Gradual Learning of Matrix-Space Models of Language for Sentiment Analysis

Shima Asaadi[*] and **Sebastian Rudolph**
Faculty of Computer Science
TU Dresden, Germany
firstname.lastname@tu-dresden.de

Abstract

Learning word representations to capture the semantics and compositionality of language has received much research interest in natural language processing. Beyond the popular vector space models, matrix representations for words have been proposed, since then, matrix multiplication can serve as natural composition operation. In this work, we investigate the problem of learning matrix representations of words. We present a learning approach for compositional matrix-space models for the task of sentiment analysis. We show that our approach, which learns the matrices gradually in two steps, outperforms other approaches and a gradient-descent baseline in terms of quality and computational cost.

1 Introduction

Recently, a lot of NLP research has been devoted to word representations with the goal to capture language semantics, compositionality, and other linguistic aspects. A prominent class of approaches to produce word representations are Vector Space Models (VSMs) of language. In VSMs, a vector representation is created for each word in the text, mostly based on distributional information. One of the recent prominent methods to extract vector representations of words is Word2vec, introduced by Mikolov et al. (2013a; 2013b). These models measure both syntactic and semantic aspects of words and also seem to exhibit good compositionality properties. The principle of compositionality states that the meaning of a complex expression can be obtained from combining the meaning of its constituents (Frege, 1884). In the Word2vec case and many other VSM approaches, some vector space operations (such as vector addition) are used as composition operation.

One of the downsides of using vector addition (or other commutative operations like the component-wise product) as the compositionality operation is that word order information is inevitably lost. Alternative word-order-sensitive compositionality models for word representations have been introduced, such as Compositional Matrix-Space Models (CMSMs) (Rudolph and Giesbrecht, 2010). In such models, matrices instead of vectors are used as word representations and compositionality is realized via matrix multiplication. It has been proven that CMSMs are capable of simulating a wide range of VSM-based compositionality operations. The question, however, how to learn suitable word-to-matrix mappings has remained largely unexplored with few exceptions (Yessenalina and Cardie, 2011). The task is exacerbated by the fact that this amounts to a non-convex optimization problem, where a good initialization is crucial for the success of gradient descent techniques.

In this paper, we address the problem of learning CMSM in the domain of sentiment analysis. As has been observed before, the sentiment of a phrase is very much influenced by the presence and position of negators and modifiers, thus word order seems to be particularly relevant for establishing an accurate sentiment score.

We propose to apply a two-step learning method where the output of the first step serves as initialization for the second step. We evaluate the performance of our method on the task of fine-grained sentiment analysis and compare it to a previous work on learning CMSM for sentiment analysis (Yessenalina and Cardie, 2011). Moreover, the performance of our representation learning in sentiment composition is evaluated on sentiment composition in opposing polarity phrases

[*]Supported by DFG Graduiertenkolleg 1763 (QuantLA)

(Kiritchenko and Mohammad, 2016b).

The rest of the paper is organized as follows. Section 2 provides the related works. A detailed description of the approach is presented in Section 3, followed by experiments and discussion in Section 4, and the conclusion in the last section.

2 Related Work

Compositional Distributional Semantics: In compositional distributional semantics, different approaches for learning word and phrase representations and ways to compose the constituents are studied. As an early work in compositional distributional semantics, Mitchell and Lapata (2010) propose vector composition models with additive and multiplicative functions as the composition operations in semantic VSMs. These models outperform non-compositional approaches in semantic similarity of complex expressions. Mikolov et al. (2013a) propose Word2vec where continuous vector representations of words are trained through continuous bag-of-words and skip-gram models. These models are supposed to reflect syntactic and semantic similarities of words. An extension to these models is a vector representation of idiomatic phrases by considering a vector for each phrase and training Word2vec accordingly (Mikolov et al., 2013b). Moreover, compositionality is captured in these models by applying certain mathematical operations on word vectors.

Rudolph and Giesbrecht (2010) introduced compositional matrix-space models in which words are represented as matrices, and defined composition operation as a matrix multiplication function. Learning such matrix representations can be done by supervised machine learning algorithms (Yessenalina and Cardie, 2011). Other approaches using matrices for distributional representations of words have been introduced more recently. Socher et al. (2012) introduce a model in which a matrix and a vector is assigned to each word. The vector captures the meaning of the word by itself and the matrix shows how it modifies the meaning of neighboring words. The model is learned through recursive neural networks. In the model of Socher et al. (2013), a unique tensor-based composition function in a recursive neural tensor network is introduced which composes all word vectors. Maillard and Clark (2015) describe a compositional model for learning adjective-noun pairs where, first, a vector representation for each word is trained using a skip-gram model. Then, adjective matrices are trained in composition to their nouns, using back-propagation.

Sentiment Analysis: There is a lot of research interest in the sentiment analysis task in NLP. The task is to classify the polarity of a text (negative, positive, neutral) or assign a real-valued score, showing the polarity and intensity of the text. Some contributions focus on learning sentiment of a short text based on supervised machine learning techniques (Yessenalina and Cardie, 2011; Agrawal and An, 2014). Recent approaches have focused on learning different types of neural networks for sentiment analysis, such as the work of Socher et al. (2013) which apply recursive neural tensor networks for both fine-grained and binary classification of phrases and sentences. Timmaraju and Khanna (2015) use recursive-recurrent neural networks for sentiment classification of long text, and Hong and Fang (2015) apply long short-term memory and deep recursive-NNs. In a very recent work by Wang et al. (2016), convolutional neural networks and recurrent neural networks are combined leading to a significant improvement in sentiment analysis of short text.

Sentiment Composition: Compositionality in sentiment analysis is used to compute the sentiment of complex phrases and sentences. Recent works of Kiritchenko and Mohammad (2016a; 2016b) deal with sentiment composition of phrases. In (Kiritchenko and Mohammad, 2016a), they create a dataset of unigrams, bigrams and trigrams, which contains phrases with at least one negative and one positive word. They analyze the performance of different learning algorithms and word embeddings on the dataset with different linguistic patterns. In (Kiritchenko and Mohammad, 2016b), they create a sentiment composition lexicon for phrases containing negators, modals and adverbs with their associated sentiment scores, and study the effect of modifiers on the overall sentiment of phrases.

3 The Approach

Learning appropriate word representations to extract syntactic and semantic information of compositional phrases is a complex task in sentiment analysis. In order to learn a sentiment-aware representation model, we propose the following approach: We use a compositional matrix-space model, where, as opposed to vector-space models of language, words are represented by matrices. In the following, we describe the representation model itself and the training in detail.

3.1 Model Description: Compositional Matrix-Space Model

Compositional Matrix-Space Models (CMSMs) consider compositionality in language by the following general idea: the semantic space consists of quadratic matrices carrying real values. In other words, the semantics of each word is represented by a matrix. Then, considering the standard matrix multiplication as the composition operation, the semantics of phrases are obtained by multiplying the word-matrices in the appropriate order (Rudolph and Giesbrecht, 2010). Training CMSM using machine learning algorithms yields a type of word embedding for each word, which is a low-dimensional real-valued matrix. Like for word embeddings into vector spaces, each matrix representation is supposed to contain syntactic and semantic information about the word. Since we consider the task of sentiment analysis, word embeddings must be trained to contain sentiment-related information.

More formally, let $p = x_1 \cdots x_k$ be a phrase consisting of k words. The CMSM assigns to each word x_j a matrix $W_{x_i} \in \mathbb{R}^{m\times m}$. Then the representation of p which is the composition of words in the phrase, is shown as the matrix product of the words in the same order:

$$W_p = \prod_{i=1}^{k} W_{x_i} = W_{x_1} W_{x_2} \cdots W_{x_k}$$

To finally associate a real-valued score to a phrase p, we map the matrix representation of p to a real number using two mapping vectors $\alpha, \beta \in \mathbb{R}^m$ as follows:

$$\omega_p = \alpha^\top W_p \beta$$

In our case, the final score ω_p is supposed to indicate the sentiment polarity and strength of the phrase p.

The learning task is now, given a set of d training examples (p_j, ω_j) with $1 \le j \le d$, to find matrix representation for all words occurring in all p_j, such that $\omega_{p_j} \approx \omega_j$ for all j. Thereby, we fix

$$\alpha = e_1 = \begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix} \text{ and } \beta = e_m = \begin{pmatrix}0\\\vdots\\0\\1\end{pmatrix},$$

which only moderately restricts the expressivity of our model as made formally precise in the following theorem.

Theorem 1. *Given matrices $W_1, \ldots, W_\ell \in \mathbb{R}^{m\times m}$ and vectors $\alpha, \beta \in \mathbb{R}^m$, there are matrices $\hat{W}_1, \ldots, \hat{W}_\ell \in \mathbb{R}^{(m+1)\times(m+1)}$ such that for every sequence $i_1 \cdots i_k$ of numbers from $\{1, \ldots, \ell\}$ holds*

$$\alpha^\top W_{i_1} \cdots W_{i_k} \beta = e_1^\top \hat{W}_{i_1} \cdots \hat{W}_{i_k} e_{m+1}$$

Proof. If α is the zero vector, all scores will be zero, so we can let all $\hat{W}_h$ be the $(m+1)\times(m+1)$ zero matrix.

Otherwise let M be an arbitrary $m \times m$ matrix of full rank, whose first row is α, i.e., $e_1^\top M = \alpha^\top$. Now, let

$$\hat{W}_h := \begin{pmatrix} MW_hM^{-1} & MW_h\beta \\ 0 \ \cdots \ 0 & 0 \end{pmatrix}$$

for every $h \in \{1, \ldots, \ell\}$. Then, we obtain

$$\hat{W}_g\hat{W}_h = \begin{pmatrix} MW_gW_hM^{-1} & MW_gW_h\beta \\ 0 \ \cdots \ 0 & 0 \end{pmatrix}$$

for every $g, h \in \{1, \ldots, \ell\}$. This leads to

$$\begin{aligned} & e_1^\top \hat{W}_{i_1} \cdots \hat{W}_{i_k} e_{m+1} \\ = \ & e_1^\top M W_{i_1} \cdots W_{i_k} \beta \\ = \ & \alpha^\top W_{i_1} \cdots W_{i_k} \beta \end{aligned}$$

□

In words, this theorem guarantees that for every CMSM-based scoring model with arbitrary vectors α and β there is another such model (with dimensionality increased by one) where α and β are distinct unit vectors. This justifies our above mentioned choice.

3.2 Model Training

Since the learning problem for CMSMs is not a convex problem, it must be trained carefully and specific attention has to be devoted to a good initialization (Yessenalina and Cardie, 2011). To this end, we (1) perform an "informed initialization" exploiting available scoring information for one-word phrases (unigrams), (2) apply a first learning step training only on parts of the matrices and using scored one- and two-word phrases from our training set, and (3) use the matrices obtained in this step as initialization for training the full matrices on the full training set.

Initialization: In this step, we first take all the words in the training data as our vocabulary, creating quadratic matrices of size $m \times m$ with entries from a normal distribution $\mathcal{N}(0, 0.1)$. Then, we consider the words which appear in unigram phrases $p_j = x$ with associated score ω_j in the training set. We exploit the fact that for any matrix W, computing $e_1^\top W e_m$ extracts exactly the entry of the first row, last column of W. Hence, we update this entry in every matrix corresponding to a scored unigram phrase by this value, i.e.:

$$W_{p_j} = \begin{pmatrix} \cdot & \cdots & \omega_j \\ \vdots & \ddots & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix}$$

This way, we have initialized the word-to-matrix mapping such that it leads to perfect scores on the unigram phrases.

First Learning Step: After initialization, we consider bigram phrases. The sentiment value of a bigram phrase $p_j = xy$ is computed by the standard multiplication of word matrices of its constituents in the same order and mapping to the real value using the mapping vectors $\alpha = e_1$ and $\beta = e_m$:

$$\begin{aligned} \omega_j &= e_1^\top W_x W_y e_m \\ &= \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix}^\top \begin{pmatrix} x_{1,1} \cdots x_{1,m} \\ \vdots \ \ddots \ \vdots \\ x_{m,1} \cdots x_{m,m} \end{pmatrix} \begin{pmatrix} y_{1,1} \cdots y_{1,m} \\ \vdots \ \ddots \ \vdots \\ y_{m,1} \cdots y_{m,m} \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{1,m} \end{pmatrix}^\top \begin{pmatrix} y_{1,m} \\ \vdots \\ y_{m,m} \end{pmatrix} = \textstyle\sum_{i=1}^m x_{1,i} y_{i,m} \end{aligned}$$

We observe that for bigrams, multiplying the first row of the first matrix (row vector) with the last column of the second matrix (column vector) yields the sentiment score of the bigram phrase. Hence, as far as the scoring of unigrams and bigrams are concerned, only the matrix entries in the first row and the last column are relevant – thanks to our specific choice of the vectors α and β.

This observation justifies the next learning step: we use the unigrams and bigrams in the training set to learn optimal values for the relevant matrix entries only.

Second Learning Step: Using the entries obtained in the previous learning step for initialization, we finally repeat the optimization process, using the full training set and optimizing all the matrix values simultaneously.

For both learning steps, we apply the batch gradient descent optimization method on the training set to minimize the cost function defined as the summed squared error (SSE):

$$C(W) = \frac{1}{2} \sum_{j=1}^{d} (\hat{\omega_j} - \omega_j)^2$$

Then, update the word matrices as follows:

$$W_{x_i} = W_{x_i} - \eta \times (\frac{\partial C(W)}{\partial W_{x_i}} + \lambda W_{x_i})$$

In order to avoid overfitting in optimization, a regularization term λW_{x_i} is used.

According to Petersen and Pedersen (2012) the derivative of a phrase with respect to the i-th word-matrix is computed as follows:

$$\frac{\partial \omega_j}{\partial W_{x_i}} = \frac{\partial(\alpha^\top W_{x_1} \cdots W_{x_i} \cdots W_{x_k} \beta)}{\partial W_{x_i}} = (\alpha^\top W_{x_1} \cdots W_{x_{i-1}})^\top (W_{x_{i+1}} \cdots W_{x_k} \beta)$$

If a word x_i occurs several times in the phrase, then the partial derivative of the phrase with respect to W_{x_i} is the sum of partial derivatives with respect to each occurrence of W_{x_i}.

4 Experiments

We evaluate our approach on two different datasets which provide short phrases annotated with sentiment values. In both cases, we perform a ten-fold cross-validation.

4.1 Evaluation on Sentiment-Annotated Phrases from the MPQA Corpus

Experimental Setting: For the first evaluation of the proposed approach, we use the *MPQA corpus*[1]. This corpus contains newswire documents annotated with phrase-level polarity and intensity. We extracted the annotated phrases from the corpus documents, obtaining 9501 phrases. We removed phrases with low intensity similar to Yessenalina and Cardie (2011). The levels of polari-

[1]http://mpqa.cs.pitt.edu/corpora/mpqa_corpus/

ties and intensities and their translation into numerical values are as per Table 1. For the evaluation, we apply a ten-fold cross-validation process on the training data as follows: eight folds are used as training set, one fold as validation set and one fold as test set. The initial number of iterations in the first learning and second learning steps are set to $T = 200$ each, but we stop iterating when we obtain the minimum ranking loss, $e = \frac{1}{n}\sum_i |\hat{\omega}_i - \omega_i|$, on the validation set. Finally, we record the ranking loss of the obtained model for the test set. The learning rate η and regularization parameter λ in gradient descent are set to 0.01, by experiment. The dimension of matrices is set to $m = 3$ in order to be able to compare our results to the approach described by Yessenalina and Cardie (2011), called Matrix-space OLogReg+BowInit. We call our approach Gradual Gradient descent-based Matrix-Space Models (Grad-GMSM). All implementations have been done in Python 2.7.

Polarity	Intensity	Score
negative	high, extreme	-1
negative	medium	-0.5
neutral	medium, high, extreme	0
positive	medium	0.5
positive	high, extreme	1

Table 1: Phrase polarities and intensities in the MPQA corpus and their translation into sentiment scores.

Results and Discussion: In Yessenalina and Cardie's Matrix-space OLogReg+BowInit, matrices are initialized with a bag-of-words model. Then, ordered logistic regression is applied in order to minimize the negative log-likelihood of the training data, as the objective function. L-BFGS-B is used as their optimizer. To avoid ill-conditioned matrices, a projection step is added to matrices during training, by shrinking the singular value of matrices to one.

The ranking losses obtained by Yessenalina and Cardie's and by our method are shown in Table 2. As we observe, Grad-GMSM obtained a significantly lower ranking loss than Matrix-space OLogReg+BowInit.

Table 3 shows the sentiment scores of some example phrases trained using these two methods. As shown in the table, the two approaches' results coincide regarding the order of basic phrases: the score of *"very good"* is greater than *"good"* (and both are positive) and the score of *"very bad"* is less than *"bad"* (and both are negative). Also, *"not good"* is characterized as negative by both approaches.

Method	Ranking loss
Grad-GMSM	0.3126
Matrix-space OLogReg+BowInit	0.6375

Table 2: Ranking loss of compared methods.

Phrase	Grad-GMSM	Matrix-space OLogReg+BowInit
good	0.73	2.81
very good	0.95	3.53
not good	-0.43	-0.16
not very good	-0.29	0.66
bad	-0.80	-1.67
very bad	-1.04	-2.01
not bad	0.38	-0.54
not very bad	0.32	-1.36

Table 3: Frequent phrases with average sentiment scores

On the other hand, there are significant differences between the two approaches: for example, our approach characterizes the phrase *"not bad"* as mildly positive while their's associates a negative score to it, the same discrepancy occurs for *"not very bad"*. Intuitively, we tend to agree more with our method's verdict on these phrases.

In general, our findings confirm those of Yessenalina and Cardie: *"very"* seems to intensify the value of the subsequent word, while the *"not"* operator does not just flip the sentiment of the word after it, but also dampens the sentiment of the words gradually. On the other hand, the scores of phrases starting with *"not very"* defy the assumption that the described effects of these operators can be combined in a straightforward way.

Figure 1 provides a more comprehensive selection of phrases and their associated scores by our approach. We obtained an average of $\omega(\textit{very very good}) = 1.23$, which is greater than *"very good"*, and $\omega(\textit{very very bad}) = -1.34$ less than *"very bad"*. Therefore, we can also consider *"very very"* as an intensifier operator. Moreover, we observe that the average score of $\omega(\textit{not really good}) = -0.463$ is not equal to the average score of $\omega(\textit{really not good}) = -0.572$, which demonstrates that the matrix-based compositionality operation shows sensitivity to word orders, arguably reflecting the meaning of phrases

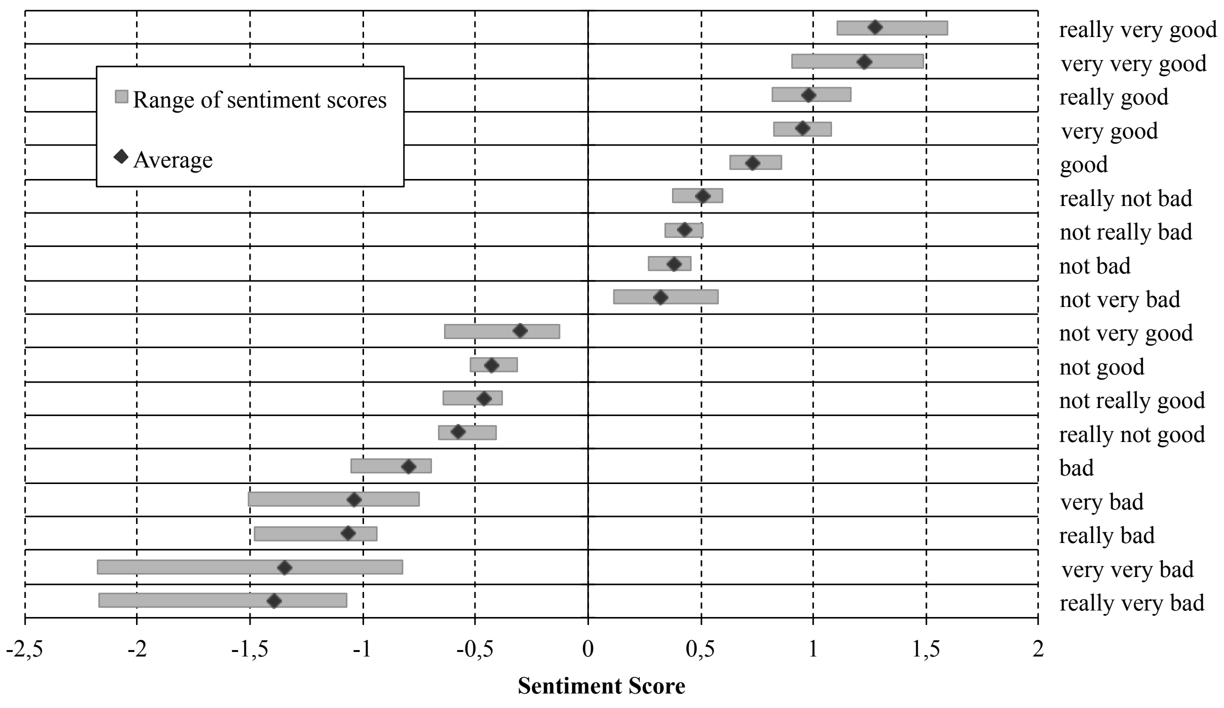

Figure 1: The order of sentiment scores for sample phrases (trained on MPQA corpus).

better than any commutative operation could.

Although the training data consists of only the values of Table 1, the training of the model is done in a way that sentiment scores for phrases with more extreme intensity might yield real values greater than $+1$ or lower than -1, since we do not constrain the sentiment scores to $[-1, +1]$. Moreover, in our experiments we observed that no extra precautions were needed to avoid ill-conditioned matrices or abrupt changes in the scores while training.

To assess the effect of our gradual two-step training method, we compared the results of Grad-GMSM against those obtained by random initialization followed by a single training phase where the full matrices were optimized (randIni-GMSM). The results, averaged over 10 runs each, are reported in Table 4. On top of a significantly better result in terms of ranking loss, we also observe faster convergence in Grad-GMSM, since the lowest ranking loss is obtained on average after 78 iterations, including both training steps. In randIni-GMSM, the lowest ranking lost happens on average after 162 iterations.

To observe the effect of a higher number of dimensions on our approach, we repeated the experiments with $m = 50$, and observed a ranking loss of $e = 0.3125$ (i.e., virtually the same as for $m = 3$) and similar values for the number of iterations confirming the observation of Yessenalina and Cardie, that increasing the number of dimensions does not significantly improve the prediction quality of the obtained model.

Method	Ranking loss	Total number of iterations
Grad-GMSM	0.3126	21.1 (Step 1) + 56.5 (Step 2) = 77.6
randIni-GMSM	0.3480	161.5

Table 4: Performance comparison for different initializations in MPQA

4.2 Evaluation on Sentiment Composition Lexicon with Opposing Polarity Phrases

Experimental Setting: For the second evaluation of the proposed approach we use the *Sentiment Composition Lexicon for Opposing Polarity Phrases (SCL-OPP)*[2]. SCL-OPP consists of 602 unigrams, 311 bigrams, and 265 trigrams, which are taken from a corpus of tweets, and annotated with real-valued sentiment scores in the interval $[-1, +1]$ by Kiritchenko and Mohammad (2016b). The multi-word phrases contain at least one negative word and one positive word. Therefore, we find this lexicon as an interesting and suitable dataset to evaluate our approach in sentiment pre-

[2] http://www.saifmohammad.com/WebPages/SCL.html

diction of opposing polarity phrases.

In this experiment, we set the dimension of matrices to $m = 200$ as in (Kiritchenko and Mohammad, 2016b) and $T = 50$. The learning rate η and regularization parameter λ in gradient descent are set to 0.1 and 0.01, respectively. In addition to the ranking loss, we also use the Pearson correlation coefficient (r) for performance evaluation, which measures the linear relation between the predicted and the annotated sentiment polarity of phrases in training data. Again, we apply ten-fold cross-validation in the same way as described before and average over ten repeated runs.

Results and Discussion: Kiritchenko and Mohammad (2016b) study different patterns of sentiment composition in phrases. They analyze the efficacy of several supervised and unsupervised methods on these phrases, and the effect of POS tags, word vector representations, etc. as their features in learning sentiment classification and regression. The word embeddings are obtained by the Word2vec model. For the task of regression, RBF kernel-based Support-Vector Regression (SVR) is applied as a supervised method, which we call *RBF-SVR*. RBF-SVR uses all unigrams, their sentiment scores, POS tags, and word embeddings as features during the training. For composition, they use maximal, average or concatenation of the embeddings. They perform learning for trigrams and bigrams separately. The best results reported by RBF-SVR on bigrams and trigrams are $r = 0.802$ and $r = 0.753$, respectively.

As opposed to the experimental scheme of RBF-SVR, we apply our regular training procedure on SCL-OPP lexicon. We consider it important that the learned model generalizes well to phrases of variable length, hence we consider the training of one model per phrase length not conducive. Rather, we argue that training CMSM can and should be done independent of the length of phrases, by ultimately using the combination of different length phrases for training and testing. Also, our approach does not use information extracted from other resources (such as Word2vec) nor POS tagging, i.e., we perform a light-weight training with fewer features. Still, we were able to obtain Pearson $r = 0.759$ in the task of regression.

Table 5 presents the results obtained by random initialization in CMSM and Grad-GMSM on the SCL-OPP dataset. In both methods we apply early stopping and perform ten-fold cross-validation. In Grad-GMSM, the total number of iterations again includes the iterations in both learning phases, and still it shows a faster convergence toward minimum ranking loss.

Method	Ranking loss	Pearson r	Total number of iterations
Grad-GMSM	0.249	0.759	2.5 (Step 1) + 7.7 (Step 2) = 10.2
randIni-GMSM	0.376	0.441	77

Table 5: Performance comparison for different initializations in SCL-OPP

Finally, we repeated the experiments on the Grad-GMSM model with values of m, i.e., different numbers of dimensions. For each dimension number, we took the average of 5 runs. As shown in Table 6, the results do improve only marginally when increasing m over several orders of magnitude. Also the number of required iterations remains essentially the same, except for $m = 1$, which does not exploit the matrix properties. We see that – as opposed to vector space models – good performance can be achieved already with a very low number of dimensions.

Number of dimensions	Ranking loss	Pearson r	Total number of iterations
1	0.441	0.487	20.3
2	0.270	0.728	12.8
3	0.269	0.731	10.6
10	0.266	0.736	12.3
20	0.263	0.741	11.1
50	0.258	0.748	10.6
100	0.253	0.754	9.6
300	0.245	0.763	9.6

Table 6: Performance comparison for different dimensions in SCL-OPP

5 Conclusion

In this paper, we addressed the problem of learning compositional matrix-space models for the task of sentiment analysis. As opposed to the standard gradient descent approach, the novelty of our approach consists in a two-step learning procedure, where the result of the first step is used as initialization for the second step. We showed that with this alternative initialization step added to the learning process, we get lower ranking loss than (1) a previously described learning method on CMSM and (2) the standard gradient descent method starting from a random initialization. Moreover, we evaluated the performance of training CMSMs in sentiment prediction of phrases with opposing polarities and observed that the model captures compositionality well in such phrases. Since CMSMs turn out to be very robust against the choice of dimensionality, we conclude that choosing low-dimensional matrices as word representations leads to a reduced training time and still very good performance. In the future, we plan to extensively compare the learning of word matrix representations with vector space models in the task of sentiment analysis on several datasets.

References

Ameeta Agrawal and Aijun An. 2014. Kea: Sentiment analysis of phrases within short texts. In Preslav Nakov and Torsten Zesch, editors, *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. pages 380–384.

Gottlob Frege. 1884. *Die Grundlagen der Arithmetik: eine logisch-mathematische Untersuchung über den Begriff der Zahl*. Breslau, Germany: W. Koebner.

James Hong and Michael Fang. 2015. Sentiment analysis with deeply learned distributed representations of variable length texts. Technical report, Stanford University.

Svetlana Kiritchenko and Saif M Mohammad. 2016a. The effect of negators, modals, and degree adverbs on sentiment composition. In Alexandra Balahur, Erik van der Goot, Piek Vossen, and Andrés Montoyo, editors, *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*. pages 43–52.

Svetlana Kiritchenko and Saif M Mohammad. 2016b. Sentiment composition of words with opposing polarities. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*. pages 1102–1108.

Jean Maillard and Stephen Clark. 2015. Learning adjective meanings with a tensor-based skip-gram model. In Afra Alishahi and Alessandro Moschitti, editors, *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL 2015)*. pages 327–331.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In Chris J.C. Burges, Léon Bottou, Max Welling, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in neural information processing systems (NIPS 2013)*. pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science* 34(8):1388–1429.

Kaare B. Petersen and Michael S. Pedersen. 2012. *The Matrix Cookbook*. Technical University of Denmark. Version 20121115.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In Jan Hajic, Sandra Carberry, and Stephen Clark, editors, *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*. pages 907–916.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In Jun'ichi Tsujii, James Henderson, and Marius Pasca, editors, *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*. pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP 2013)*. pages 1631–1642.

Aditya Timmaraju and Vikesh Khanna. 2015. Sentiment analysis on movie reviews using recursive and recurrent neural network architectures. *Semantic Scholar*.

Xingyou Wang, Weijie Jiang, and Zhiyong Luo. 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In Yuji Matsumoto and Rashmi Prasad, editors, *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*. pages 2428–2437.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*. pages 172–182.