Data Sci. Eng. (2017) 2:178–186
DOI 10.1007/s41019-017-0040-6

Guiding the Training of Distributed Text Representation with Supervised Weighting Scheme for Sentiment Analysis

Zhe Zhao[1] · Tao Liu[1] · Shen Li[2] · Bofang Li[1] · Xiaoyong Du[1]

Received: 4 January 2017 / Revised: 13 March 2017 / Accepted: 14 March 2017 / Published online: 27 April 2017

Abstract With the rapid growth of social media, sentiment analysis has received growing attention from both academic and industrial fields. One line of researches for sentiment analysis is to feed bag-of-words (BOW) text representation into classifiers. Usually, raw BOW requires weighting schemes to obtain better performance, where important words are given more weights while unimportant ones are given less weights. Another line of researches focuses on neural models, where distributed text representations are learned from raw texts automatically. In this paper, we take advantages of techniques in both lines of researches. We use words' weights to guide neural models to focus on important words. Various supervised weighting schemes are explored in this work. We discover that better text features are learned for sentiment analysis when suitable weighting schemes are applied upon neural models.

Keywords Distributed representation · Sentiment analysis · Weighting scheme · word2vec · GloVe

1 Introduction

Sentiment analysis aims at extracting users' subjective information from texts. It is a useful technique which turns user-generated texts into knowledge for decision support. A successful paradigm for sentiment analysis is to feed bag-of-words (BOW) text representation into linear classifiers. In BOW, each word corresponds to one dimension of representation. Usually, weighting schemes are used to give those important words more weights and reduce weights of unimportant ones such as stop words [17]. Though this paradigm is quite simple, strong baselines are achieved on a range of sentiment analysis datasets [24].

However, traditional BOW representation suffers from sparsity problem. They take each word as an atomic unit, which totally ignores the internal semantics of words. Recently, many researchers have turned their attention to learning distributed text representations by neural networks (NNs) [6], e.g., convolutional NNs [9] and recursive NNs [23]. Neural models are known for their automatic feature learning ability. They are able to extract features from raw data directly with no requirements of prior knowledge. However, a natural question is raised here: when prior knowledge is available, is it possible for us to utilize the knowledge to help the neural networks to achieve better results? This is the motivation of our paper.

In this work, we propose a novel approach which takes advantages of both traditional weighting schemes and neural models. Weighting schemes tell us which words are important and which are not. Taking the examples of movie reviews: Compared with neutral words like 'of' and 'watch,' words like 'wonderful' and 'worst' are more important in terms of reflecting reviewers' sentiment tendencies. Many works have proven the effectiveness of weighting schemes on raw BOW representation [10, 17, 20, 24]. Inspired by their successes on BOW, we discover that weighting technique is also useful in guiding the training of neural networks. To be specific, we

✉ Tao Liu
tliu@ruc.edu.cn

Zhe Zhao
helloworld@ruc.edu.cn

[1] School of Information, Renmin University of China, Information building, Beijing, China

[2] Institute of Chinese Information Processing, Beijing Normal University, Beijing, China

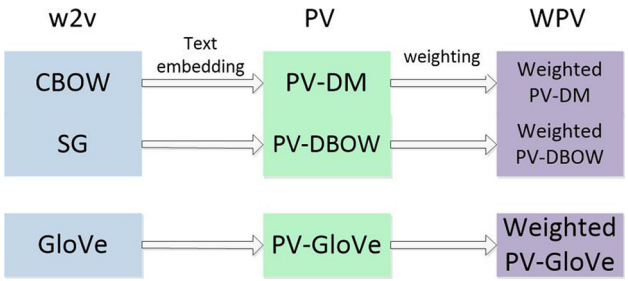

Fig. 1 *Top* (*bottom*): relationships among word2vec (GloVe), PV (PV-GloVe), and WPV (WPV-GloVe)

introduce weighting information into Paragraph Vector (PV) [11]. The details of PV are discussed in Sect. 3.1. Figure 1 (top) illustrates the relationships among word2vec, PV, and our proposed model, weighted PV (WPV). Word2vec (w2v) is a toolkit for generating word embedding. It contains two models, continuous bag-of-words (CBOW) and skip gram (SG). PV introduces text embedding into word2vec for training distributed text representation. Our models further use words' weights to guide the training of PV. Text embeddings are trained to pay more attention to those important words while ignore unimportant ones.

Inspired by the line of researches in Fig. 1 (top), we also introduce text embedding and weighting information into GloVe [22]. Same with word2vec, GloVe is a popular word embedding model. Its training objective is to reconstruct word–word co-occurrence matrix. We firstly propose PV-GloVe, where text and word embeddings are trained on the basis of co-occurrence matrix of text–word type. Then, we weight text–word co-occurrence matrix to guide the text embeddings to focus on those important words (WPV-GloVe). Relationships of these models are illustrated in Fig. 1 (bottom). At last, we review some supervised weighting schemes. They take class preference into consideration and are proven to be useful on BOW representation. We show that significant and consistent improvements are witnessed when weighting schemes are applied to PV and PV-GloVe.

Section 2 reviews the related work of sentiment analysis. Section 3 discusses how to integrate weighting schemes into Paragraph Vector. In Sect. 4, PV-GloVe and weighted PV-GloVe are proposed, where text embedding and weighting information are introduced into GloVe. The weighting schemes used in this paper are discussed in Sect. 5. Experimental results are given in Sect. 6, followed by the conclusion in Sect. 7.[1]

[1] This paper extends the work done by [25]. The extension includes the introduction of PV-GloVe and weighted PV-GloVe. Besides that, more comprehensive discussions about various weighting schemes are provided.

2 Related Work

2.1 Bag-of-Words Representation

Text representation is of vital importance for sentiment analysis. One of the most popular text representation methods is bag-of-words (BOW) (also known as Vector Space model). Early work that uses BOW on sentiment analysis is done by [21]. They find that binary weighted BOW representation with SVM classifier gives promising results. Following their work, various weighting schemes are proposed to weight raw BOW text representation. Word weighting has been intensively studied in the Information Retrieval (IR) literature. However, weighing schemes in IR do not consider the label information which is available at sentiment analysis datasets. A popular supervised weighting scheme in sentiment analysis is naive Bayes (NB) weighting [17]. It calculates the document frequency (DF) of words in positive and negative classes, respectively, and then uses the ratios of them to weight words. [24] apply NB weighting to bag-of-ngrams features and achieve very strong baselines on a range of sentiment analysis and text classification datasets. [10] further add credibility information upon supervised weighting scheme to determine the importance of words for sentiment analysis. [20] and [3] give comprehensive discussions about different weighting schemes over BOW representation. The former work introduces supervised information into classic weighting schemes in IR. The latter one summarizes various supervised weighting schemes.

2.2 Neural Models

Recently, neural networks (NNs) have become increasingly popular in the natural language processing (NLP) community. They learn distributed word and text representations and achieve state-of-the-art results on a range of NLP tasks. Compared with BOW representation, NNs are able to capture word order and even complex structures of textual data. Besides that, the internal semantics of words are captured due to the distributed representation. CNNs are used for sentiment analysis in [8, 9]. CNNs use convolutional layers to extract ngram features and use max-pooling layers to select the most distinct one. Recursive NNs (RecNNs), proposed by [23], construct neural networks on the basis of parse trees and are able to extract fine-grained information of sentences. Another family of NNs is Recurrent NNs (RNNs). Words in the text are fed into RNNs one by one, updating the states of hidden layers. In theory, the hidden layers store all previous information, and the hidden layer of the last word can be used as the representation of the whole text.

Most recently, combinations of neural networks are proposed to capture complex structures of texts. They can be divided into two components. The first component is to learn sentence features from word features, and the second component is to learn document features from sentence features [4, 15]. As a result, these models can not only capture word order and syntactic information in a sentence, but also take relationships among sentences into consideration. The drawbacks of these deep neural models are also obvious. These models are expensive in computational resources. Besides that, they are not as robust as traditional BOW approaches. The performance of these models closely relies on careful hyper-parameter tuning and some sub-tasks such as pre-trained word embedding and parsing.

2.3 Neural Bag-of-Words Models

Though deep neural models are powerful in theory, only limited improvements are achieved on sentiment analysis compared with traditional BOW with weighting schemes. It seems that information like sentence and document structures is not very crucial for sentiment analysis. Another line of neural models is neural bag-of-words models. Instead of constructing complex compositions upon word embeddings, these models basically ignore order and syntax information. Representative neural bag-of-words models include Deep Averaging Network (DAN) [7] and Paragraph Vector (PV) [11]. DAN firstly takes the average of word embeddings as the inputs and then constructs multiple neural layers upon them. PV embeds text by making it useful to predict the words it includes. These models enjoy the advantages of being simple and robust compared with complex deep neural models, and can still achieve competitive results on many NLP tasks. Following these models, we propose another neural bag-of-words model, PV-GloVe, which introduces text embedding into GloVe model [22]. Several works reveal the intrinsic connections between GloVe and word2vec [13, 22]. Though word2vec and Glove are different apparently, they essentially use word–word co-occurrence to train word embeddings. In fact, GloVe is more flexible than word2vec since GloVe explicitly utilizes word–word co-occurrence information (word2vec uses co-occurrence information implicitly).

One problem of neural bag-of-words models is that they treat each word equally. Intuitively, some words are more important for sentiment analysis task, such as ‘amazing’ and ‘best.’ To capture better features, we use weighting schemes to guide the training of these models. To this end, the new models are able to pay more attention to words that reflect polarities of texts. When suitable weighting schemes are used, significant improvements over traditional neural bag-of-words models are witnessed.

3 Weighted Paragraph Vector

3.1 Paragraph Vector Revisit

In this section, we introduce weighting schemes into a neural bag-of-words model, Paragraph Vector (PV). In PV, text and word embeddings are initialized randomly. During the training, the text embedding is trained to be useful for predicting the words it contains (As shown in Fig. 2 (left)). The objective of PV is as follows:

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|t_i|}\log P(w_{ij}|w_{ij}^{\text{context}}) \tag{1}$$

where $\mathrm{t}_i = \{\mathrm{w}_{i1}, \mathrm{w}_{i2}, \ldots, \mathrm{w}_{i|t_i|}\}$ denotes the ith text and $\mathrm{T} = \{\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_{|T|}\}$ denotes the whole dataset. $\mathrm{w}^{\text{context}}$ represents the context of the target word w. In PV, the context includes both the words in the local window and the text that the target word belongs to. In fact, the definition of context is the major difference between PV and word2vec. In word2vec, the context only includes the surrounding words. PV consists of two models: PV-DBOW and PV-DM. In addition, PV provides two ways of defining conditional probability $P(|)$, which are negative sampling (NS) and hierarchical softmax (HS). As a result, PV has four variants. For brevity, we hide the details of conditional probability. One can see [19] for more information.

In the following two subsections, we introduce weighting schemes into PV-DBOW and PV-DM, respectively.

3.2 Weighted PV-DBOW

The original objective function of PV-DBOW (implemented in [18]) is as follows:

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|t_i|}\left(\log P(w_{ij}|t_i) + \sum_{-c\le k\le c, k\neq 0}\log P(w_{ij}|w_{i(j+k)})\right) \tag{2}$$

where c denotes the window size. The first part of the objective uses text that contains the target word as context. The second part is just the objective of ordinary word embedding model, where words in the local window are used as the context. Traditional Paragraph Vector model treats each word in the text equally. In this sense, it can be viewed as the neural counterpart of BOW where each feature is represented by the raw count of word in the text.

Obviously, some words are more important. The main idea of our model is to make the text embedding pay more attention to those important words, instead of neutral words which have little value for determining polarities of texts. We give each word a weight (a real value), which represents the importance of the word for sentiment analysis.

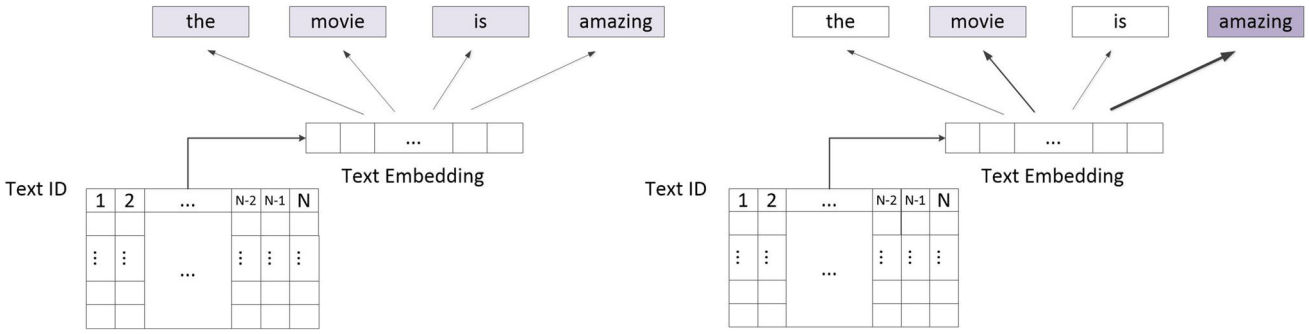

Fig. 2 Illustration of PV-DBOW (*left*) and weighted PV-DBOW (*right*)

How to assign the weights to words is discussed in Sect. 5. The objective of weighted PV-DBOW is as follows:

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|t_i|}\Big(\text{Weight}(w_{ij})\log P(w_{ij}|t_i) + \sum_{-c\le k\le c, k\neq 0}\log P(w_{ij}|w_{i(j+k)})\Big) \tag{3}$$

where Weight(w) is the weight assigned to word w. By optimizing the above weighted objective, the conditional probabilities of words with large weights will have more proportions to the entire objective. To this end, the trained text embedding is able to predict important words in larger probabilities while ignore those unimportant ones. Figure 2 (right) illustrates the framework of weighted PV-DBOW.

3.3 Weighted PV-DM

PV-DM uses the average of text embedding and word embeddings in the local window to predict the target word. The original objective of PV-DM (implemented in [18]) is as follows:

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|t_i|}\log P(w_{ij}|w_{ij}^{\text{context}}) \quad \text{where} \quad w_{ij}^{\text{context}} = e(t_i) + \sum_{-c\le k\le c, k\neq 0} e(w_{i(j+k)}) \tag{4}$$

where $e(\cdot)$ denotes the embedding of the word–text $\cdot$. Like the way we introduce weighting information into PV-DBOW, we can optimize the following weighted objective function:

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|t_i|}\text{Weight}(w_{ij})\log P(w_{ij}|w_{ij}^{\text{context}}) \tag{5}$$

An alternative of introducing weighting information into PV-DM is to change the representation of w^{context}. Since text embeddings should pay more attention to important words, we give text embeddings more weights in constructing contexts when target words are important. In this way, the text embeddings are more affected by those important words while less affected by those unimportant words. The objective function is as follows:

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|t_i|}\log P(w_{ij}|w_{ij}^{\text{context}}) \quad \text{where} \quad w_{ij}^{\text{context}} = \text{Weight}(w_{ij})e(t_i) + \sum_{-c\le k\le c, k\neq 0} e(w_{i(j+k)}) \tag{6}$$

We find the latter objective performs slightly better in practice. In the rest of the paper, we use weighted PV-DM to denote the latter model.

4 (Weighted) PV-GloVe

4.1 GloVe Revisit

GloVe is a popular word embedding model and achieves competitive results on a range of linguistic tasks. Compared with word2vec, GloVe requires constructing word–word co-occurrence matrix and directly uses it for training word embeddings. During the training, the word (context) embeddings is trained to be able to reconstruct nonzero values in the co-occurrence matrix. The objective is as follows:

$$\sum_{i,j=1}^{|V|} f(X_{ij})(e(w_i)^T e(\widetilde{w}_j) + b_i + \widetilde{b}_j - \log X_{ij})^2 \tag{7}$$

where $|V|$ denotes the vocabulary size, X_{ij} denotes the number of times the word i and word j co-occur. Function f is used to smooth the raw count X_{ij}. $e()$ and $e(\widetilde{})$, respectively, denote the word and context embeddings. b denotes the bias. One can see [22] for more details of GloVe.

GloVe is a more general model compared with word2vec. In fact, the objective of word2vec implicitly factorizes a weighted word–word co-occurrence matrix. However, GloVe provides the flexibility of applying different

weighting schemes to the co-occurrence matrix. To some extent, word2vec can be viewed as a special case of GloVe. In the following two sections, we introduce text embedding and weighting information into GloVe for generating distributed text representations.

4.2 PV-GloVe

In this section, we introduce text embedding into GloVe model. Firstly, we give each text a unique embedding and initialize text embeddings randomly. During the training, the model is not only required to reconstruct word–word co-occurrence matrix, but also required to reconstruct text–word co-occurrence matrix. To this end, the objective of PV-GloVe consists of two components. The first component is the same as the original objective of GloVe. The objective of reconstructing text–word co-occurrence matrix is as follows:

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|V|} f(Y_{ij})(e(t_i)^T e(w_j) + b_i^t + b_j - \log Y_{ij})^2 \tag{8}$$

where Y_{ij} denotes the values in text–word co-occurrence matrix. b^t is the bias of the text. After training, text and word embeddings are able to reconstruct text–word co-occurrence matrix, thus contain the information of the words in the texts.

4.3 Weighted Co-occurrence Matrix

To guide the model to pay more attention to those important words, we introduce weighting schemes into PV-GloVe. As we have discussed above, GloVe has the advantage of being flexible for weights assignment. To this end, we directly put weights on text–word co-occurrence matrix and the following objective is optimized:

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|V|} f(Y_{ij}\text{Weight}(w_{ij}))(e(t_i)^T e(w_j) + b_i^t + b_j - \log(Y_{ij}\text{Weight}(w_{ij})))^2 \tag{9}$$

where $Y_{ij}\text{Weight}(w_{ij})$ substitutes the raw count Y_{ij}. After weighting, the text embeddings are largely affected by those important words. Figure 3 illustrates the change from text–word pair's original counts to weighted counts.

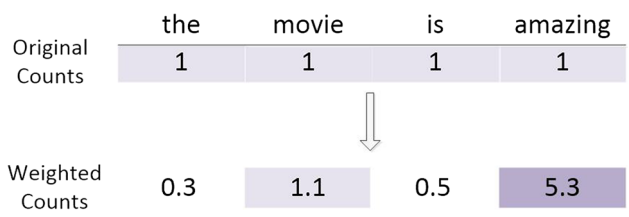

Fig. 3 Illustration of (weighted) text–word pairs counts

5 Weighting Schemes

For several decades, intensive researches have been done on designing weighting schemes over raw BOW representation. In sentiment analysis, supervised weighting schemes have shown to be effective [3, 10, 17, 24]. They take advantage of label information in the dataset to determine the importance of a word. In this work, we explore five commonly used supervised weighting schemes. They share similar ideas, but are different in details. All of them require counting some basic statistics of the dataset and use them to calculate the word's weight. The statistics used in this paper and their notations are listed in Table 1. Intuitively, a word that has uneven distribution over classes should obtain higher weights, e.g., words like 'great' are more likely to appear in positive texts than negative texts. However, to the best of our knowledge, there is still rare work applying weighting schemes to neural models. Since these supervised weighting schemes are useful in sparse BOW representation, we use them on neural models and expect that better text representations are learned for sentiment analysis.

5.1 Naive Bayes (NB)

Naive Bayes (NB) weighting is one of the most popular supervised weighting schemes in sentiment analysis [24]. It is also called delta IDF in [17] and likelihood ratio in [20]. The definition of NB weighting is as follows:

$$\log \frac{\#\text{POS}(w) \times \#N_{\text{neg}}}{\#\text{NEG}(w) \times \#N_{\text{pos}}} \tag{10}$$

where log function is used to smooth the ratio between #POS and #NEG. In this work, we use square root to replace log. It is an engineer choice. We find that square root is more effective on the models used in this work. To this end, the weight of word w is defined as follows:

$$\max\left(\text{sqrt}\frac{\#\text{POS}(w) \times \#N_{\text{neg}}}{\#\text{NEG}(w) \times \#N_{\text{pos}}}, \text{sqrt}\frac{\#\text{NEG}(w) \times \#N_{\text{pos}}}{\#\text{POS}(w) \times \#N_{\text{neg}}}\right) \tag{11}$$

Table 1 Statistics used in this paper and their notations. Only binary classification is considered

Notation	Meaning
# N_{pos}	The number of positive texts in the dataset
# N_{neg}	The number of negative texts in the dataset
#N	The number of texts in the dataset
#POS(w)	The number of positive texts that contain word w
#NEG(w)	The number of negative texts that contain word w
#$N(w)$	The number of texts that contain word w

5.2 Pointwise Mutual Information (PMI)

In the natural language processing community, pointwise mutual information (PMI) is usually used to measure word associations [1]. [12] show that word2vec implicitly factorizes PMI matrix of word–word type. The positive PMI matrix is shown to be effective on a range of linguistic tasks [13]. Here, we use PMI criterion to measure the associations between words and labels [20]. Different from classic definition of PMI, *square root* replaces log. The weighting is defined as follows:

$$\max\left(\text{sqrt}\frac{\#\text{POS}(w)\times\#N}{\#N(w)\times\#N_{\text{pos}}},\text{sqrt}\frac{\#\text{NEG}(w)\times\#N}{\#N(w)\times\#N_{\text{neg}}}\right) \tag{12}$$

5.3 Average Likelihood of Making Correct Classification (AL)

Average likelihood of making correct classification (AL) is used as supervised weighting scheme in [10]. Suppose we are asked to guess whether a word 'watch' occurs in positive text or not. We probably have a 50–50 chance of making correct answer since 'watch' is distributed relatively equally on positive and negative classes. However, for words like 'best,' we have larger likelihood of guessing correctly by answering yes all time. The detailed derivation of AL is provided in [10]. The final weighting is as follows:

$$\frac{\#\text{POS}(w)^2+\#\text{NEG}(w)^2}{\#N(w)^2} \tag{13}$$

when a word has even distribution over classes, the weight is around 1/2. The weight equals to 1 if a word always occurs in one class.

5.4 Odds Ratio (OR)

In binary classification, odds ratio (OR) only makes a slight adjustment upon NB weighting, where $\#N_{\text{pos}}$ and $\#N_{\text{neg}}$ are replaced by $\#N_{\text{pos}}-\#\text{POS}(w)$ and $\#N_{\text{neg}}-\#\text{NEG}(w)$. For low-frequency words, the difference between NB and OR weighting is negligible. Same with the situation in NB weighting, square root is used to smooth the ratios:

$$\max\left(\text{sqrt}\frac{\#\text{POS}(w)\times(\#N_{\text{neg}}-\#\text{NEG}(w))}{\#\text{NEG}(w)\times(\#N_{\text{pos}}-\#\text{POS}(w))},\ \text{sqrt}\frac{\#\text{NEG}(w)\times(\#N_{\text{pos}}-\#\text{POS}(w))}{\#\text{POS}(w)\times(\#N_{\text{neg}}-\#\text{NEG}(w))}\right) \tag{14}$$

5.5 Weighted Naive Bayes (WNB)

Weighted naive Bayes (WNB) further introduces credibility information into NB weighting [20]. If a word occurs twice in positive texts and once in negative texts, we cannot reject the hypothesis that the word has even distribution over classes with reasonable significance levels. If a word occurs 200 and 100 times in positive and negative texts, respectively, we can safely conclude that the word has uneven distribution over classes, even though the ratio is the same with the previous case. WNB adds a term before NB weighting to take the words' frequencies into consideration:

$$\frac{\#N(w)}{\#N}\max\left(\text{sqrt}\frac{\#\text{POS}(w)\times\#N_{\text{neg}}}{\#\text{NEG}(w)\times\#N_{\text{pos}}},\text{sqrt}\frac{\#\text{NEG}(w)\times\#N_{\text{pos}}}{\#\text{POS}(w)\times\#N_{\text{neg}}}\right) \tag{15}$$

6 Experiments

6.1 Experimental Setup and Dataset

Our work is based on two baseline models: Paragraph Vector and GloVe. In this paper, the hyper-parameter settings of two baselines follow the [18][2] (an implementation of PV) and [22][3] (an implementation of GloVe) respectively. The number of training epochs is the only hyper-parameter that is determined by validation data. The trained text embeddings are then fed into logistic regression classifier [5] for classification.

IMDB dataset [16] is used for evaluating different models. IMDB is one of the most popular benchmarks in sentiment analysis [16]. It comprises 50000 labeled movie reviews. Half (25000) of them are positive and half are negative. The training set contains 12500 positive and 12500 negative reviews. The rest belong to the testing set. Table 2 lists the statistics of some commonly used words. According to our understanding to these words, we group them into three categories: neutral, positive, and negative. The results are intuitive. The ratios in positive/negative groups are larger/smaller than one. In the neutral group, the ratios are close to one.

6.2 Effectiveness of Weighting Schemes on PV and PV-GloVe

In this section, we show the effectiveness of various weighting schemes on PV and PV-GloVe. In the original PV implementation, negative sampling (NS) and hierarchy

[2] https://github.com/mesnilgr/iclr15.

[3] https://github.com/stanfordnlp/GloVe.

Table 2 Some commonly used words and their statistics

Type	Word	#POS	#NEG	Ratio(#POS/#NEG)
Neutral	Movies	2482	2804	0.885
	Something	1692	2307	0.733
	Watch	2650	2775	0.955
	Saw	1451	1294	1.121
	Other	3454	3216	1.074
Positive	Amazing	866	240	3.608
	Awesome	307	111	2.766
	Wonderful	1159	273	4.245
	Marvelous	123	27	4.556
	Remarkable	218	62	3.516
Negative	Bad	1465	4344	0.337
	Terrible	214	1113	0.192
	Ugly	79	223	0.354
	Horrible	152	842	0.181
	Suck	23	134	0.172

softmax (HS) are used together. To this end, PV has 5 baseline results (4 variants discussed in Sect. 3.1 plus the original implementation). Table 3 shows the results of (weighted) PV and (weighted) PV-GloVe. To better illustrate the performance of different methods, we use bar graph (Fig. 4) to present the results. From the 'baseline' column in Table 3, we can observe that PV-GloVe performs comparably with PV. It is a reasonable result since PV-GloVe and PV essentially utilize text, word co-occurrence to train text embeddings. They are both BOW models, where no word order and other complex information are considered. Among variants in PV, the PV-DM with negative sampling performs the best.

Comparing the baselines with the results in other columns, we find that significant and consistent improvements are witnessed when supervised weighting schemes are introduced into the models. Though neural models are known for their automatic feature learning ability, weighting information is still useful for these models to train better text embeddings. Different weighting schemes perform comparably. In general, NB weighting is robust. While it is not the best weighting scheme for every case, it always achieves above-average accuracies.

6.3 Comparisons of State-of-the-Art Methods

In this section, we compare our models with state of the arts. For clarity, models are classified into 3 groups according to the way of exploiting the information in texts. Both sparse BOW and neural BOW models are put into the BOW group. They only consider whether a word occurs in a text or not. BOW models seem to oversimplify the text modeling since the semantics of texts often lie in word

Table 3 Effectiveness of weighting schemes on PV and PV-GloVe

Baselines	Baseline	NB	PMI	AL	OR	WNB
PV-DBOW + NS + HS	88.7	89.6	89.3	88.9	**89.7**	89.3
PV-DBOW + NS	89.0	89.9	89.6	89.6	**90.1**	90.0
PV-DBOW + HS	87.8	**88.6**	88.3	88.0	88.5	88.0
PV-DM + NS	89.5	**90.1**	**90.1**	89.8	90.0	89.5
PV-DM + HS	87.4	**88.5**	88.2	87.8	88.3	88.2
PV-GloVe	89.3	**90.0**	89.5	89.8	89.7	88.5

The top weighting schemes are in bold

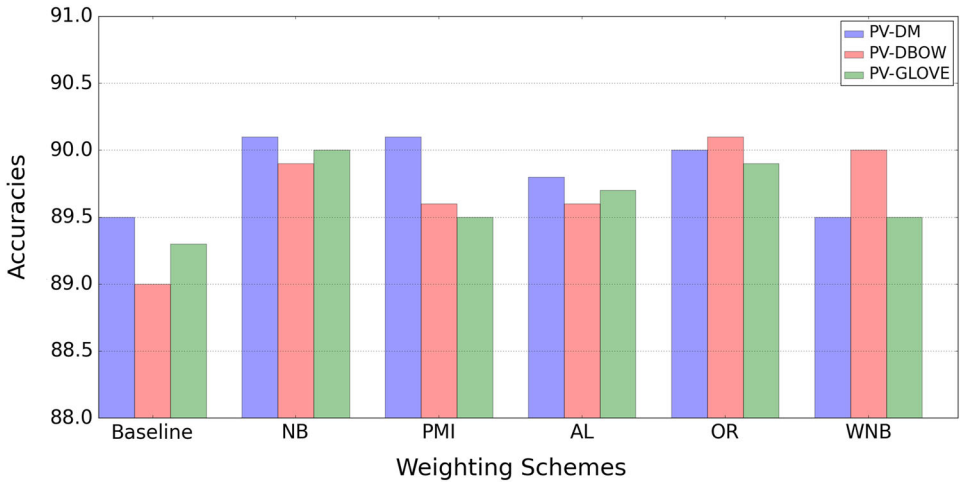

Fig. 4 Effectiveness of weighting schemes on PV and PV-GloVe. Negative sampling is selected as softmax

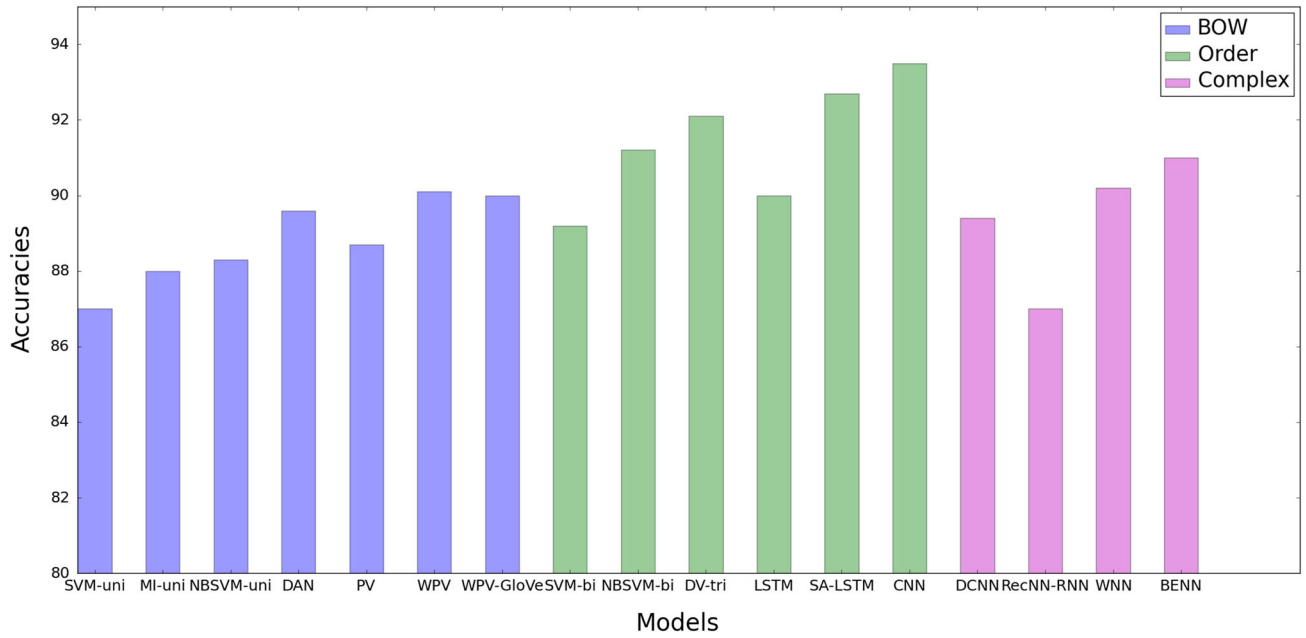

Fig. 5 Comparisons of our models with state of the arts. SVM-uni, NBSVM-uni, SVM-bi, and NBSVM-bi: [24]. MI-uni: [20]. DAN: [7]. PV: [18]. DV-tri: [14]. LSTM and SA-LSTM: [2], CNN: [8]. DCNN: [4]. RecNN-RNN, WNN, and BENN: [15]

order, syntax, and other complex factors. Many models such as NBSVM-bi [24], CNNs [8], and RNNs [2] take word order into consideration. Upon word order, Four models in 'complex' group use the combination of neural networks to capture more complex information from textual data. Compared with neural models such as CNNs and RNNs, BOW models have the advantages of being efficient and robust, but are inferior to those complex models in accuracies [7]. NBSVM introduces supervised weighting schemes into bag-of-ngrams representation and provides strong baselines. The downside of NBSVM is that it requires too much memory due to the large number of ngrams. Similar problem arises in DV-ngram [14], where ngram embedding is introduced into PV (Fig. 5).

We can observe that our models perform the best in BOW group. The word order seems to be important on IMDB dataset. LSTMs and CNNs exceed our methods by around 3 percent. However, it should be pointed out that many tricks are required to achieve the best results reported in [2, 8]. The original implementation of LSTM only obtains 90.0 accuracy. From the results in the third group, we find that accuracies do not benefit from more complex information of textual data.

7 Conclusion and Future Work

In this paper, we introduce weighting schemes into Paragraph Vector (PV). The weighting information guides PV to focus on important words while ignores unimportant ones. Inspired by the line of models 'word2vec–PV-weighted PV,' we further introduce text embedding and weighting schemes into GloVe, which are PV-GloVe and weighted PV-GloVe, respectively.

Various supervised weighting schemes are explored in this work. Significant and consistent improvements are witnessed with the introduction of weighting information. Weighting schemes are effective on PV and PV-GloVe, just as weighting schemes on sparse BOW representation. Besides that, we discover that PV-GloVe is useful in generating text representations. PV-GloVe and weighted PV-GloVe achieve comparable results with PV and weighted PV, respectively. Finally, we compare our methods with newly proposed neural models. In terms of accuracy, our methods do not perform as well as them. Nevertheless, the gap is not big and our models are simple, efficient, and do not require additional resources.

A direction that remains to be explored is using attention mechanism to learn the weight of word automatically. Existing weighting schemes are proposed on the basis of people's prior knowledge on textual data. It is a more elegant way to integrate the word weighting process into neural models. In our future work, we will explore attention-based neural models on sentiment analysis and compare them with weighting schemes designed by hands.

References

1. Church KW, Hanks P (1990) Word association norms, mutual information, and lexicography. Comput Linguist 16(1):22–29
2. Dai AM, Le QV (2015) Semi-supervised sequence learning. In: Advances in neural information processing systems 28: annual conference on neural information processing systems 2015, December 7–12, Montreal, Quebec, Canada, pp 3079–3087
3. Deng ZH, Luo KH, Yu HL (2014) A study of supervised term weighting scheme for sentiment analysis. Expert Syst Appl 41(7):3506–3513
4. Denil M, Demiraj A, Kalchbrenner N, Blunsom P, de Freitas N (2014) Modelling, visualising and summarising documents with a single convolutional neural network. arXiv preprint arXiv:1406.3830
5. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) Liblinear: a library for large linear classification. J Mach Learn Res 9:1871–1874
6. Goldberg Y (2016) A primer on neural network models for natural language processing. J Artif Intell Res 57:345–420
7. Iyyer M, Manjunatha V, Boyd-Graber JL, Daumé III H (2015) Deep unordered composition rivals syntactic methods for text classification. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing of the asian federation of natural language processing, ACL 2015, July 26–31, Beijing, China, volume 1: long papers, pp 1681–1691
8. Johnson R, Zhang T (2015) Semi-supervised convolutional neural networks for text categorization via region embedding. In: Advances in neural information processing systems 28: annual conference on neural information processing systems 2015, December 7–12, Montreal, Quebec, Canada, pp 919–927
9. Kim Y (2014) Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882
10. Kim Y, Zhang O (2014) Credibility adjusted term frequency: A supervised term weighting scheme for sentiment analysis and text classification. arXiv preprint arXiv:1405.3518
11. Le QV, Mikolov T (2014) Distributed representations of sentences and documents. ICML 14:1188–1196
12. Levy O, Goldberg Y (2014) Neural word embedding as implicit matrix factorization. In: Advances in neural information processing systems 27: annual conference on neural information processing systems 2014, December 8–13, Montreal, Quebec, Canada, pp 2177–2185
13. Levy O, Goldberg Y, Dagan I (2015) Improving distributional similarity with lessons learned from word embeddings. Trans Assoc Comput Linguist 3:211–225
14. Li B, Liu T, Du X, Zhang D, Zhao Z (2015) Learning document embeddings by predicting n-grams for sentiment classification of long movie reviews. arXiv preprint arXiv:1512.08183
15. Li J (2014) Feature weight tuning for recursive neural networks. arXiv preprint arXiv:1412.3714
16. Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, Vol 1. pp 142–150. Association for Computational Linguistics
17. Martineau J, Finin T (2009) Delta tfidf: an improved feature space for sentiment analysis. Icwsm 9:106
18. Mesnil G, Mikolov T, Ranzato M, Bengio Y (2014) Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. arXiv preprint arXiv:1412.5335
19. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems 26: 27th annual conference on neural information processing systems 2013, December 5–8, Lake Tahoe, NV, USA, pp 3111–3119
20. Paltoglou G, Thelwall M (2010) A study of information retrieval weighting schemes for sentiment analysis. In: Proceedings of the 48th annual meeting of the association for computational linguistics, pp 1386–1395. Association for Computational Linguistics
21. Pang B, Lee L, Vaithyanathan S (2002) Thumbs up?: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing, Vol 10, pp 79–86. Association for Computational Linguistics
22. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. EMNLP 14:1532–1543
23. Socher R, Huval B, Manning CD, Ng AY (2012) Semantic compositionality through recursive matrix-vector spaces. In: Proceedings of the 2012 Joint Conference on empirical methods in natural language processing and computational natural language learning, pp 1201–1211. Association for Computational Linguistics
24. Wang S, Manning CD (2012) Baselines and bigrams: simple, good sentiment and topic classification. In: Proceedings of the 50th annual meeting of the asssociation for computational linguistics: short papers, Vol 2, pp 90–94. Association for Computational Linguistics
25. Zhao Z, Liu T, Hou X, Li B, Du X (2016) Distributed text representation with weighting scheme guidance for sentiment analysis. In: Asia-Pacific web conference, Springer, pp 41–52