Weakly-Supervised Deep Learning for Customer Review Sentiment Classification

Ziyu Guan[1], Long Chen[1], Wei Zhao[2], Yi Zheng[3], Shulong Tan[4], Deng Cai[3]
[1]Northwest University [2]Xidian University [3]Zhejiang University [4]Baidu Big Data Laboratory
Xi'an, China Xi'an, China Hangzhou, China Sunnyvale, USA
[1]{ziyuguan@, longchen@stumail.}nwu.edu.cn, [2]ywzhao@mail.xidian.edu.cn
[3]{delostik@, dengcai@cad.}zju.edu.cn, [4]laos1984@gmail.com

Abstract

Sentiment analysis is one of the key challenges for mining online user generated content. In this work, we focus on customer reviews which are an important form of opinionated content. The goal is to identify each sentence's semantic orientation (e.g. positive or negative) of a review. Traditional sentiment classification methods often involve substantial human efforts, e.g. lexicon construction, feature engineering. In recent years, deep learning has emerged as an effective means for solving sentiment classification problems. A neural network intrinsically learns a useful representation automatically without human efforts. However, the success of deep learning highly relies on the availability of large-scale training data. In this paper, we propose a novel deep learning framework for review sentiment classification which employs prevalently available ratings as weak supervision signals. The framework consists of two steps: (1) learn a high level representation (embedding space) which captures the general sentiment distribution of sentences through rating information; (2) add a classification layer on top of the embedding layer and use labeled sentences for supervised fine-tuning. Experiments on review data obtained from Amazon show the efficacy of our method and its superiority over baseline methods.

1 Introduction

With the booming of Web 2.0 and e-commerce, more and more people start consuming online and leave comments about their purchase experiences on merchant/review Websites. These opinionated contents are valuable resources both to future customers for decision-making and to merchants for improving their products and/or service. However, as the volume of reviews grows rapidly, people have to face a severe information overload problem. To alleviate this problem, many opinion mining techniques have been proposed, e.g. opinion summarization [Hu and Liu, 2004; Ding *et al.*, 2008], comparative analysis [Liu *et al.*, 2005] and opinion polling [Zhu *et al.*, 2011]. A key component for these opinion mining techniques is a sentiment classifier for natural sentences.

Popular sentiment classification methods generally fall into two categories: (1) lexicon-based methods and (2) machine learning methods. Lexicon-based methods [Turney, 2002; Hu and Liu, 2004; Ding *et al.*, 2008] typically take the tack of first constructing a sentiment lexicon of opinion words (e.g. "good", "bad"), and then design classification rules based on appeared opinion words and prior syntactic knowledge. Despite effectiveness, this kind of methods require substantial efforts in lexicon construction and rule design. Furthermore, lexicon-based methods cannot well handle implicit opinions, i.e. objective statements such as "I bought the mattress a week ago, and a valley appeared today". As pointed out in [Feldman, 2013], this is also an important form of opinions. Factual information is usually more helpful than subjective feelings. Lexicon-based methods can only deal with implicit opinions in an ad-hoc way [Zhang and Liu, 2011].

A pioneering work [Pang *et al.*, 2002] for machine learning based sentiment classification applied standard machine learning algorithms (e.g. Support Vector Machines) to the problem. After that, most research in this direction revolved around feature engineering for better classification performance. Different kinds of features have been explored, e.g. n-grams [Dave *et al.*, 2003], Part-of-speech (POS) information and syntactic relations [Mullen and Collier, 2004], etc. Feature engineering also costs a lot of human efforts, and a feature set suitable for one domain may not generate good performance for other domains [Pang and Lee, 2008].

In recent years, deep learning has emerged as an effective means for solving sentiment classification problems [Glorot *et al.*, 2011; Kim, 2014; Tang *et al.*, 2015; Socher *et al.*, 2011; 2013]. A deep neural network intrinsically learns a high level representation of the data [Bengio *et al.*, 2013], thus avoiding laborious work such as feature engineering. A second advantage is that deep models have exponentially stronger expressive power than shallow models. However, the success of deep learning heavily relies on the availability of large-scale training data [Bengio *et al.*, 2013; Bengio, 2009]. Constructing large-scale labeled training datasets for sentence level sentiment classification is still very laborious.

Fortunately, most merchant/review Websites allow customers to summarize their opinions by an overall rating score (typically in 5-stars scale). Ratings reflect the overall senti-

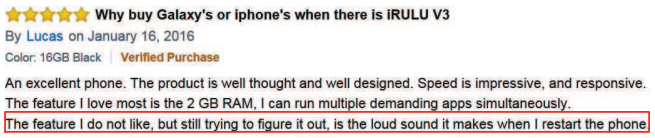


Figure 1: A negative sentence in a 5-stars review.

ment of customer reviews and have already been exploited for sentiment analysis [Maas *et al.*, 2011; Qu *et al.*, 2012]. Nevertheless, review ratings are not reliable labels for the constituent sentences, e.g. a 5-stars review can contain negative sentences and we may also see positive words occasionally in 1-star reviews. An example is shown in Figure 1. Therefore, treating binarized ratings as sentiment labels could confuse a sentiment classifier for review sentences.

In this work, we propose a novel deep learning framework for review sentence sentiment classification. The framework leverages weak supervision signals provided by review ratings to train deep neural networks. For example, with 5-stars scale we can deem ratings above/below 3-stars as positive/negative weak labels respectively. It consists of two steps. In the first step, rather than predicting sentiment labels directly, we try to learn an *embedding space* (a high level layer in the neural network) which reflects the general sentiment distribution of sentences, from a large number of weakly labeled sentences. That is, we force sentences with the same weak labels to be near each other, while sentences with different weak labels are kept away from one another. To reduce the impact of sentences with rating-inconsistent orientation (hereafter called *wrong-labeled sentences*), we propose to penalize the relative distances among sentences in the embedding space through a ranking loss. In the second step, a classification layer is added on top of the embedding layer, and we use labeled sentences to fine-tune the deep network. Regarding the network, we adopt Convolutional Neural Network (CNN) as the basis structure since it achieved good performance for sentence sentiment classification [Kim, 2014]. We further customize it by taking aspect information (e.g. screen of cell phones) as an additional context input. The framework is dubbed Weakly-supervised Deep Embedding (WDE). Although we adopt CNN in this paper, WDE also has the potential to work with other types of neural networks. To verify the effectiveness of WDE, we collect reviews from Amazon.com to form a weakly labeled set of 1.1M sentences and a manually labeled set of 11,754 sentences. Experimental results show that WDE is effective and outperforms baselines methods.

2 Related Work

Sentiment analysis is a long standing research topic. Readers can refer to [Liu, 2012] for a recent survey. Sentiment classification is one of the key tasks in sentiment analysis and can be roughly categorized as document level, sentence level and aspect level. Our work falls into the last category since we consider aspect information. In the next we review two subtopics closely related to our work.

2.1 Deep Learning for Sentiment Classification

In recent years, deep learning has received more and more attention in the sentiment analysis community. Researchers have explored different deep models for sentiment classification. Glorot *et al.* used stacked denoising auto-encoder to train review representation in an unsupervised fashion, in order to address the domain adaptation problem of sentiment classification [Glorot *et al.*, 2011]. Socher *et al.* [Socher *et al.*, 2011; 2012; 2013] proposed a series of Recursive Neural Network (RecNN) models for sentiment classification. These methods learn vector representations of variable-length sentences through compositional computation recursively. Kim investigated using CNN for sentence sentiment classification and found it outperformed RecNN [Kim, 2014]. A variant CNN with dynamic k-max pooling and multiple convolutional layers was proposed in [Kalchbrenner *et al.*, 2014]. Researchers have also investigated using sequential models such as Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) for sentiment classification [Tang *et al.*, 2015].

However, none of the above works tried to use review ratings to train deep sentiment classifiers for sentences. This is not a trivial problem since ratings are too noisy to be used directly as sentence labels (see Section 3 and experiments for discussions of this issue). To our knowledge, The WDE framework is the first attempt to make use of rating information for training deep sentence sentiment classifiers. Note that although we choose CNN as the deep model due to its competitive performance on sentiment classification [Kim, 2014], the idea of WDE could also be applied to other types of deep models. The major contribution of this work is a weakly-supervised deep learning framework, rather than specific deep models.

2.2 Exploiting Ratings in Sentiment Classification

Rating information has been exploited in sentiment classification. Qu *et al.* incorporated ratings as weak labels in a probabilistic framework for sentence level sentiment classification [Qu *et al.*, 2012]. However, their method still required careful feature design and relied on base predictors. While our method automatically learns a meaningful sentence representation for sentiment classification. Täckström and McDonald used conditional random fields to combine review level and sentence level sentiment labels for sentence sentiment analysis [Täckström and McDonald, 2011]. This method also required feature engineering. Maas *et al.* [Maas *et al.*, 2011] proposed to learn sentiment-bearing word vectors by incorporating rating information in a probabilistic model. For sentiment classification, they simply averaged the word vectors of a document as its representation. A similar work is [Tang *et al.*, 2014], which developed a variant of the C&W neural model [Collobert *et al.*, 2011] for learning sentiment-bearing word vectors from weak tweet labels derived from emoticons. The tweet representation was obtained by min, max and avg pooling on word vectors. Although this kind of methods can generate sentence representations automatically, the representations were derived by simple pooling of the learned word vectors. In comparison, our method generates a sentence representation by feeding word vectors through an ex-

pressive deep neural network. Moreover, we directly optimize sentence representation, rather than word vectors. We take the above two methods as baselines in experiments.

3 Weakly-supervised Deep Embedding

The classic deep learning methods take an "unsupervised training then supervised fine-tuning" scheme, where restricted Boltzmann machines (RBM) or auto-encoders are used to pre-train network parameters from large quantities of unlabeled data [Bengio, 2009]. This works well when the data distribution is correlated with label prediction [Bengio, 2009]. Nevertheless, in sentiment analysis the word co-occurrence information is usually not well correlated with sentiment prediction [Maas *et al.*, 2011], which motivates us to exploit large-scale rating data for training deep sentiment classifiers.

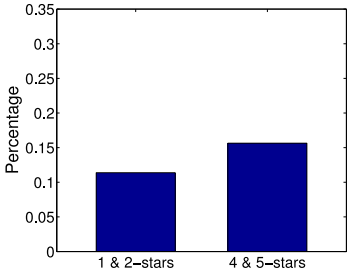


Figure 2: Percentages of wrong-labeled sentences by ratings in our labeled review dataset. The overall percentage is 13.4%.

However, ratings are noisy labels for review sentences and would mislead classifier training if directly used in supervised training. In this paper, we adopt a simple rule to assign weak labels to sentences with 5-stars rating scale:

$$\ell(s) = \begin{cases} \text{pos}, & \text{if } s \text{ is in a 4 or 5-stars review} \\ \text{neg}, & \text{if } s \text{ is in a 1 or 2-stars review} \end{cases}, \quad (1)$$

where $\ell(s)$ denotes the weak sentiment label of sentence s. Figure 2 shows the percentages of wrong-labeled sentences by $\ell(s)$, estimated in our labeled review dataset (detailed description of the dataset is in Section 4.1). We can see the noise level is moderate but not ignorable.

The general idea behind WDE is that we use large quantities of weakly labeled sentences to train a *good embedding space* so that a linear classifier would suffice to accurately make sentiment predictions. Here good embedding means in the space sentences with the same sentiment labels are close to one another, while those with different labels are kept away from each other. In the following, we first present the network architecture, and then discuss how to train it with large-scale rating data, followed by supervised fine-tuning on labeled sentences.

3.1 Network Architecture

The network architecture, depicted in Figure 3, is a variant of the CNNs described in [Collobert *et al.*, 2011; Kim, 2014]. In what follows, we use upper case bold letters such as $\mathbf{W}$ to denote matrices and lower case bold letters such as $\mathbf{x}$ to denote column vectors. The i-th element in vector $\mathbf{x}$ is denoted by $x(i)$.

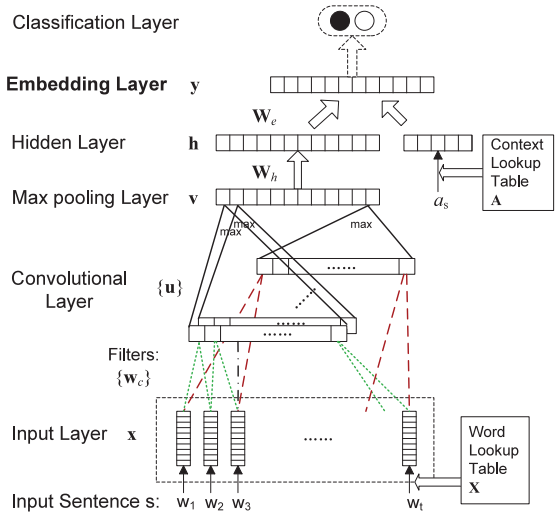


Figure 3: The CNN network architecture for sentence sentiment classification.

Input Layer. An input sentence of length t is a word sequence $s =< w_1 w_2 \dots w_t >$. Each word w in the vocabulary is described by a word vector $\mathbf{x}$. Let k be the length of $\mathbf{x}$ and n be the total number of words in the vocabulary. The trainable word lookup table $\mathbf{X}$ is then a $k \times n$ matrix with word vectors as its columns. The input layer simply maps $s =< w_1 w_2 \dots w_t >$ to its corresponding word vector representation $< \mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_t >$. The lookup table is initialized using the publicly available 300-dimensional word vectors trained on 100 billion words from Google News by word2vec [Mikolov *et al.*, 2013]. Out-of-sample words are initialized randomly.

Convolutional Layer and Max pooling Layer. The convolutional layer applies a set of *filters* on the sentence. Each filter $\mathbf{w} \in \mathbb{R}^{hk}$ is applied to a window of h words to produce a local feature value:

$$u(i) = f(\mathbf{w}^T \mathbf{x}_{i:(i+h-1)} + b), \quad (2)$$

where $\mathbf{x}_{i:(i+h-1)}$ represents the concatenated vector $[\mathbf{x}_i^T \mathbf{x}_{i+1}^T \dots \mathbf{x}_{i+h-1}^T]^T$, $u(i)$ is the computed feature value at position i, b is the bias of the current filter, and $f(\cdot)$ is a non-linear activation function such as hyperbolic tangent. Computing $u(i)$ at all possible positions in s yields a $(t-h+1)$-dimensional feature vector (i.e. a *feature map*) $\mathbf{u} = [u(1)u(2) \dots u(t-h+1)]^T$. Then the max pooling layer performs a max operation over each feature map $\mathbf{u}_j$ to find the most salient value of the filter's corresponding feature as its final value [Collobert *et al.*, 2011]

$$v(j) = \max_i \{u_j(i)\}. \quad (3)$$

This pooling scheme keeps the most important indicator of a feature and naturally leads to a fixed-length vector output $\mathbf{v}$ at the max pooling layer.

A filter with window size h is intrinsically a feature extractor which performs "feature selection" from the h-gram features of a sentence. When the input h-gram matches its $\mathbf{w}$, we will obtain a high feature value, indicating this h-gram activates the feature. This resembles the traditional feature selection in sentiment classification [Pang and Lee, 2008], but is done automatically by the network. Since traditional machine learning based methods often exploit unigrams, bigrams and trigrams [Pang and Lee, 2008], we also employ filters with different window sizes, i.e. $h = 1, 2, 3$.

Hidden Layer and Embedding Layer. The fixed-length feature vector $\mathbf{v}$ is then fed to the fully connected hidden layer and embedding layer to extract nonlinear higher level features. For the hidden layer, the computation is straightforward with weigh matrix $\mathbf{W}_h$ and bias vector $\mathbf{b}_h$:

$$\mathbf{h} = f(\mathbf{W}_h \mathbf{v} + \mathbf{b}_h). \tag{4}$$

The embedding layer gets its input from two sources: the output of the hidden layer $\mathbf{h}$, and context vector $\mathbf{a}_s$ of sentence s. A context vector is the semantic representation of an aspect that customers can comment on with respect to a sort of entities. For instance, battery life is an aspect for cell phones. The motivation for incorporating aspect information as the context of a sentence is that similar comments in different contexts could be of opposite orientations, e.g. "the screen is big" vs. "the size is big". Context vectors of all aspects constitute the context lookup table $\mathbf{A}$ (as columns). The embedding layer output is computed as

$$\mathbf{y} = f\left(\mathbf{W}_e \begin{bmatrix} \mathbf{h} \\ \mathbf{a}_s \end{bmatrix} + \mathbf{b}_e\right). \tag{5}$$

Classification Layer. This layer (and the connection below) is drawn using dotted lines since it is not until the supervised fine-tuning phase that the layer will be added to the network. We defer the description of this layer to Section 3.3.

3.2 Embedding Training with Ratings

With the weak label definition in Eq. (1), we can divide review sentences into two sets: $\mathcal{P} = \{s|\ell(s) = \text{pos}\}$ and $\mathcal{N} = \{s|\ell(s) = \text{neg}\}$. Since $\mathcal{P}$ and $\mathcal{N}$ contain wrong-labeled sentences, they cannot directly be used to train a classifier. Therefore, we propose to first train an embedding space that captures the general sentiment distribution of sentences. Intuitively, we should let sentences in $\mathcal{P}/\mathcal{N}$ stick together, while keeping $\mathcal{P}$ and $\mathcal{N}$ separated. A straightforward training scheme could be adapted from [Weston *et al.*, 2008] by stochastic gradient descent (SGD): we sample sentence pairs and reduce distances for same-label pairs and increase distances for opposite-label pairs. However, when wrong-labeled sentences are sampled, there is still a relatively high chance that we make a wrong move. To alleviate this issue, we propose to penalize relative distances for sentence triplets. The training objective is defined as a ranking loss [Collobert *et al.*, 2011]

$$\mathcal{L}_{weak} = \sum_{<s_1,s_2,s_3>} \max\left(0, \lambda - dst(s_1, s_3) + dst(s_1, s_2)\right), \tag{6}$$

where λ is the margin parameter, $dst(\cdot)$ is the Euclidean distance between sentences computed by their embedding layer representation:

$$dst(s_i, s_j) = \|\mathbf{y}_i - \mathbf{y}_j\|_2, \tag{7}$$

and $< s_1, s_2, s_3 >$ denotes a valid triplet with $\ell(s_1) = \ell(s_2) \neq \ell(s_3)$. Eq. 6 means we require the distance between same-label sentences s_1 and s_2 to be shorter than that between s_1 and a sentence s_3 with the opposite label by at least λ. A sample triplet is generated as follows. First, we randomly choose $\mathcal{P}$ or $\mathcal{N}$ as the focus. Suppose we choose $\mathcal{P}$. Then two sentences s_1 and s_2 are sampled from $\mathcal{P}$ in turn, and a sentence s_3 is sampled from $\mathcal{N}$. The case for $\mathcal{N}$ as the focus is just a mirror case.

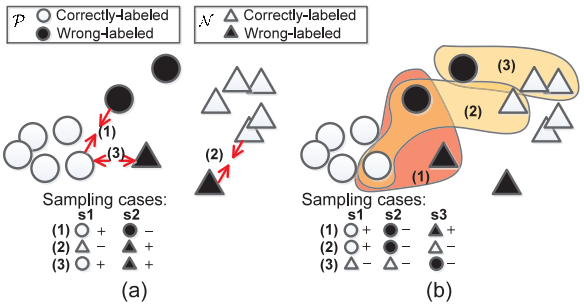


Figure 4: Comparison between (a) pair-based training and (b) triplet-based training. Please see the text for detailed explanations.

Figure 4 illustrates the advantages of triplet-based training over pair-based training via a toy example. We use circles and triangles to represent sentences in $\mathcal{P}$ and $\mathcal{N}$ respectively. Black nodes denote wrong-labeled sentences. Since the majority of sentences are with correct labels, they would gather together in the training process. Wrong-labeled sentences would go towards the wrong clusters, but with slower speeds. In both training methods, undesirable moves could happen when wrong-labeled sentences are sampled. For clarity, we just show three such cases that are representative for respective methods. The three cases in Figure 4(a) all result in undesirable moves: sentences with different orientations become closer (cases 1 and 2), while same-orientation sentences become more separated (case 3). In Figure 4(b), case 1 generates only undesirable moves: since s_3 (black triangle) is closer to s_1 (white circle) than s_2 (black circle), the algorithm will drag s_3 away from s_1 and drag s_2 toward s_1. Cases 2 and 3 lead to a mixed behavior: one move is desirable while the other one is not. Therefore, cases 2 and 3 in Figure 4(b) are not as harmful as the cases in Figure 4(a). Furthermore, in triplet-based training there will not be a move if the difference in distances exceeds the margin λ, since the derivative of $\mathcal{L}_{weak}$ becomes 0. This is useful in that we will not make things too bad. For example, in case 2 of Figure 4(b) s_2 is actually a negative sentence and should not be too close to s_1. Notice s_3 is far away from s_1. Hence, the distance difference may already exceed λ and there will be no move for this triplet. As a comparison, cases 1 and 2 in Figure 4(a)

will continually move s_1 and s_2 toward each other until their distance becomes 0, which is the worst result.

Training. Embedding training is done by taking the derivative of $\mathcal{L}_{weak}$ in (6) with respect to all the parameters under the Embedding Layer (Figure 3). We do SGD over sampled sentence triplets with AdaGrad update rule [Duchi *et al.*, 2011]. We empirically set context vector size to 50, number of filters for each window size to 200, and both hidden layer size and embedding layer size to 300. Hyperbolic tangent is employed as the activation function for all layers. The training is accelerated using GPU (28min for processing 1M triplets on a Nvidia GTX 980ti GPU).

3.3 Supervised Fine-tuning

After obtaining a good enough sentence representation by the embedding layer, we add a classification layer on the top (Figure 3) to further train the network using labeled sentences. The classification layer simply performs standard affine transformation of the embedding layer output $\mathbf{y}$ and then applies a softmax activation function [Bishop, 2006] to the result for label prediction. In this work, we focus on binary sentiment prediction (i.e. positive or negative) since we only consider sentences which comment on specific aspects of an entity. This kind of sentences hardly contain neutral sentences. Nevertheless, WDE could also be adapted to multi-class prediction problems. For binary prediction, the classification layer is equivalent to a logistic regression model. We train the network using standard SGD, since AdaGrad can easily "forget" the prior model learned in the first phase.

4 Experiments

In this section, we present the empirical evaluation of WDE on reviews collected from Amazon.com.

Table 1: Statistics of the labeled dataset.

	Positive	Negative	Total
Subjective	3750	2024	5774
Objective	1860	4120	5980
Total	5610	6144	11754

4.1 Data and Preprocessing

We collected Amazon customer reviews of 3 domains: digital cameras, cell phones and laptops. All unlabeled reviews were extracted from the Amazon data product dataset [McAuley *et al.*, 2015]. In particular, we extracted all the reviews from 12 leaf categories closely related to the above three domains (3-stars reviews were ignored). For the labeled dataset, we crawled latest reviews in 2015 for random products in the above 12 categories, in order to be disjoint with the unlabeled data. We tried to keep a balance between reviews with 4 & 5-stars and those with 1 & 2-stars. We then summarized product aspects and their keywords by traditional method [Ding *et al.*, 2008] with manual calibration. A total of 107 aspects were extracted from the obtained reviews. Next, all reviews were split into sentences and those with no aspect keywords were discarded. In case a sentence mentioned multiple aspects, we tried to split it into multiple single-aspect sentences. After preprocessing, we obtained a vocabulary of 148,183 terms, an unlabeled set of 1,143,721 sentences with rating information only (named *1.1M dataset*), and 11,754 sentences for labeling. We labeled each sentence with respect to its subjectivity and orientation. Three students were instructed to do each labeling task. The statistics of the labeled dataset is shown in Table 1. We can see the dataset is roughly balanced. For the subjectivity and orientation labeling tasks, we achieved Fleiss's kappa values of 0.81 and 0.79 respectively. The labeled dataset was randomly split into training set (50%), validation set (20%) and test set (30%) and we maintain the proportion as shown in Table 1.

4.2 Baselines and Evaluation Settings

We compare WDE with the following baseline methods:
Lexicon: this is the popular lexicon-based method proposed in [Ding *et al.*, 2008].
SVM: the support vector machine with n-gram features [Pang *et al.*, 2002] is widely employed as a baseline for sentiment classification. We use up to tri-grams since this setting is shown to yield good performance for product reviews. Liblinear [Fan *et al.*, 2008] is used to train the classifier.
NBSVM: NBSVM combines Naive Bayes and NB-enhanced SVM to predict sentiment labels [Wang and Manning, 2012]. It generates good performance on many sentiment classification datasets.
SSWE: SSWE learns sentiment-bearing word vectors by a neural network applied on weakly labeled data. We use min, max and avg pooling [Tang *et al.*, 2014] on word vectors to generate the sentence representation which is then fed to a classifier.
SentiWV: this is the sentiment word vector learning method on rating data described in Section 2 [Maas *et al.*, 2011]. We also use the aforementioned three pooling functions to generate sentence vectors. Liblinear is used to train the classifiers for SentiWV and SSWE.
CNN-rand: we train the same CNN network (Figure 3) on labeled data with random parameter initialization.
CNN-weak: we train the same CNN network on 1.1M dataset by treating weak labels defined in Eq. (1) as real labels. This baseline will answer whether rating data can be used to train sentence sentiment classifiers directly.

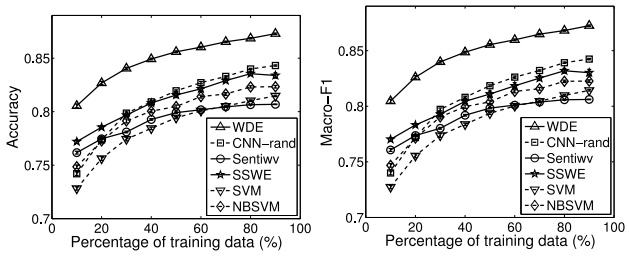
All methods (expect CNN-weak and Lexicon) are trained on the training set and evaluated on the test set. WDE, SSWE, Senti-WV have a pre-training phase on the 1.1M dataset. The validation set is used for parameter tuning of all the methods and early stopping of CNN training. We employ *Accuracy* and *Macro-F1* as the evaluation metrics.

4.3 Performance Comparison

The results are shown in Table 2. We also report performance for subjective sentences and objective sentences separately. The key observations are as follows. Lexicon performs poorly on objective sentences, since factual statements would not contain opinion words. When no opinion word is detected, we can only make random predictions. The machine learning methods all achieve acceptable performance, on both subjective and objective sentences. One exception is CNN-weak,

Table 2: Performance comparison.

Method	Accuracy			Macro-F1		
	All	Subj	Obj	All	Subj	Obj
Lexicon	.722	.827	.621	.721	.812	.613
SVM	.818	.838	.800	.818	.821	.765
NBSVM	.826	.844	.808	.825	.831	.773
SSWE	.835	.857	.815	.834	.826	.804
SentiWV	.808	.806	.809	.807	.786	.771
CNN-rand	.847	.861	.835	.847	.848	.802
CNN-weak	.771	.773	.770	.771	.755	.741
WDE	**.877**	**.886**	**.868**	**.876**	**.875**	**.843**



(a) (b)

Figure 5: Impact of labeled training data size on each method's performance.

which is trained on weakly labeled sentences. We find its validation performance fluctuates drastically during training. This indicates directly binarizing ratings as labels for supervised training is not a good idea. SSWE performs better than traditional classifiers (SVM and NBSVM) by applying a neural model on 1.1M dataset. However, it just uses word vectors to encode the useful information in the 1.1M dataset. The classifier is still a "shallow" linear model. In comparison, WDE encodes both a large number of weak supervision signals and supervision signals in a deep neural network and beats all the baselines.

4.4 Varying the Size of Training Set

Next we examine the impact of the size of labeled training data on each method's performance. CNN-weak and Lexicon are not involved since they do not depend on labeled training data. We randomly select $d\%$ training data to train the classifiers and test them on the test set, with d ranging from 10 to 90. For each d, we generate the training set 30 times and the averaged performance is reported. Figure 5 shows the results. We can see that as the number of available training instances decreases, the performance of CNN-rand, NBSVM and SVM drops faster than that of WDE, SSWE and SentiWV. This should be because the latter methods have gained prior knowledge about the sentiment distribution through pre-training, though with different capabilities. With 10% training set (nearly 600 instances), WDE can still achieve around 80% accuracy on the test set. According to t-test, WDE significantly outperforms the other methods with p-value < 0.01.

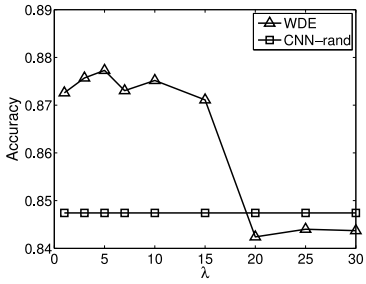


Figure 6: Impact of λ on classification performance.

4.5 Effect of λ in WDE

The margin parameter λ in Eq. (6) controls the extent to which we require weakly labeled positive instances to be separated from weakly labeled negative ones. A small value of λ may not effectively capture the sentiment distribution, while too large λ could amplify the impact of wrong-labeled sentences (Figure 4). Here we investigate λ's impact on the classification performance. Recall that the embedding layer is a 300-dimensional vector, and the output range of its neural nodes is $[-1, 1]$. It forms a hypercube where the maximal distance between any two points in the hypercube is $dia = \sqrt{1200} \approx 35$. Hence, we vary λ from 1 to 30. Figure 6 plots the performance curve. We also show the best baseline performance achieved by CNN-rand for comparison. We find the performance drops quickly when $\lambda > 15$, and when $\lambda < 15$ we can easily find a value leading to good performance. Moreover, when λ is set to a relatively high value ($> 0.5dia$), the network is more easily to be trapped in saturating regions [Bengio *et al.*, 2013] after long time training. In this paper we set $\lambda = 5$.

5 Conclusions

In this work we proposed a novel deep learning framework named Weakly-supervised Deep Embedding for review sentence sentiment classification. WDE trains deep neural networks by exploiting rating information of reviews which is prevalently available on many merchant/review Websites. The training is a 2-step procedure: first we learn an embedding space which tries to capture the sentiment distribution of sentences by penalizing relative distances among sentences according to weak labels inferred from ratings; then a softmax classifier is added on top of the embedding layer and we finetune the network by labeled data. Experiments on reviews collected from Amazon.com show that WDE is effective and outperforms baseline methods. For future work, we will investigate applying WDE on other types of deep networks and other problems involving weak labels.

Acknowledgments

This research was supported by the National Natural Science Foundation of China under Grant No. 61522206, National Basic Research Program of China (973 Program) under Grant No. 2013CB336500 and the Program for Changjiang Scholars and Innovative Research Team in University (No.

IRT13090). The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The authors would like to thank prof. Jinye Peng and prof. Jianping Fan for their helpful suggestions for this work.

References

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI*, 35(8):1798–1828, 2013.

[Bengio, 2009] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[Bishop, 2006] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537, 2011.

[Dave *et al.*, 2003] Kushal Dave, Steve Lawrence, and David M Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *WWW*, pages 519–528, 2003.

[Ding *et al.*, 2008] Xiaowen Ding, Bing Liu, and Philip S Yu. A holistic lexicon-based approach to opinion mining. In *WSDM*, pages 231–240, 2008.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[Feldman, 2013] Ronen Feldman. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89, 2013.

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

[Hu and Liu, 2004] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *SIGKDD*, pages 168–177, 2004.

[Kalchbrenner *et al.*, 2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *ACL*, 2014.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[Liu *et al.*, 2005] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *WWW*, pages 342–351, 2005.

[Liu, 2012] Bing Liu. *Sentiment analysis and opinion mining*. Morgan & Claypool Publishers, 2012.

[Maas *et al.*, 2011] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150, 2011.

[McAuley *et al.*, 2015] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *SIGKDD*, pages 785–794, 2015.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Mullen and Collier, 2004] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, volume 4, pages 412–418, 2004.

[Pang and Lee, 2008] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[Pang *et al.*, 2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86, 2002.

[Qu *et al.*, 2012] Lizhen Qu, Rainer Gemulla, and Gerhard Weikum. A weakly supervised model for sentence-level semantic orientation analysis with multiple experts. In *EMNLP-CoNLL*, pages 149–159, 2012.

[Socher *et al.*, 2011] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, pages 151–161, 2011.

[Socher *et al.*, 2012] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP-CoNLL*, pages 1201–1211, 2012.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, volume 1631, page 1642, 2013.

[Täckström and McDonald, 2011] Oscar Täckström and Ryan McDonald. Semi-supervised latent variable models for sentence-level sentiment analysis. In *ACL*, pages 569–574, 2011.

[Tang *et al.*, 2014] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, volume 1, pages 1555–1565, 2014.

[Tang *et al.*, 2015] Duyu Tang, Bing Qin, and Ting Liu. Deep learning for sentiment analysis: successful approaches and future challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(6):292–303, 2015.

[Turney, 2002] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *ACL*, pages 417–424, 2002.

[Wang and Manning, 2012] Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, pages 90–94, 2012.

[Weston *et al.*, 2008] Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *ICML*, pages 1168–1175, 2008.

[Zhang and Liu, 2011] Lei Zhang and Bing Liu. Identifying noun product features that imply opinions. In *ACL*, pages 575–580, 2011.

[Zhu *et al.*, 2011] Jingbo Zhu, Huizhen Wang, Muhua Zhu, Benjamin K Tsou, and Matthew Ma. Aspect-based opinion polling from customer reviews. *IEEE Transactions on Affective Computing*, 2(1):37–49, 2011.