Incorporating Positional Information into Deep Belief Networks for Sentiment Classification

Yong Jin[1], Harry Zhang[1], and Donglei Du[2]

[1] Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada, E3B 5A3
{yjin1,hzhang}@unb.ca,
[2] Faculty of Business Administration, University of New Brunswick, Fredericton, NB, Canada, E3B 5A3
ddu@unb.ca

Abstract. Deep belief networks (DBNs) have proved powerful in many domains including natural language processing (NLP). Sentiment classification has received much attention in both engineering and academic fields. In addition to the traditional bag-of-word representation for each sentence, the word positional information is considered in the input. We propose a new word positional contribution form and a novel word-to-segment matrix representation to incorporate the positional information into DBNs for sentiment classification. Then, we evaluate the performance via the total accuracy. Consequently, our experimental results show that incorporating positional information performs better on ten short text data sets, and also the matrix representation is more effective than the linear positional contribution form, which further proves the positional information should be taken into account for sentiment analysis or other NLP tasks.

Keywords: Deep belief networks, Sentiment classification, Positional information, Matrix representation

1 Introduction

Sentiment classification task is a popular research issue in the NLP community, which is to determine the sentiment polarity of a text. The fast growing amount of online opinion resources, such as product review sites and social media websites, has produced much interest on the task of sentiment classification [8,6,18]. Therefore, people in academic and engineering areas are paying attention to develop an automatic system that can identify the polarities of the opinions.

Various techniques have been applied in sentiment analysis area, including traditional machine learning approaches, language models [11,19], and the recently developed deep learning methods, such as deep neural networks [2], deep belief networks [15], recursive matrix-vector model [16], and recursive deep network [17]. Pang *et al* use three traditional machine learning methods: naive Bayes (NB), maximum entropy (ME) classification, and support vector machines

P. Perner (Ed.): ICDM 2017, LNAI 10357, pp. 1–15, 2017.
DOI: 10.1007/978-3-319-62701-4_1

(SVMs) for movie sentiment classification [10]. Raychev and Nakov introduce a novel language independent approach to the task of determining sentiment polarities of the author's opinion regarding a specific topic in natural language texts [12]. It intakes the positional information into NB classifier. In particular, it introduces a method to measure the positional importance, that is, instead of value one, the occurrences of the words at different positions contribute different values to their frequency values in the sentence. The DBN [5] is an effective deep learning approach consisting of multiple layers of hidden variables, which are accommodated to abstract higher representations from the raw inputs. Sarikaya *et al* use additional unlabeled data for DBN pre-training and combine DBN-based learned features for text classification [15].

Some researchers consider the word order information into several different NLP tasks. Pahikkala *et al* introduce a framework based on a word-position matrix representation of text for natural language disambiguation tasks [9]. Specifically, in disambiguation tasks, each input example is consisting of a word to be disambiguated and its surrounding context words, then a kernel function is applied to map the features. Johnson and Zhang propose an effective way of bag-of-words conversion into convolutional layer to exploit the word order of text for text categorization using convolutional neural networks (CNNs) [7]. In detail, the region representation called *seq-CNN* is to embed text regions into low dimensional vector space. Meanwhile, considering the model becomes too complex and the training is too expensive, they provide an alternative way *bow-CNN* to perform bag-of-words conversion to make region vectors.

In this paper, we propose two ways to incorporate the positional information into DBNs for sentiment classification. Firstly, inspired by [12], we propose a new linear positional contribution form by scale normalization. Secondly, motivated by the work in [9,7], we propose a new word-to-segment matrix to denote a sentence, in which each sentence is divided into several segments (not convolutional). We do not consider the word order within each segment, but the segment order is represented using the word-to-segment matrix. Overall, this paper explores the effect of positional information on the sentiment classification task for short texts such as Twitter messages. Compared to the basic bag-of-word representation, the word positional information will capture some grammar information of a text.

The rest of the paper is organized as follows. In Section 2, we introduce three types to represent positional information; In Section 3, the experiments and results are described in detail; In Section 4, we come to the conclusion and discuss some future work.

2 DBN Incorporating Positional Information

2.1 Positional Contribution

Typically, a word is a grammar component in a sentence, and it has neighboring words resulting in different combinations of words, which means each word has a

positional contribution value in the sentence. Basically, a word should be assigned a relatively higher weight as its position occurrence increases in the sentence. The reason is that when we read the first word, the sentence's sentiment polarity is not clear at all, but when we go to the last word after reading all previous words, the sentiment polarity becomes unambiguous.

The positional information has been introduced into the NB classifier [12] in the form of Eq. 1, in which a simple linear interpolation is used to measure the position-dependent information in a sentence. Suppose the vocabulary has N word attributes, W_j denotes the j^{th} word (attribute) in the vocabulary, and it has value of x_j that is equal to zero when word W_j does not exist in the sentence, otherwise it is calculated via Eq. 1.

$$x_j = q_0 + q \cdot \frac{p}{n}, q_0 \geq 0, q > 0, j = 1, 2, ..., N, \tag{1}$$

where q_0 represents some constant value from the starting position of the sentence, and q is the position fractional weight, p is the position occurrence of word W_j in the sentence, n is the length of the sentence.

In order to normalize the values from Eq. 1 within the same scale, we force the value x_j to fall in the range of $[0, 1]$ (scale normalization). So we propose a new positional contribution form described in Eq. 2.

$$x_j = \theta + (1 - \theta) \cdot \frac{p}{n}, 0 \leq \theta \leq 1, \tag{2}$$

where p and n represent the same as those in Eq. 1, θ is the ratio between the word's presence and its positional contribution within the range of [0, 1]. When $\theta = 0$, the value only represents the positional contribution, while it means the traditional presence value if $\theta = 1$. Actually, the form of Eq. 2 is a special form of Eq. 1 just through the assumption of $q + q_0 = 1$, but it has two parameters. However, given that $0 < p \leq n$ is correct for each sentence because p represents the word's position in a sentence while n denotes the length of the sentence, this form offers the flexibility to adjust the ratio θ to assign the value into the interval of [0, 1] that is fed into the DBNs. Each sentence incorporating the positional contribution is represented by an N-dimensional vector.

2.2 Matrix Representation

To better represent a sentence with the vocabulary words and incorporate the word order information into the model, also inspired by the linear transformation of the word-position matrix representation in the word disambiguation task [9], we try to use a matrix to represent a sentence. Intuitively, word-to-word matrix representation is introduced here. Suppose that a sentence is represented by an $N*N$ matrix M with the following two definitions:

- $M_{ii} = 1$, if the word W_i exists in the sentence, otherwise is 0, for i = 1, 2, ..., N;

– $M_{ij} = 1$,if the word W_i occurs before W_j with only one space (the two words are neighbors in the sentence), otherwise is 0, for i, $j = 1, 2, ..., N$ (i is not equal to j).

The word-to-word matrix representation can exactly describe the word order in each sentence. However, because there is normally a large vocabulary size for each training data set, the word-to-word matrix representation will consequently result in an extremely large size of input for training. Specifically, if the vocabulary size is N, the word-to-word matrix will be N^2, then the training time will be squarely increased, which will also cost too much memory of the computer.

In order to decrease the dimension size, we come up with an idea: each sentence is roughly divided into three segments in grammar order of the sentence. For example, the sentence "*the cat sat on that mat*" is divided into three segments as: segment 1 of {"*the cat*"}, segment 2 of {"*sat on*"}, and segment 3 of {"*that mat*"}. In this case, we do not consider the word order in a local region (within each segment), but we take into account the order information of the three segments in the sentence, which will not lose much information for short texts, and meanwhile the dimensional size is not too large.

Hence, we propose a simplified form of word-to-word matrix representation called word-to-segment matrix representation. But there is still some difference between word-to-word and word-to-segment: word-to-word is denoted by a 0-1 matrix, while word-to-segment would distinguish word's impact on different segments (detailed in the later matrix definitions). Based on the different impact relationship values, we propose two types of matrix representations. Considering the words in a sentence are normally organized in a logical order, and also to simplify the problem, the number of segments is not defined from the grammar view, but from the view that the number of words in each segment would be approximately similar. Specifically, let $nSeg$ denote the number of segments, n is the sentence length for a specific sentence. Then we define $nSeg$ as below:

– If $n < nSeg$, then each segment is at most one word in the sentence, for example, the sentence "*love it*" to be divided into three segments, then the first segment is {"*love*"}, the second segment is {"*it*"}, while the third segment is empty;
– If $n \geq nSeg$, the length of each segment $segLen$ (except for the last segment) is defined as the floor integer of n over $nSeg$. In detail, for the segment k through 1 to $nSeg - 1$, the position of segment k in the sentence is from $1 + (k-1) * segLen$ to $k * segLen$; while for the segment $nSeg$, the position is from $k * segLen + 1$ to n. For example, a sentence "*it is the best*" to be divided into three segments in order is {"*it*"}, {"*is*"}, and {"*the best*"}.

There are two types of impact definitions introduced here. Let the word-to-segment matrix denoted by M (each column represents each word in the vocabulary and each row denotes each segment), its element value is defined in the following two types accordingly. For both two types' definitions, $M_{ij} = 1$ when the word W_j occurs in the segment i for i = 1,2,3 and j = 1,2, ..., N,

which means this word has full impact on its own segment. The word's impact on different segments is described below.

To clearly explain the word-to-segment matrix representations, we take the previously mentioned sentence "*the cat sat on that mat*" for example, and assume the vocabulary is {"*are*", "*cat*", "*mat*", "*on*", "*sat*", "*take*", "*that*", "*the*"} with size $N = 8$. The first type is illustrated in Table 1, where the word-to-segment matrix M has size of 3*8 (same in Table 2, and also the empty cells are zeros). In Table 1, the words existing in the first segment(S1: "*the cat*") have no impact on the latter two segments, the words in the second segment (S2: "*sat on*") have impact value of a on the first segment, and the words in the third segment (S3: "*that mat*") have impact on both the second and first segments with impact values a and b respectively. Given that the distance between the third segment and the first segment is larger than that between the third segment and the second segment, the impact values should satisfy the condition of $0 < b < a < 1$.

Table 1: Type one of word-to-segment matrix representation for one sentence (matrix1)

Segment	*are*	*cat*	*mat*	*on*	*sat*	*take*	*that*	*the*	words
S1		1	b	a	a		b	1	"*the cat*"
S2			a	1	1		a		"*sat on*"
S3			1				1		"*that mat*"

The second type is shown in Table 2. Another reasonable assumption is that a word in the segment not only has impact on its previous segment but also on its latter segment, but does not have impact on its remote segment (e.g., the first and the third segments). Furthermore, the three segments do not have the same impact on each other because they locate in different positions of the sentence. Specifically, the words in segment 1 (or segment 3) have impact on segment 2 with value c (or e), and the words in segment 2 have impact on both segment 1 and segment 3 with value d. The impact values have the restriction of $0 < c < d < e < 1$.

Table 2: Type two of word-to-segment matrix representation for one sentence (matrix2)

Segment	*are*	*cat*	*mat*	*on*	*sat*	*take*	*that*	*the*	words
S1		1		d	d			1	"*the cat*"
S2		c	e	1	1		e	c	"*sat on*"
S3			1	d	d		1		"*that mat*"

2.3 DBN settings

Through the definitions of the above two types of word-to-segment representations for each sentence, the input into the DBN model is transformed via the average operation over each column of the matrix. That is, each sentence is represented by an N-dimensional vector and each element value equals to the average of corresponding column of the three segments, which not only keeps the same size of input features as the original bag-of-words representation, but also contains the prior segment order information of the sentence.

Therefore, compared to traditional bag-of-words representation, there are overall four kinds of inputs: baseline bag-of-words representation, positional representation, matrix1 representation and matrix2 representation. The inputs are then fed into our DBNs respectively for further experimental comparison.

The DBNs introduced in our experiments are similar in [5], which includes RBMs to train the initial weights in an unsupervised way and then transmitted to ordinary neural networks for back propagation. In the DBNs for sentiment classification, from the visible input to the penultimate layer, we accommodate the widely used sigmoid function as the hidden neuron in Eq 3.

$$\varphi(v) = sigm(v) = \frac{1}{1 + e^{-v}}, \tag{3}$$

where v is the independent variable for sigmoid function φ.

Besides, there are some function options from the penultimate layer to output layer, such as softmax, and sigmoid functions. However, the sigmoid function proves to be more effective in both neuron models in this paper, which is written as Eq 4.

$$Label^S = sigm(w^{out} \cdot S + b^{out}), \tag{4}$$

where w^{out} is a weight matrix connecting the output layer and its previous hidden layer, b^{out} is the corresponding bias vector, and S represents the "Sentence" of the penultimate layer, and $Label^S$ is finally calculated as a C-vector (C is the number of classes) in which the largest value indicates its class.

The training process of DBNs is divided into two steps [5]: unsupervised RBM training and supervised neural network training. The first step follows the practical guide written by Hinton [4], and the second one is actually the traditional back propagation, a widely applied method to fine-tune the weights, introduced by Rumelhart *et al* [13]. Specially, for each data set, the same size of input is fed into the DBNs, so the running time difference lies in the computation of different input transformations that will not cost much time compared to DBN training. Therefore, here we focus on their classification performance on total accuracy.

3 Experiments and Results

In this section, we design a variety of experiments to verify the power of positional information of sentences in the DBNs. Then the experimental results are

compared from different angles to analyze the effect and sensitivity of positional information for sentiment classification.

3.1 Data Collection and Pre-processing

We mainly focus on short text sentiment classification since the positional representation and word-to-segment matrix representation described above will lose some important effect for long text sentiment analysis. Therefore, several short text data sets, e.g. Twitter messages, are selected here for implementations.

(1) STS-T: Stanford Twitter Sentiment (STS) Test Set [14], a manually annotated data set of STS with positive and negative labels.
(2) STS-G: a gold data set extracted from STS with positive and negative labels [14].
(3) SST: Sentiment Strength Twitter data set [14], including three sentiment labels (positive, negative and neutral). We will use the data set as two, one is tri-class data set and the other is binary class removing neutral tweets.
(4) HCR: Health Care Reform (HCR) Twitter data set [14], including three classes (positive, negative and neutral). Similar as (3), the data set is used a tri-class set and a binary class set.
(5) FT: Full Twitter data [1],including three classes (positive, negative and neutral). Similar as (3), the data set is used of tri-class data set and binary class data set.
(6) GT: Game Tweets regarding the video games, are real-time collected and labeled by us with three labels (positive, negative and neutral). Also, this data set is used for tri-class and binary data sets.
(7) HCR2, SST2, FT2, GT2: These four data sets are respectively derived from HCR, SST, FT and GT with neutral tweets removed for binary classification.

Each raw data set needs to be pre-processed for training the model. Firstly, all characters are converted to lowercase since upper case and lower case have no differences for sentiment polarities; Secondly, the URLs in the data set are removed, because they do not make much sense for the sentiment; Thirdly, we transform some acronyms and abbreviations to their completely expanded forms. For example, "*i've*" is replaced by "*i have*", "*can't*" or "*cannt*" is "*can not*", "*won't*" or "*wont*" is "*will not*", "*shouldn't*" or "*shouldnt*" is "*should not*", with details in Table 3. In this way some meaningful words especially the negation word "*not*" are kept as they are essential for sentiment. Finally, some punctuations, such as @, /, |, $, are deleted as well since they contribute little to the text sentiment.

After the above pre-processing, we need to obtain initial trainable data that can be directly used in DBNs. Since Twitter texts are all limited to 140 characters long, each word in one Twitter text occurs only once for most of the time, so the vocabulary of each data set is extracted as attributes, each word token is denoted by its presence or not, and then each sentence (training example) can be represented by a vector where the element is one if the word exists in it,

Table 3: Corresponding expanded forms for abbreviations

abbrev.	expanded forms	abbrev.	expanded forms
i've	i have	won't/wont	will not
i'm	i am	wouldn't/wouldnt	would not
i've	i have	shouldn't/shouldnt	should not
i'll	i will	can't/cannt	can not
it's	it is	couldn't/couldnt	could not
let's	let us	isn't/isnt	is not
she's	she is	wasn't/wasnt	was not
he's	he is	aren't/arent	are not
she'll	she will	weren't/werent	were not
he'll	he will	don't/dont	do not
you've	you have	doesn't/doesnt	does not
there're	there are	didn't/didnt	did not
there's	there is	haven't/havent	have not

otherwise zero (filtered by the *StringToWord* in Weka 3.6.12 [3]). Besides, the sentence's label is denoted by a vector with value one at corresponding position and zeros elsewhere. For example, here is an example with positive label in a binary classification task, then its label vector (output) can be defined as (1, 0) in which the first element denotes positive while the second element is negative.

Table 4: Summary statistics of data sets used in the experiments

Data set	size	N	avgL	maxL	C	class sizes	mini-batch
STS-T	495	2067	14.5	32	3	179/139/177	11
STS-G	2000	1172	16.4	33	2	632/1368	50
HCR	2300	1018	18.8	32	3	541/400/1359	46
HCR2	1900	1084	19.1	32	2	541/1359	38
SST	4000	985	16.6	37	3	1251/1800/949	50
SST2	2200	1030	17.6	37	2	1251/949	44
FT	5000	1066	14.0	34	3	1664/1664/1672	50
FT2	3250	1162	15.2	34	2	1625/1625	50
GT	12000	929	13.8	33	3	3983/4013/4004	50
GT2	8000	984	14.3	33	2	3984/4016	50

*size: number of examples; N: number of words extracted as the vocabulary; avgL: average text length; maxL: maximal text length; C: number of classes; class sizes: number of examples for positive/neutral/negative if $C = 3$, otherwise for positive/negative if $C = 2$; mini-batch: the size of mini-batch for each data set in the experiments.

3.2 Results and Analysis

In this paper we implement the experiments based on the following four model variations:

(1) DBN-presence: DBN model with the basic bag-of-words representation (presence / non-presence) as input with $\theta = 1$ in Eq. 2.
(2) DBN-position: DBN model with the vocabulary incorporating the word's presence and positional contribution value as input with $0 \leq \theta < 1$ in Eq. 2.
(3) DBN-matrix1: DBN model with the average of first type word-to-segment matrix representation as input. Each word attribute has value one if existing in its own segment of the sentence (full impact on its own segment), while it only has impact on its all preceding segments. That is, the words existing in the first segment have no impact on the other two segments, the words in the second segment have some impact (value a) on the first segment, and finally the words in the third segment have impact on both the second and first segments with decreasing impact values of a and b respectively in Table 1, with the restriction of $0 < b < a < 1$.
(4) DBN-matrix2: DBN model with the average of second type word-to-segment matrix representation as input. Each word attribute has value one if it exists in its own segment of the sentence, and also has impact on its nearest segment(s) including its preceding and posterior segments with the impact values of c, d, e with $0 < c < d < e < 1$ in Table 2.

We perform the above four models on ten data sets listed in Table 4. The DBN structure is manually set to consist of two hidden layers. Specifically, it is: input (visible units) $\rightarrow$ 400 hidden units $\rightarrow$ 100 hidden units $\rightarrow$ output layer (class labels). Meanwhile, the NB classifier is also performed for reference comparison. On the other hand, to speed up the experiments in this paper, each data set is performed using five-fold cross validation (four for training and one for testing) to obtain an average accuracy. For some hyper-parameters in our experiments, firstly in RBM unsupervised training, the momentum is 0.1 and learning rate is 0.1, the number of epochs is set as 50. Secondly, for supervised BP training, the mini-batch sizes are listed in Table 4, the sparsity penalty parameter is 0.1, and the maximum number of epochs is 500 with 10^{-6} as the convergence control based on early stopping rule.

We firstly manually set the parameters for the classification results and give a comparison among different DBN model variations. Each accuracy value in the following tables is average total accuracy of cross validation. Then, we perform a range of experiments to analyze the effect of parameters with respect to θ, (a, b), and (c, d, e), investigating whether there exist some hidden patterns for these parameters in different data sets or whether they are robust to the classification performance. The DBNs are performed on the platform of MATLAB 2014a and the NB model is performed in Weka 3.6.12 in the PC of 64-bit OS, Intel Core i5-5200U, CPU 2.20GHz, and RAM 8.0GB.

Classification Results. To obtain the classification results, we set the parameters for each data set listed in Table 5 for experimental results through a number of tests.

Table 5: Positional parameters of each data set used in the experiments

Model	Position (θ)	Matrix1 (a, b)	Matrix2 (c, d, e)
STS-T	0.3	(0.6, 0.1)	(0.3, 0.4, 0.6)
STS-G	0.6	(0.8, 0.3)	(0.3, 0.4, 0.8)
HCR	0.9	(0.6, 0.5)	(0.2, 0.5, 0.7)
HCR2	0.6	(0.8, 0.3)	(0.2, 0.5, 0.8)
SST	0.8	(0.4, 0.3)	(0.2, 0.4, 0.7)
SST2	0.3	(0.8, 0.1)	(0.2, 0.5, 0.7)
FT	0.4	(0.6, 0.1)	(0.3, 0.5, 0.8)
FT2	0.9	(0.8, 0.3)	(0.2, 0.5, 0.8)
GT	0.7	(0.6, 0.5)	(0.2, 0.5, 0.8)
GT2	0.7	(0.8, 0.3)	(0.2, 0.4, 0.7)

The classification results of different model variations on the ten data sets are shown in Table 6. Note that the NB classifier is performed based on the presence/non-presence word features of each data set. The accuracy values of the three models (DBN-position, DBN-matrix1, DBN-matrix2) are respectively compared with DBN-presence using a two-tailed t-test. The numbers followed by the sign * indicate the accuracy passes the t-test at the significance level of 95% for each data set. The results indicate that all the four DBN models perform better than the NB model (average 63.74%). Meanwhile, DBN-matrix2 has a relatively highest average accuracy (68.26%) for all the data sets, and DBN-presence performs worse than the other three DBN models.

In Table 7, an explicit comparison among the four model variations is summarized, where $w/l/t$ indicates that the model wins in w data sets, loses in l data sets, and ties in t data sets. In Table 7, the three rows (DBN-position, DBN-matrix1, and DBN-matrix2) show their comparisons with DBN-presence respectively. Especially, because the DBN-presence only has ties or losses with the other three models, it is omitted in Table 7. To summarize more clearly, the models are compared in three aspects: all data sets (10), tri-class data sets (5) (Note: five data sets with three labels: STS-T, HCR, SST, FT, and GT), and binary data sets (5) (Note: five data sets with only two labels: STS-G, HCR2, SST2, FT2, and GT2.).

Table 7 shows that DBN-matrix2 performs best on all the ten data sets for one-by-one comparisons (seven wins and three ties). Besides, the DBN-matrix1 and DBN-position also have four wins and five wins respectively, while the DBN-presence only has loses or ties. Consequently, it is obvious that the positional information of sentences exactly provides positive effect on the sentiment classification issues, and also the second type matrix representation is more effective than the positional contribution form. On the other hand, the positional infor-

Table 6: Experimental results on classification accuracy (%)

Model	NB	DBN-presence	DBN-position	DBN-matrix1	DBN-matrix2
STS-T	66.9	61.01	64.65 *	65.66 *	66.06 *
STS-G	79.7	83.50	84.85	82.60	83.25
HCR	63.1	64.91	66.40 *	65.61	66.33 *
HCR2	74.4	78.11	78.66	78.26	78.89
SST	51.7	53.40	54.45	54.80	55.30 *
SST2	66.5	70.00	71.59 *	71.64 *	72.77 *
FT	47.0	51.16	52.70 *	52.56 *	53.44 *
FT2	63.3	68.18	68.71	68.95	68.92
GT	54.4	57.69	59.93 *	60.73 *	60.71 *
GT2	70.4	75.39	75.83	75.46	76.94 *
average	63.74	66.34	67.78	67.61	**68.26**

Table 7: Summary of classification accuracy comparisons

Model	all data sets (10)	tri-class data sets (5)	binary data sets (5)
DBN-position	5/0/5	4/0/1	1/0/4
DBN-matrix1	4/0/6	3/0/2	1/0/4
DBN-matrix2	7/0/3	5/0/0	2/0/3

mation seems more effective for tri-class data sets, as DBN-matrix2 has five wins for all the five tri-class data sets, and DBN-position and DBN-matrix1 have four wins and three wins out of five respectively. While the results for binary data sets are not so good as tri-class data sets. It is probably because the word position and the word order will account more exact information for the sentiment. So the more positional information is integrated, the better for multi-class sentiment classification.

In essence, the positional contribution form and word-to-segment matrix representation are different ways to describe the word positional information of a sentence. For positional contribution form, the contribution values are linearly augmented as the position increases; for matrix1 representation, the words have relatively larger weights in the latter segments, but they are the same in the same segments; while for matrix2 representation, the words in the middle segments have largest weights since they play impact on both its previous and latter segments. To summarize, the word positional representations improve the word presence features for short text sentiment classification.

Effect of Parameters. In our experiments, there are some important parameters which need to be set manually. Whether the results are sensitive to the

parameters needs further investigation. Hence, we perform a variety of experiments in DBNs to examine the effect of parameters.

Firstly, we investigate the effect of position parameter θ in Eq. 2. Here the values are set from 0.0 through 1.0 with the interval of 0.1. Each data set is performed with five-fold cross validation for each parameter. The results are shown in Table 8.

Table 8: Classification accuracy of four data sets vs. position parameter

θ	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
STS-T	63.03	58.18	63.03	**64.65**	61.21	61.01	60.81	61.62	57.37	57.37	61.01
STS-G	81.65	82.30	82.75	83.00	83.50	83.50	**84.85**	82.85	84.25	83.75	83.50
HCR	64.30	64.96	64.74	66.00	64.83	65.91	65.78	65.00	64.61	**66.43**	64.91
HCR2	77.89	77.32	76.95	77.95	78.42	78.05	**78.66**	78.11	78.16	78.00	78.11
SST	52.05	51.80	53.85	53.33	53.78	53.73	53.80	54.33	**54.45**	52.50	53.40
SST2	69.95	70.82	71.00	**71.59**	71.41	70.82	71.05	71.09	70.45	69.91	70.00
FT	49.88	49.96	51.28	52.18	**52.70**	51.10	52.28	51.42	51.44	51.90	51.16
FT2	66.83	68.68	67.69	68.43	68.52	67.91	68.09	68.55	67.57	**68.71**	68.18
GT	56.18	58.85	57.74	58.93	59.66	59.81	59.10	**59.93**	59.58	59.34	57.69
GT2	74.00	74.59	75.84	74.15	74.76	75.05	75.43	**75.83**	75.58	74.53	75.39

Table 8 shows the effect of position parameter θ on the average classification accuracy of the ten data sets. In this table, the last data column for each data set with position ratio at 1.0 represents the traditional bag-of-words representation (presence), and the bold number is the highest accuracy value in each row. It means that different ratio values between the word's presence and its positional contribution have different contributions to the text's sentiment. Normally, if θ is relatively bigger, less positional information is integrated, while the word contributes more information if θ is smaller. It indicates that, the parameter value of θ needs a careful investigation for each data set. In Table 8, the θ value corresponding with the bold number for each row is selected for the comparison on the previous classification results (similar in Table 9 and Table 10).

Secondly, the matrix1 representation is investigated for the values of $a = 0.2$, 0.4, 0.6, 0.8 and $b = 0.1$, 0.3, 0.5, with the restriction of $0 < b < a < 1$. Table 9 shows the results of accuracy on each set of (a, b). For instance, the data set of STS-T has the highest average accuracy of 65.66% at (0.6, 0.1) among all the nine parameter settings for matrix1 representation. It gives a picture of the sensitivity for the values of (a, b) on the performance and provides evidence to choose good parameters.

Finally, the matrix2 representation is performed with the parameter values of $c = 0.1$, 0.2, 0.3, $d = 0.4$, 0.5 and $e = 0.6$, 0.7, 0.8. The accuracy results of each set of (c, d, e) are given in Table 10. For example, the parameter setting $(c = 0.3, d = 0.5, e = 0.8)$ reaches the highest accuracy 53.44% for FT data set.

Table 9: Classification accuracy of four data sets vs. matrix1 parameters

(a, b)	(2,1)	(4,1)	(6,1)	(8,1)	(4,3)	(6,3)	(8,3)	(6,5)	(8,5)
STS-T	64.44	60.00	**65.66**	64.85	61.21	65.45	64.65	65.25	63.84
STS-G	68.40	68.40	72.15	71.40	73.35	74.20	**82.60**	77.85	81.90
HCR	62.13	65.48	65.26	65.13	64.74	64.17	65.09	**65.61**	64.57
HCR2	72.63	73.53	77.53	77.26	77.42	76.89	**78.26**	78.21	77.63
SST	54.23	53.78	52.83	51.10	**54.60**	52.70	53.40	53.25	53.78
SST2	62.27	70.64	70.09	**71.64**	70.82	71.05	70.77	69.41	69.91
FT	45.82	49.84	**52.56**	51.48	50.42	51.72	52.12	49.78	52.36
FT2	48.92	52.55	58.49	65.02	52.00	65.26	**68.95**	67.45	68.28
GT	46.17	48.72	55.51	59.79	51.37	50.78	60.05	**60.73**	59.97
GT2	64.08	64.99	70.78	75.16	74.75	74.91	**75.46**	71.26	75.44

*E.g.(2,1) indicates that (a, b) is (0.2, 0.1), others are similar.

The results of different parameter settings not only show how to choose the parameters, but also demonstrate these parameters play a significant role in DBN models, because the performance results are not very robust to those parameters. Furthermore, it is possible that some other better parameters are not included in our experiments since it is difficult to perform all experiments with exhaustive parameter searching. However, it is really necessary to point out that the positional information (positional contribution and word-to-segment) affects the sentiment classification positively if we set the right parameters.

4 Conclusions and Future Work

In this paper, we propose several ways to incorporate the positional information of texts into DBNs with four model variations for sentiment classification and perform a variety of experiments to verify the effect of positional information towards the sentence sentiments. By choosing the good parameters, the experiments reveal that the word position and word order do provide positive effect on the classification performance. The results indicate that the traditional bag-of-words representation can be improved by incorporating some positional information represented by the word positional contribution form and the word-to-segment matrix representation. Also, it can be seen that positional information works more effective for tri-class classification compared to binary classification. In other words, each word attribute in a sentence has different effect for sentence sentiment.

In the future, this work can be improved from the following views: (1) For the positional representation, the linear positional transformations implemented in this paper seem too simple, and some more sophisticated curve functions (e.g. logit function or symmetrical function) probably perform relatively better; (2)

Table 10: Classification accuracy of four data sets vs. matrix2 parameters

(c, d, e)	(1,4,6)	(1,4,7)	(1,4,8)	(1,5,6)	(1,5,7)	(1,5,8)	(2,4,6)	(2,4,7)	(2,4,8)
STS-T	61.62	65.66	65.86	64.04	60.81	62.02	65.25	65.05	59.80
STS-G	69.20	68.40	82.75	71.35	82.25	83.10	78.60	73.70	79.95
HCR	65.78	65.43	64.43	65.09	64.70	65.35	64.65	65.22	65.17
HCR2	77.00	77.95	77.95	77.00	77.11	76.68	77.68	77.16	76.68
SST	54.15	53.98	52.98	52.90	53.45	53.15	53.15	**55.30**	53.23
SST2	70.41	71.45	71.41	71.36	71.45	71.14	72.14	72.32	71.64
FT	49.80	51.92	52.18	50.14	50.96	53.00	52.82	50.58	52.44
FT2	67.66	68.43	68.40	63.66	68.09	68.28	66.95	65.02	64.28
GT	58.40	55.41	54.12	57.57	55.90	58.96	56.08	56.79	58.34
GT2	71.94	76.06	76.35	70.89	75.59	75.15	76.15	**76.94**	71.40
(c,d,e)	(2,5,6)	(2,5,7)	(2,5,8)	(3,4,6)	(3,4,7)	(3,4,8)	(3,5,6)	(3,5,7)	(3,5,8)
STS-T	63.23	62.42	64.04	**66.06**	64.04	63.64	61.41	64.85	61.21
STS-G	81.20	83.10	83.15	82.20	79.30	**83.25**	83.15	83.20	77.70
HCR	63.91	**66.33**	65.22	65.57	64.22	65.30	65.30	64.87	65.00
HCR2	77.79	77.47	**78.89**	77.00	75.21	76.63	77.42	76.74	77.68
SST	54.13	53.65	53.60	54.39	53.08	53.40	54.05	53.28	53.13
SST2	72.36	**72.77**	71.36	71.14	68.32	70.14	72.05	72.14	71.68
FT	52.24	52.86	52.42	53.14	51.76	52.96	52.72	52.40	**53.44**
FT2	66.03	67.23	**68.92**	64.89	67.91	67.26	66.31	68.43	68.03
GT	58.40	56.28	**60.71**	43.08	60.21	60.31	59.19	56.75	56.06
GT2	75.54	75.99	76.29	76.15	75.80	75.74	76.20	76.30	75.26

*E.g. (1,4,6) means (c, d, e) is (0.1, 0.4, 0.6). Similar to other numbers.

Try to extend the matrix representation into a row vector for each sentence (even though it will be an extremely large sparse matrix and cost much memory during the training, this is an approach), letting each element value in the matrix be a single feature into the model; (3) Only three segments are introduced here, more segments (e.g. four or more segments) for a sentence (or long text) may be more reasonable, which will probably account more word order information into the model. These will be some of our future research directions.

References

1. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., Passonneau, R.: Sentiment analysis of twitter data. In: Proceedings of the workshop on languages in social media. pp. 30–38. Association for Computational Linguistics (2011)
2. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning. pp. 160–167. ACM (2008)

3. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter 11(1), 10–18 (2009)
4. Hinton, G.: A practical guide to training restricted boltzmann machines. Momentum 9(1), 926 (2010)
5. Hinton, G., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural computation 18(7), 1527–1554 (2006)
6. Hu, X., Tang, J., Gao, H., Liu, H.: Unsupervised sentiment analysis with emotional signals. In: WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web. pp. 607–617 (2013)
7. Johnson, R., Zhang, T.: Effective use of word order for text categorization with convolutional neural networks. In: NAACL HLT 2015 - 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference. pp. 103–112 (2015)
8. Liu, B.: Sentiment analysis and opinion mining. Synthesis lectures on human language technologies 5(1), 1–167 (2012)
9. Pahikkala, T., Pyysalo, S., Boberg, J., Jrvinen, J., Salakoski, T.: Matrix representations, linear transformations, and kernels for disambiguation in natural language. Machine Learning 74(2), 133–158 (2009)
10. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. pp. 79–86. Association for Computational Linguistics (2002)
11. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. pp. 275–281. ACM (1998)
12. Raychev, V., Nakov, P.: Language-independent sentiment analysis using subjectivity and positional information. In: RANLP. pp. 360–364 (2009)
13. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cognitive modeling 5 (1988)
14. Saif, H., Fernandez, M., He, Y., Alani, H.: Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold (2013)
15. Sarikaya, R., Hinton, G.E., Deoras, A.: Application of deep belief networks for natural language understanding. IEEE Transactions on Audio, Speech and Language Processing 22(4), 778–784 (2014)
16. Socher, R., Huval, B., Manning, C.D., Ng, A.Y.: 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 (2012)
17. Socher, R., Perelygin, A., Wu, J.Y., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the conference on empirical methods in natural language processing (EMNLP). vol. 1631, p. 1642. Citeseer (2013)
18. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference. vol. 1, pp. 1555–1565 (2014)
19. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM Transactions on Information Systems (TOIS) 22(2), 179–214 (2004)