can be applied to help past tasks. The second method uses weighted voting, which needs the past task training data for retraining. Note that the tasks in Xia et al. (2017) are actually from the same domain as they partitioned the same dataset into subsets and treated each subset as a task. Our tasks are from different domains (different types of products from Amazon.com). Lv et al. (2019) proposed a deep learning method for lifelong sentiment classification by jointly training two networks, one for retaining domain-specific knowledge and the other for learning target-domain classification feature. However, it cannot improve any past task's model using the knowledge learned subsequently without retraining using the training data of the past task. Moreover, our experimental results (in Section 4) show that this deep learning method does not perform well on small data.

2.3. Multi-task Learning and Transfer Learning

Multi-task learning optimizes the learning of multiple tasks at the same time Caruana (1997); Evgeniou and Pontil (2004); Zhang and Yang (2017); Ruder (2017); Liu et al. (2019). However, (batch) multitask learning does not learn incrementally (incremental multitask learning is regarded as lifelong learning Chen and Liu (2018)).

Transfer learning (or domain adaptation) uses the labeled data from one or more source domains to help the target domain learning which has few or no labeled training data Blitzer et al. (2007); Pan and Yang (2010). Note, transfer learning is not a continual learning process. In sentiment classification, there is a large body of transfer learning literature Tan et al. (2009); Pan et al. (2010); Fan et al. (2011); Glorot et al. (2011); Wu et al. (2017); Li et al. (2018); He et al. (2018), to name a few. These works cannot go back to improve the source domains as the target domain has few or even no labeled data.

3. Lifelong Naïve Bayes

In this section, we present the proposed Lifelong Naïve Bayes (LNB). We first give some preliminaries, and then detail how our LNB performs forward and backward/reverse knowledge transfer to help the future task and past tasks without the training data of the past tasks. Lastly, we discuss the computational complexity of the proposed algorithm.

3.1. Preliminaries

Naïve Bayes (NB) text classifier is a generative model Nigam et al. (1998). Each document is assumed to be generated from a mixture of multinomial distributions. In training, NB computes the parameters of the multinomial distributions. In testing, given a test document d with words $\{w_1, ..., w_n\}$, the NB classifier with multinomial distributions is defined as:

$$P(c_j|d) = \frac{P(c_j)\prod_{k=1}^{n} P(w_k|c_j)}{\sum_{r=1}^{|C|} P(c_r)\prod_{k=1}^{n} P(w_k|c_r)}, \tag{1}$$

where c_j is positive (+) or negative (-) class in our case, and $|C|$ is the number of classes. The multinomial distribution parameter $P(w_k|c_j)$ is estimated with:

$$P(w_k|c_j) = \frac{\lambda + N_{c_j,w_k}}{\lambda|V| + \sum_{v=1}^{|V|} N_{c_j,w_v}}, \tag{2}$$

where N_{c_j,w_k} is the number of times that word w_k occurs in the training documents of class c_j, $|V|$ is the vocabulary V size, and λ is the smoothing parameter. We use $\lambda = 0.1$ as it is shown in Agrawal et al. (2000) that $\lambda = 0.1$ (Lidstone smoothing) is superior to $\lambda = 1$ (Laplacian smoothing).

The final sentiment polarity (positive or negative) of this document d is given by computing $\arg\max_j P(c_j|d)$. From Eq. (1) and Eq. (2), we can see that NB mainly depends on the word frequency count N_{c_j,w_k}, which is also the core *knowledge* that we will retain for each domain in addition to the class prior $P(c_j)$. How to integrate the knowledge into the proposed model will be clear shortly.

3.2. LNB for Sentiment Classification

Recall that we use the terms domain and task interchangeably throughout the paper because we treat the classification in each domain (e.g., a type of product) as a learning task following Chen et al. (2015). The architecture of the proposed LNB, in brief, is shown in Figure 1.

Below we present the proposed LNB in detail. As given in Figure 1, LNB has three key components: Knowledge Miner (KM), Knowledge Base (KB), and Knowledge-Based Learner (KBL). KM mines knowledge from training data of each task/domain, KB stores the mined knowledge, and KBL abstracts some high-level knowledge from the KB and leverages it in learning the *target task* t, which can be a new task or a previous task.

The key idea of LNB is to revise the multinomial distribution parameters (i.e., Eq. (2)) for the target task t using prior knowledge (stored in the KB) from the previous tasks to build a better target task classifier. Next we describe how our system LNB works.

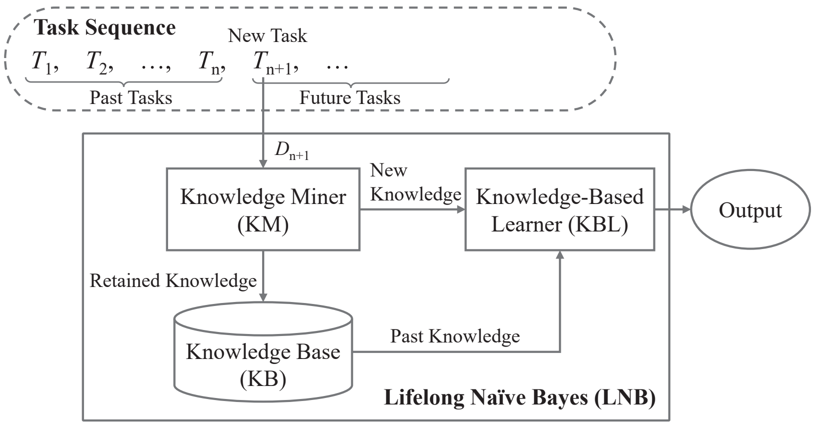

Figure 1: LNB system architecture.

Given a task sequence $\{T_1, T_2, ..., T_n, T_{n+1}, ...\}$, the time point moves along the task sequence, where each time point corresponds to a task respectively. At any point (e.g., $n+1$) in time, our LNB has learned n tasks and the knowledge of the n tasks has been retained in the KB. When a new/future task (target task) T_{n+1} comes, KM first extracts the new knowledge from its training data D_{n+1}. Then KBL leverages both the newly extracted knowledge and the past knowledge in the KB to build a classifier for the target task T_{n+1}. After this, the task T_{n+1} becomes a past task and its knowledge is also incorporated into the KB. Now if our model LNB moves back to a point of any previous time point i ($1 \leq i \leq n$), our LNB can use the knowledge (stored in the KB) of the tasks $(T_1, ..., T_{n+1})$ to build a classifier for the past task (target task) T_i. In such a way, we can see that LNB performs (1) forward knowledge transfer for helping future task learning and (2) reverse knowledge transfer for improving past task learning.

In particular, KM extracts two types of knowledge from the training data of each task T_i. (a) **Word-level knowledge $N^i_{+,w}$ and $N^i_{-,w}$**: number of times word w occurs in the training documents of the positive (+) and negative (−) class in the task T_i, respectively. (b) **Document-level knowledge N^i_+ and N^i_-**: number of training documents in the positive (+) and negative (−) class in task T_i, respectively. These document-level knowledge are for computing the class prior $P_i(+)$ and $P_i(-)$ in Eq. (1).

The above two-types of knowledge are treated as the *base-knowledge*, which will be stored in the KB for the task T_i. All the stored base-knowledge in the KB is available for KBL to compute the generative model parameters and to build a NB classifier for the target domain t (a new or a past domain).

Now we introduce how KBL utilizes the newly extracted base-knowledge (if target domain is a new domain) and the past base-knowledge (stored in the KB) to construct a classifier for the target domain t. Specially, KBL first abstracts *three types of high-level knowledge* from the base-knowledge:

(1) Word-level knowledge $N^{KB}_{+,w}$ and $N^{KB}_{-,w}$: number of times word w occurs in the training documents of the positive (+) and negative (-) class in all domains except the target domain. They are formulated by $N^{KB}_{+,w} = \sum_f N^f_{+,w}$ and $N^{KB}_{-,w} = \sum_f N^f_{-,w}$ respectively.

(2) Target domain-dependent knowledge $Q^t_{+,w}$ and $Q^t_{-,w}$: ratio of word probability in positive (and negative) vs. negative (positive) class in the target domain t, i.e., $Q^t_{+,w} = \frac{P(w|+)}{P(w|-)}$ and $Q^t_{-,w} = \frac{P(w|-)}{P(w|+)}$.

(3) Domain-level knowledge $M^{KB}_{+,w}$ and $M^{KB}_{-,w}$: number of non-target domains in which $\frac{P(w|+)}{P(w|-)} \geq \gamma$ and $\frac{P(w|-)}{P(w|+)} \geq \gamma$ (regarded as a reliable word), where $P(w|+)$ and $P(w|-)$ are estimated using Eq. (2) in that domain, and γ is a parameter.

Then, KBL integrates these pieces of knowledge to revise $N^t_{+,w}$ and $N^t_{-,w}$ from the target domain, which are the word count in the positive (+) and negative (-) classes in the target domain respectively. We denote the revised results as $\hat{N}^t_{+,w}$ and $\hat{N}^t_{-,w}$. The intuition here is that if a word w can distinguish classes very well in the target domain, KBL in our LNB should rely on the target domain base-knowledge (i.e., $N^t_{+,w}$ and $N^t_{-,w}$) of this word in the target domain. Thus, we define a set of target domain-dependent words, denoted by V^t. A word w belongs to V^t if $Q^t_{+,w} \geq \sigma$ or $Q^t_{-,w} \geq \sigma$, where σ is a parameter. On the other hand, KBL should believe in domain-level knowledge among non-target domains. In practice, if a word w is reliable among most non-target domains (e.g., half of the non-target domains), KBL should follow the knowledge associated with this word. Similarly, we define a set of domain-reliable words, denoted by V^{KB}. For a word w, KBL regards it as a domain-reliable word if more than half of the non-target domains (i.e., $M^{KB}_{+,w} > n/2$ or $M^{KB}_{-,w} > n/2$, where n is the number of the non-target domains) treat this word as a reliable word. Such a method works like a voting method. We will give an example of the resulting domain-dependent words and domain-reliable words in the experiment section.

Improving past and future domain classification: It is clear that LNB can treat any past or future domain as the target domain t and improve its classification. LNB only needs the frequency count of each word in each class of each (past or future) domain (which is stored in the KB). Thus, for a past domain, no retraining using its original training data is needed. Given these knowledge and information, our LNB model works for a test document d_u with words $\{w_1, w_2, ..., w_n\}$ in the target domain t as shown in Algorithm 1.

3.3. Computational Complexity Analysis

Since LNB is a continuous learning system, we use the target domain model building to analyze its complexity. If the target domain is a new domain, the computation complexity of LNB consists of two parts: *extracting base-knowledge from its training data* and *building a classifier for this new domain.* Specially, extracting the base-knowledge is clearly linear in the number of words in all training documents, $\mathcal{O}(|D||d|)$, where $|D|$ is the number of training documents and $|d|$ is the document length. Building a classifier for this new domain is only $\mathcal{O}(|V|)$, where $|V|$ is the vocabulary size, because the two types of high-level knowledge (i.e., word-level knowledge and domain-level knowledge) can be incrementally updated after each new domain is learned. Note that the computing of another high-level knowledge (i.e., target domain-dependent knowledge) is done in the first part. If the target domain is a past domain, LNB only needs to build a classifier for this past domain, which takes $\mathcal{O}(|V|)$. Considering the above two cases together, the computational complexity of LNB is $\mathcal{O}(|D||d|)$, linear in the number of words in the documents as $|V|$ is normally smaller.

As LNB stores base-knowledge for each task, LNB requires some storage costs. Storage cost of each task is $O(|V|)$. Thus, storage costs of LNB are linear in the number of tasks.

Algorithm 1 LNB for a test document d_u in the target domain t (a new or a past domain)

```
Data: Test document $d_u$ with words $\{w_1, w_2, ..., w_n\}$
Knowledge: Base-knowledge stored in the knowledge base, and the newly extracted base-
           knowledge (if target domain is a new domain)
Result: Sentiment classification result
begin
    Extract uni-gram features for words $\{w_1, w_2, ..., w_n\}$
    for each feature word $w_k$ do
        if $w_k$ belongs to $V^{KB}$ then
            $\hat{N}^t_{+,w_k} = R_{w_k} \times N^{KB}_{+,w_k}$, $\hat{N}^t_{-,w_k} = (1 - R_{w_k}) \times N^{KB}_{-,w_k}$
            /* where $R_{w_k}$ is a voting weight, $R_{w_k} = M^{KB}_{+,w_k}/(M^{KB}_{+,w_k} + M^{KB}_{-,w_k})$ */
        else if $w_k$ belongs to $V^t$ then
            $\hat{N}^t_{c_j,w_k} = N^t_{c_j,w_k}$
        else
            $\hat{N}^t_{c_j,w_k} = N^{KB}_{c_j,w_k} + N^t_{c_j,w_k}$
        end
    end
    $\arg\max_j P(c_j|d_u)$, where $c_j = \{+, -\}$
end
```

4. Experiments

In this section, we evaluate the proposed LNB in terms of knowledge analysis, task evaluation, effect of the number of domains, and running time. We performed most experiments on Windows Server 2008 R2 with Intel Xeon processor, 24GB RAM and JAVA development environment. We used JAVA development environment because our main baselines (except for a deep learning baseline) used it. Regarding the deep learning baseline Lv et al. (2019), we ran it on a GPU server with Python as its original system used Python.

4.1. Datasets

Since the main baseline of our work is the LSC system Chen et al. (2015), we experiment using the same datasets [1] as in Chen et al. (2015). The datasets contains a collection of product reviews from 20 types of product domains from Amazon.com. Each domain contains 1000 reviews. Each review has been assigned a sentiment label, i.e., positive or negative, based on the rating score. The names of these 20 domains with a serial number for each domain and the proportion of negative reviews are shown in Table 1. From the table, we can see that the proportion of negative reviews in each domain is in the range [10%, 31%]; and thus the negative reviews in each domain are minorities, which are harder to classify.

Following the work in Chen et al. (2015), we use two variations of the datasets with different class distributions in our experiments. (a) ***Natural class distribution***: We

1. https://www.cs.uic.edu/~zchen/downloads/ACL2015-Chen-Datasets.zip

Table 1: Names of 20 domains with a serial number for each domain and the proportion of negative reviews in each domain.

① Alarm Clock	30.51	⑥ Flashlight	11.69	⑪ Home Theater System	28.84	⑯ Projector	20.24
② Baby	16.45	⑦ GPS	19.50	⑫ Jewelry	12.21	⑰ Rice Cooker	18.64
③ Bag	11.97	⑧ Gloves	13.76	⑬ Keyboard	22.66	⑱ Sandal	12.11
④ Cable Modem	12.53	⑨ Graphics Card	14.58	⑭ Magazine Subscriptions	26.88	⑲ Vacuum	22.07
⑤ Dumbbell	16.04	⑩ Headphone	20.99	⑮ Movies TV	10.86	⑳ Video Games	20.93

use the original data with the natural class distribution as shown in Table 1. We use F-score of the negative class in evaluation as each domain has imbalanced class distribution and the reviews of each domain in the negative class are minorities. (b) ***Balanced class distribution***: We sampled 200 reviews (100 positive and 100 negative) from each domain and created a balanced variation from the original natural distribution dataset. This dataset is small because the number of negative reviews in each domain is small. We use accuracy of both classes in evaluation as each domain has a balanced class distribution.

We randomly partitioned the data of each domain into training set and test set with 80% and 20%, respectively. We extracted uni-gram features with no feature selection from the raw reviews. Also, we followed Pang et al. (2002) to deal with negation words. In all experiments, we set $\gamma = 2$ and $\sigma = 3$ to evaluate the proposed LNB.

4.2. Knowledge Analysis

As mentioned earlier, we will provide the *domain-dependent words* V^t and *domain-reliable words* V^{KB} learned by our LNB in the experiment section. Here we take the small dataset in balanced class distribution as an example. The results are shown in Table 2, where the words appearing in both V^t and V^{KB} are marked in *italics* together with color "blue". From the table, we have the following observations:

1. Generally speaking, the words in V^{KB} are more reliable than the words in V^t in terms of their opinion polarity. The reason is clear because the word in V^{KB} is believed by more than half of the non-target domains.

2. Most words (the ratio is 561/772)[2] in V^t are specific to the target domain. Only a small portion (the ratio is 211/772)[3] of V^t belong to the domain-reliable words V^{KB}. Thus, V^t and V^{KB} should be considered simultaneously because they are complementary to each other. The result also shows the domain-specific sentiment problem as mentioned in Section 1.

4.3. Task Evaluation

Now we evaluate how the future task learning performance and the previous task learning performance can be improved using the proposed model LNB.

2. According to Table 2, $\frac{405-91+367-120}{405+367} = \frac{561}{772}$.
3. According to Table 2, $\frac{91+120}{405+367} = \frac{211}{772}$.

Table 2: The resulting domain-dependent words and domain-reliable words on the dataset in the balanced class distribution.

Positive (+) words	Negative (-) words
Domain-dependent words V^t in domain "Alarm Clock" (vocabulary size: **1984**)	
Results in descending order of $Q^t_{+,w}$: *worth*, *great*, *highly*, *recommend*, *perfect*, *love*, *impressed*, *not|beat*, *durability*, *setup*, adjustable, thick, kit, boat, downside, not|forget, heaven, definately, picky, not|happier, satellite, plant, moderately, contrary, tour, not|overly, role, digit, havent, projector, attend, sufficiently, sensor, not|leg, sensitivity, camping, not|reset, exterior, projection, seperate, purple, schedule, island, delighted, not|bear, converter, orient, horn, luggage, smoke, japan, execute, guest, not|demand, alike, not|sheet, northern, not|disturb, sunshine, skill, adorable, grateful, not|class, not|energy, wooden, lightest, not|blinking, lodging, readable, not|nap, not|bedroom, thunderstorm, underside, manageable, distraction, not|scare, wisconsin, not|saver, stuffer, girly, not|rain, gigantic, grandaughter, tide, not|deep, nighttime, not|opportunity, legible, not|overwhelming, cleverly, litte, not|halfway, awake, lull, sincere, lithium, grandchild, not|proof, dresser, clap, ... (in total **405** words).	**Results in descending order of $Q^t_{-,w}$:** *return*, *not|buy*, *poor*, *not|waste*, *refund*, *not|return*, *terrible*, *horrible*, *junk*, *garbage*, *quit*, restart, not|suppose, reset, not|avail, resort, not|junk, distracting, hum, not|blue, not|market, times, not|barely, intermittently, not|clock, buzz, tuning, leap, bum, backlight, inspection, boo, illumination, horrify, bleed, not|functionality, coil, nuisance, oct, reinstall, not|terribly, nope, not|credit, not|worst, not|appearance, not|twenty, not|functional, drift, chirp, interval, not|snooze, aus, rhythm, relocate, ineffective, not|backup, not|photo, spiffy, unreadable, critic, annoyingly, compound, bearly, not|inadvertently, forgive, embarrass, timex, yup, consequence, realy, mailing, not|task, babble, module, oxidation, embarrassment, abruptly, voicemail, grainy, not|sun, optimistic, not|coby, not|faint, temperamental, inaudible, vague, not|weak, abysmal, not|reliably, forgivable, nt, cliff, *not|worth*, *company*, *disappoint*, *not|money*, *bad*, *not|product*, *spend*, *disappointed*, ... (in total **367** words).
Domain-reliable words V^{KB} over 20 domains (accumulated vocabulary size: **16061**)	
Sorted words in terms of $M^t_{+,w}$ from big to small: *perfect*, add, *easy*, *worth*, *good*, decide, *bit*, *night*, *enough*, like, *job*, *great*, *love*, *highly*, simply, *fast*, *size*, *long*, live, *recommend*, *happy*, *not|need*, *excellent*, *glad*, *value*, expect, *need*, *best*, *nice*, *price*, *well*, *fit*, true, simple, *complaint*, house, *home*, set, *wonderful*, solid, *easier*, *friend*, *awesome*, mind, feature, *amazing*, bring, not|big, *complain*, cheaper, *negative*, difference, drop, *quick*, *enjoy*, space, amount, pick, heavy, *extra*, *color*, perfectly, *life*, offer, lower, clear, *easily*, *power*, larger, person, fantastic, pro, *provide*, *older*, *thank*, *ease*, foot, investment, deliver, *strong*, *favorite*, *impressed*, *available*, test, *daily*, *construction*, amaze, *pleased*, insert, *handy*, bigger, *not|beat*, weight, special, blue, *access*, *useful*, family, *carry*, *eye*, ... (in total **219** words).	**Sorted words in terms of $M^t_{-,w}$ from big to small**: *month*, *product*, *not|time*, *not|purchase*, *money*, *not|work*, *break*, *amazon*, *not|worth*, day, *not|buy*, item, not|well, *poor*, *happen*, state, *not|good*, *not|recommend*, *completely*, *piece*, receive, *bad*, *not|make*, *guess*, call, buy, *not|money*, *ok*, *customer*, *waste*, not|enough, suppose, *problem*, *return*, *end*, pay, *contact*, *disappoint*, *not|product*, *company*, *week*, *wrong*, *disappointed*, *spend*, *back*, order, *refund*, cheap, *not|happen*, *send*, save, dollar, couple, *stop*, lose, hard, design, *fail*, review, issue, notice, throw, wait, service, low, *unfortunately*, *not|waste*, today, *brand*, stick, *hour*, ship, not|back, *begin*, rating, *not|anymore*, *horrible*, *die*, *not|hold*, actual, *barely*, total, *terrible*, *annoying*, *useless*, *description*, *suck*, *not|case*, *not|return*, not|star, *mail*, *worse*, *worst*, not|amazon, not|close, *pain*, *wonder*, *poorly*, *junk*, cut, ... (in total **258** words).
The number of positive co-occurrence words in V^t and V^{KB} is **91**.	The number of negative co-occurrence words in V^t and V^{KB} is **120**.

* Due to space limitations, here we only show the top 100 words in each block.

Baselines. We compare our LNB model with Naïve Bayes (NB), SVM Chang and Lin (2011), LSC Chen et al. (2015), Lifelong Voting (LLV) Xia et al. (2017), and Sentiment classification by leveraging the Retained Knowledge (SRK) Lv et al. (2019). For SVM, LSC and SRK, we obtained the original systems from its authors. For LLV, we use its first voting method that can improve a past model using future knowledge. Note that traditional NB and SVM only work on a single domain data. To have a comprehensive comparison, three variations of NB and SVM are created respectively following Chen et al. (2015):

a) NB and SVM are trained and tested on the target domain, denoted by **NB-T** and **SVM-T**.

b) NB and SVM are trained on the combined data from all non-target domains and tested on the target domain, denoted by **NB-S** and **SVM-S**.

c) NB and SVM are trained on the combined data from all domains (including the target domains) and tested on the target domain, denoted by **NB-ST** and **SVM-ST**.

As NB-T and SVM-T only use the data from the target task, they are non-lifelong learning methods. NB-S, NB-ST, SVM-S and SVM-ST can be regarded as simple lifelong methods because they use the data from other tasks. For NB, λ is set to 0.1 for Lidstone smoothing. For SVM, we use the default parameters settings [4]. For LSC, LLV and SRK, we use their original parameters settings. Note that, for the deep learning baseline SRK, we didn't use validation sets to perform early stop as all the other methods do not require any validation sets.

New Task Evaluation: Following Chen et al. (2015), each domain in the 20 domains is treated as the new (target) domain with the rest 19 domains as the past domains. For those non-lifelong methods (NB-T and SVM-T), we ran them on the new domain data independently. For those simple lifelong methods (NB-S, NB-ST, SVM-S and SVM-ST), we ran them on the combined data from past domains or new domain. For the three existing lifelong methods (LSC, LLV and SRK), we ran them as they were done in their original papers. Table 3 shows the average F1-scores of the negative class in the natural class distribution over the 20 domains. Table 4 shows the average accuracy of two classes in the balanced class distribution over the 20 domains.

Table 3: Average F1-score of the negative class in the natural class distribution over 20 domains. Note that negative class is the minority class (see Table 1) and thus very harder to classify.

Method	NB-T	NB-S	NB-ST	SVM-T	SVM-S	SVM-ST	LSC	LLV	SRK	LNB
F1-score	45.20	55.00	56.49	50.39	52.66	59.15	56.62	47.46	52.45	**64.96**

Table 4: Average accuracy of the positive and negative classes in the balanced class distribution over 20 domains.

Method	NB-T	NB-S	NB-ST	SVM-T	SVM-S	SVM-ST	LSC	LLV	SRK	LNB
ACC	77.40	74.82	80.04	76.09	75.79	79.29	82.09	78.59	80.34	**83.17**

From Table 3 and Table 4, we make the following observations:

- Our model LNB achieves the best F1-scores and accuracy on two variant sets of the datasets respectively. The results show the superiority of our LNB.
- NB-S (and SVM-S) is inferior to NB-T (and SVM-T), both of which are inferior to NB-ST (and SVM-ST). This shows that simply combining the training data from all past domains and the new domain is slightly beneficial, but worse than our model.

4. See: `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

- LLV performs poorly as its voting method does not fit our setting because in Xia et al. (2017) all tasks are from the same domain (the dataset of each task is a subset of a large dataset), but our tasks are from different domains.
- The deep learning baseline SRK is inferior to our method as deep learning usually requires a large amount of training data, but the training data of each domain in our setting is very small. Another reason is probably due to the catastrophic forgetting problem French (1999) in deep learning. Although SRK can address this problem slightly by training a knowledge retention network to memorize domain-specific knowledge, it still forgets previously learned knowledge after learning a long sequence of domains (e.g., 15). Our model has a knowledge base to store the previously learned knowledge from each domain separately.
- Our model is slightly better than LSC on the dataset in the balanced class distribution, but markedly better than LSC on the dataset in the natural class distribution [5]. The reason is that LSC extracts knowledge with bias from non-target domains. Our model adopts a voting method to extract knowledge. In practice, we follow the knowledge associated with a word only if more than half of non-target domains believe in this word. Note that this is not our main results as our main goal is to go back to help previous/past domain models without retraining, which LSC cannot do because LSC needs the past domain data to optimize its objective function.

Previous Task Evaluation: In this experiment, we evaluate how each previous domain performs after some new/future domains have been learned. Since LSC and SRK cannot use the future knowledge to improve past domain models without retraining, for each past domain, we use the classifier built when the past domain was the new domain at that time. For NB-S, NB-ST, SVM-S and SVM-ST, we also use the classifier built at that time. For NB-T and SVM-T, we use the classifier built on each previous domain. Since LLV is a voting model, it can also use future models to vote in any past domain classification. We give the results after all 20 domains have been learned. Thus, in this setting, we show the test results of the previous 19 domains only. Since in this case the ordering of domains is significant and can affect the experiment results, we randomly created 10 domain sequences as shown in Figure 2. For each sequence, the test results of the previous 19 domains are averaged and the average values are reported in Table 5 and Table 6.

S1:	⑰	⑦	⑯	⑪	⑩	⑬	⑫	⑥	⑤	⑧	⑱	④	⑭	②	⑨	①	⑮	⑲	③	⑳
S2:	⑭	⑬	⑲	⑩	②	⑫	⑳	①	⑯	⑤	⑪	④	⑧	⑨	②	⑰	⑦	⑥	⑮	⑱
S3:	⑤	⑦	⑰	⑨	④	⑯	⑮	⑫	⑩	⑭	⑪	⑧	⑲	⑱	⑥	⑬	②	③	⑳	①
S4:	⑭	④	⑳	⑦	⑰	⑮	⑯	⑧	⑬	⑨	⑫	⑥	②	③	⑤	⑩	⑱	⑪	①	⑲
S5:	⑤	⑯	②	⑳	⑱	⑧	⑬	④	⑥	⑨	⑩	⑲	⑦	③	⑪	①	⑭	⑮	⑫	⑰
S6:	⑮	⑪	④	⑳	⑰	③	⑦	⑩	⑯	⑫	⑱	①	②	⑬	⑤	⑧	⑲	⑥	⑨	⑭
S7:	⑳	⑬	②	⑮	⑨	⑰	⑭	⑤	⑯	⑱	⑦	④	⑪	⑥	①	⑧	⑩	③	⑲	⑫
S8:	⑱	⑪	⑥	⑫	⑬	①	⑤	⑦	③	⑯	②	④	⑭	⑧	⑮	⑲	⑰	⑩	⑨	⑳
S9:	⑯	②	⑩	⑨	⑫	⑲	⑥	⑪	⑱	⑳	④	①	③	⑰	⑤	⑬	⑮	⑧	⑭	⑦
S10:	⑫	⑯	⑥	⑧	⑲	②	⑦	①	⑬	⑰	③	⑨	④	⑪	⑱	⑮	⑳	⑤	⑩	⑭

Figure 2: The randomly created 10 domain sequences.

From Table 5 and Table 6, we clearly see that LNB again outperforms all baselines on the two datasets. Although in helping future domain learning on the dataset in the balanced

5. In Table 3, the F1-scores (56.62) for LSC is not the same as that reported in Chen et al. (2015) because we used 80% reviews of each past domain for training while Chen et al. (2015) used all reviews for training because we need to test on past domains, while Chen et al. (2015) does not do that.

Table 5: Average F1-score of the negative class (in natural class distribution) over 19 previous domains for each sequence.

Task-Sequence	NB-T	NB-S	NB-ST	SVM-T	SVM-S	SVM-ST	LSC	LLV	SRK	LNB
S1	43.04	49.66	54.25	51.16	50.21	57.34	54.89	44.25	51.43	**63.80**
S2	44.42	51.16	51.98	51.80	51.13	58.62	53.93	44.32	48.69	**63.56**
S3	42.35	49.37	52.29	49.99	46.70	59.60	51.80	44.76	50.78	**63.62**
S4	42.64	42.22	45.07	50.60	45.22	54.90	50.15	43.87	48.54	**63.66**
S5	42.65	46.16	51.90	50.05	49.54	59.35	48.43	44.46	47.82	**63.39**
S6	42.48	45.56	52.10	50.50	49.30	60.16	53.45	43.83	51.74	**63.64**
S7	45.63	46.28	52.21	51.44	50.15	59.29	52.02	46.78	50.31	**65.09**
S8	43.05	50.74	51.34	51.16	48.96	57.52	52.64	44.25	50.56	**63.80**
S9	43.15	49.61	50.78	50.72	48.41	58.08	54.14	43.15	51.11	**63.30**
S10	42.48	51.23	52.26	50.50	48.95	57.11	51.05	43.83	49.25	**63.64**
(↑) Ave.[1]	43.19	48.20	51.42	50.79	48.86	58.20	52.25	44.35	50.02	**63.75**

[1] (↑) Ave. denotes the average value over the above 10 task sequences (S1, ..., S10).

Table 6: Average accuracy of two classes (in balanced class distribution) over 19 previous domains for each sequence.

Task-Sequence	NB-T	NB-S	NB-ST	SVM-T	SVM-S	SVM-ST	LSC	LLV	SRK	LNB
S1	78.81	71.84	77.76	76.57	73.42	80.26	81.84	79.47	80.29	**85.26**
S2	78.15	71.71	78.55	76.18	69.73	78.68	81.97	80.26	81.17	**84.74**
S3	78.55	73.02	79.86	75.92	71.31	78.55	81.18	79.21	79.96	**84.60**
S4	78.29	70.00	78.55	75.79	69.34	76.97	79.99	79.07	79.84	**84.74**
S5	78.16	71.97	77.50	75.65	71.97	80.26	80.65	78.94	79.52	**84.87**
S6	78.68	68.55	80.26	75.92	67.89	76.71	81.84	79.60	80.01	**85.26**
S7	78.68	69.73	77.76	75.52	69.34	77.23	79.21	79.34	80.25	**84.47**
S8	78.81	71.44	80.39	76.57	71.84	79.21	82.36	79.47	81.13	**85.26**
S9	78.95	71.97	79.21	76.71	71.05	80.26	81.57	79.60	81.65	**85.26**
S10	78.68	72.10	78.29	75.92	71.97	80.92	80.39	79.60	80.30	**85.26**
(↑) Ave.	78.57	71.23	78.81	76.07	70.78	78.90	81.10	79.46	80.41	**84.97**

class distribution, LNB is only slightly better than LSC (see Table 4), the ability of LNB to improve past domain models using future knowledge clearly shows its superiority to LSC.

4.4. Effect of Number of Future Domains

As our main contribution is to go back to help the past domain models without retraining, here we empirically evaluate the effects of our model using different number of new/future domains (denoted by #future domains). In this experiment, we treat the first domain in each domain sequence (see Figure 2) as the target domain and vary the number of future domains (max is 19). For each domain sequence, we record the test results with different number of future domains. The curve of the average test results over 10 domain sequences is shown in Figure 3. The curve clearly shows that LNB performs better with more future domains. This indicates that LNB indeed has the ability to go back to improve the past domain models using future knowledge.

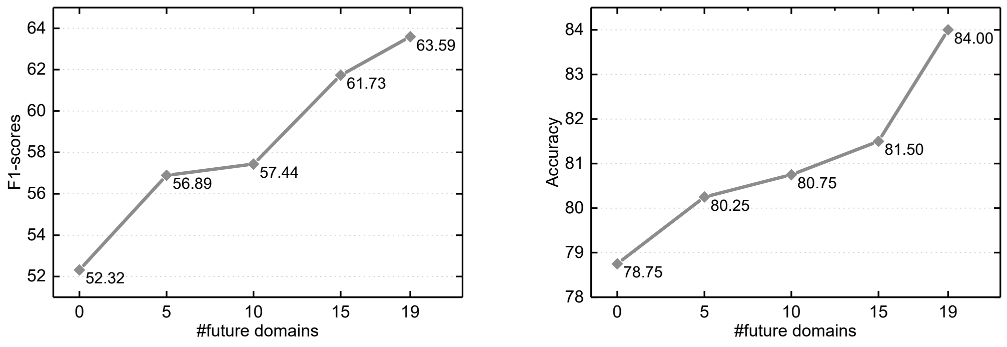

Figure 3: Effect of #future domains on LNB. (Left): F1-scores of negative class with #future domains in the natural class distribution. (Right): Accuracy of two classes with #future domains in the balanced class distribution.

4.5. Running Time

For the running time experiment, we take domain sequence S1 in Figure 2 as an example. We recorded the running time of each method (except for SRK because SRK used a different running environment than the others as introduced early) in the previous task evaluation. Due to the space limitation, here we only give the results on the dataset in the natural class distribution because the results for the other are similar. The results are shown in Table 7. From the table, we can see that LNB performs more efficiently than all lifelong learning methods including those simply methods NB-S, NB-ST, SVM-S and SVM-ST. The reasons are 2-fold: (1) LNB does not need iterative optimization, whereas SVM-S, SVM-ST and LSC need. (2) LNB accumulates knowledge from each new domain. Updating the knowledge takes $\mathcal{O}(|V|)$, where $|V|$ is the vocabulary size, as discussed in the computational complexity section. NB-S, NB-ST, SVM-S, SVM-ST and LLV do not accumulate knowledge. That is, they need to retrain for each target domain.

Table 7: Running time (in seconds) of each method on the dataset in the natural class distribution.

Method	NB-T	NB-S	NB-ST	SVM-T	SVM-S	SVM-ST	LSC	LLV	LNB
running time	7.915	21.309	28.035	12.011	230.070	273.800	45.211	312.641	11.470

5. Conclusions

This paper studied a new lifelong learning setting where the system can use the knowledge learned in future tasks to improve past task models with no retraining using the training data of the past tasks. We proposed a novel approach in this new lifelong learning setting for sentiment classification by exploiting the generative model parameters of naïve Bayes. Extensive experiment results on 20 product domains from Amazon.com showed the effectiveness of the proposed approach. We believe that this new setting is a promising direction for lifelong learning because we humans often learn new knowledge to solve past problems and future problems.

Acknowledgments

Hao Wang and Yan Yang's work was partially supported by by the National Natural Science Foundation of China under grand nos. 61976247 and 61572407, and the Project of National Science and Technology Support Program under grand no. 2015BAH19F02.

References

Rakesh Agrawal, Roberto Bayardo, and Ramakrishnan Srikant. Athena: Mining-based interactive management of text databases. In *EDBT*, pages 365–379, 2000.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, pages 3366–3375, 2017.

Gary Anthes. Lifelong learning in artificial neural networks. *Communications of the ACM*, 62(6):13–15, 2019.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, pages 440–447, 2007.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *ICLR*, pages 1–20, 2019.

Zhiyuan Chen and Bing Liu. Topic modeling using topics from many domains, lifelong learning and big data. In *ICML*, pages 703–711, 2014.

Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

Zhiyuan Chen, Nianzu Ma, and Bing Liu. Lifelong learning for sentiment classification. In *ACL*, pages 750–756, 2015.

Christopher Clingerman and Eric Eaton. Lifelong learning with gaussian processes. In *ECML/PKDD*, pages 690–704, 2017.

Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *KDD*, pages 109–117, 2004.

Wen Fan, Shutao Sun, and Guohui Song. Probability adjustment naïve bayes algorithm based on nondomain-specific sentiment and evaluation word for domain-transfer sentiment analysis. In *the 18th International Conference on Fuzzy Systems and Knowledge Discovery*, volume 2, pages 1043–1046, 2011.

Geli Fei, Shuai Wang, and Bing Liu. Learning cumulatively to become more knowledgeable. In *KDD*, pages 1565–1574, 2016.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520, 2011.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. Adaptive semi-supervised learning for cross-domain sentiment classification. In *EMNLP*, pages 3467–3476, 2018.

James J Heckman. Sample selection bias as a specification error. *Econometrica: Journal of the Econometric Society*, pages 153–161, 1979.

David Isele, Mohammad Rostami, and Eric Eaton. Using task features for zero-shot knowledge transfer in lifelong learning. In *IJCAI*, pages 1620–1626, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Shoushan Li and Chengqing Zong. Multi-domain sentiment classification. In *ACL-HLT*, pages 257–260, 2008.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. Hierarchical attention transfer network for cross-domain sentiment classification. In *AAAI*, pages 5852–5859, 2018.

Bing Liu. Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

Guangyi Lv, Shuai Wang, Bing Liu, Enhong Chen, and Kun Zhang. Sentiment classification by leveraging the shared knowledge from a sequence of domains. In *DASFAA*, pages 795–811, 2019.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bo Yang, Justin Betteridge, Andrew Carlson, B Dalvi, Matt Gardner, Bryan Kisiel, et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *ICLR*, pages 1–18, 2018.

Kamal Nigam, Andrew McCallum, Sebastian Thrun, Tom Mitchell, et al. Learning to classify text from labeled and unlabeled documents. *AAAI/IAAI*, pages 792–799, 1998.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Cross-domain sentiment classification via spectral feature alignment. In *WWW*, pages 751–760, 2010.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86, 2002.

Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Anastasia Pentina and Christoph H Lampert. Lifelong learning with non-iid tasks. In *NIPS*, pages 1540–1548, 2015.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Paul Ruvolo and Eric Eaton. ELLA: An efficient lifelong learning algorithm. In *ICML*, pages 507–515, 2013.

Lei Shu, Hu Xu, and Bing Liu. Lifelong learning CRF for supervised aspect extraction. In *ACL*, pages 148–157, 2017.

Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, volume 13, pages 48–55, 2013.

Gan Sun, Yang Cong, and Xiaowei Xu. Active lifelong learning with "watchdog". In *AAAI*, pages 4107–4114, 2018.

Songbo Tan, Xueqi Cheng, Yuefen Wang, and Hongbo Xu. Adapting naive bayes to domain adaptation for sentiment analysis. In *European Conference on Information Retrieval*, pages 337–349, 2009.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to Learn*, pages 181–209. Springer, 1998.

Fangzhao Wu, Yongfeng Huang, and Jun Yan. Active sentiment domain adaptation. In *ACL*, pages 1701–1711, 2017.

Rui Xia, Jie Jiang, and Huihui He. Distantly supervised lifelong learning for large-scale social media sentiment analysis. *IEEE Transactions on Affective Computing*, 8(4):480–491, 2017.

Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. Lifelong domain word embedding via meta-learning. In *IJCAI*, pages 4510–4516, 2018.

Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *ICML*, page 114, 2004.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.