Compositional Matrix-Space Models for Sentiment Analysis

Ainur Yessenalina
Dept. of Computer Science
Cornell University
Ithaca, NY, 14853
ainur@cs.cornell.edu

Claire Cardie
Dept. of Computer Science
Cornell University
Ithaca, NY, 14853
cardie@cs.cornell.edu

Abstract

We present a general learning-based approach for phrase-level sentiment analysis that adopts an ordinal sentiment scale and is explicitly compositional in nature. Thus, we can model the compositional effects required for accurate assignment of phrase-level sentiment. For example, combining an adverb (e.g., "very") with a positive polar adjective (e.g., "good") produces a phrase ("very good") with increased polarity over the adjective alone. Inspired by recent work on distributional approaches to compositionality, we model each word as a matrix and combine words using iterated matrix multiplication, which allows for the modeling of both additive and multiplicative semantic effects. Although the multiplication-based matrix-space framework has been shown to be a theoretically elegant way to model composition (Rudolph and Giesbrecht, 2010), training such models has to be done carefully: the optimization is non-convex and requires a good initial starting point. This paper presents the first such algorithm for *learning* a matrix-space model for semantic composition. In the context of the phrase-level sentiment analysis task, our experimental results show statistically significant improvements in performance over a bag-of-words model.

1 Introduction

Sentiment analysis has been an active research area in recent years. Work in the area ranges from identifying the sentiment of individual words to determining the sentiment of phrases, sentences and documents (see Pang and Lee (2008) for a survey). The bulk of previous research, however, models just positive vs. negative sentiment, collapsing positive (or negative) words, phrases and documents of differing intensities into just one positive (or negative) class. For word-level sentiment, therefore, these methods would not recognize a difference in sentiment between words like "good" and "great", which have the same direction of polarity (i.e., positive) but different intensities. At the phrase level, the methods will fail to register compositional effects in sentiment brought about by intensifiers like "very", "absolutely", "extremely", etc. "Happy" and "very happy", for example, will both be considered simply "positive" in sentiment. In real-world settings, on the other hand, sentiment values extend across a polarity spectrum — from very negative, to neutral, to very positive. Recent research has shown, in particular, that modeling intensity at the phrase level is important for real-world natural language processing tasks including question answering and textual entailment (de Marneffe et al., 2010).

This paper describes a general approach for phrase-level sentiment analysis that takes these real-world requirements into account: *we adopt a five-level ordinal sentiment scale and present a learning-based method that assigns ordinal sentiment scores to phrases*.

Importantly, our approach will also be explicitly *compositional*[1] in nature so that it can accurately account for critical interactions among the words in

[1]The *Principle of Compositionality* asserts that the meaning of a complex expression is a function of the meanings of its constituent expressions and the rules used to combine them.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 172–182,
Edinburgh, Scotland, UK, July 27–31, 2011.

each sentiment-bearing phrase. Consider, for example, combining an adverb like "very" with a polar adjective like "good". "Good" has an *a priori* positive sentiment, so "very good" should be considered **more** positive even though "very", on its own, does not bear sentiment. Combining "very" with a negative adjective, like "bad", produces a phrase ("very bad") that should be characterized as more negative than the original adjective. Thus, it is convenient to think of the effect of combining an intensifying adverb with a polar adjective as being *multiplicative* in nature, if we assume the adjectives ("good" and "bad") to have positive and a negative sentiment scores, respectively.

Next, let us consider adverbial negators like "not" combined with polar adjectives. When modeling only positive and negative labels for sentiment, negators are generally treated as flipping the polarity of the adjective it modifies (Choi and Cardie, 2008; Nakagawa et al., 2010). However, recent work (Taboada et al., 2011; Liu and Seneff, 2009) suggests that the effect of the negator when ordinal sentiment scores are employed is more akin to dampening the adjective's polarity rather than flipping it. For example, if "perfect" has a strong positive sentiment, then the phrase "not perfect" is still positive, though to a lesser degree. And while "not terrible" is still negative, it is less negative than "terrible". For these cases, it is convenient to view "not" as shifting polarity to the opposite side of polarity scale by some value.

There are, of course, more interesting examples of compositional semantic effects on sentiment: e.g., *prevent cancer*, *ease the burden*. Here, the verbs *prevent* and *ease* act as content-word negators (Choi and Cardie, 2008) in that they modify the negative sentiment of their direct object arguments so that the phrase as a whole is perceived as somewhat positive.

Nonetheless, the vast majority of methods for phrase- and sentence-level sentiment analysis do not tackle the task compositionally: they, instead, employ a bag-of-words representation and, at best, incorporate additional features to account for negators, intensifiers, and for contextual valence shifters, which can change the sentiment over neighboring words (e.g., Polanyi and Zaenen (2004), Wilson et al. (2005) , Kennedy and Inkpen (2006), Shaikh et al. (2007)).

One notable exception is Moilanen and Pulman (2007), who propose a compositional semantic approach to assign a positive or negative sentiment to newspaper article titles. However, their knowledge-based approach presupposes the existence of a sentiment lexicon and a set of symbolic compositional rules.

But learning-based compositional approaches for sentiment analyis also exist. Choi and Cardie (2008), for example, propose an algorithm for phrase-based sentiment analysis that learns proper assignments of intermediate sentiment analysis decision variables given the *a priori* (i.e., out of context) polarity of the words in the phrase and the (correct) phrase-level polarity. As in Moilianen and Pulman (2007), semantic inference is based on (a small set of) hand-written compositional rules. In contrast, Nakagawa et. al (2010) use a dependency parse tree to guide the learning of compositional effects. Each of the above, however, uses a binary rather than an ordinal sentiment scale.

In contrast, our proposed method for phrase-level sentiment analysis is inspired by recent work on distributional approaches to compositionality. In particular, Baroni and Zamparelli (2010) tackle adjective-noun compositions using a vector representation for nouns and *learning* a matrix representation for each adjective. The adjective matrices are then applied as functions over the meanings of nouns — via matrix-vector multiplication — to derive the meaning of adjective-noun combinations. Rudolph and Giesbrecht (2010) show theoretically, that multiplicative matrix-space models are a general case of vector-space models and furthermore exhibit desirable properties for semantic analysis: they take into account word order and are algebraically, neurologically and psychologically plausible. This work, however, does not present an algorithm for learning such models; nor does it provide empirical evidence in favor of matrix-space models over vector-space models.

In the sections below, we propose a *learning-based* approach to assign *ordinal sentiment scores* to sentiment-bearing phrases using a general *compositional matrix-space model of language*. In contrast to previous work, all words are modeled as matrices, independent of their part-of-speech, and compositional inference is uniformly modeled as ma-

trix multiplication. To predict an ordinal scale sentiment value, we employ Ordered Logistic Regression, introducing a novel training algorithm to accommodate our compositional matrix-space representations (Section 2). To our knowledge, this is the first such algorithm for learning matrix-space models for semantic composition. We evaluate the approach on a standard sentiment corpus (Wiebe et al., 2005) (Section 3), making use of its manually annotated phrase-level annotations for polarity and intensity, and compare our approach to the more commonly employed bag-of-words model. We show (Section 4) that our matrix-space model significantly outperforms a bag-of-words model for the ordinal scale sentiment prediction task.

2 The Model for Ordinal Scale Sentiment Prediction

As described above, our task is to predict an ordinal scale sentiment value for a phrase. To this end, we employ a sentiment scale with five ordinal values: VERY NEGATIVE, NEGATIVE, NEUTRAL, POSITIVE and VERY POSITIVE. Given a set of phrase-level training examples with their gold-standard ordinal sentiment value, we then use an Ordered Logistic Regression (OLogReg) model for prediction. Unfortunately, our matrix-space representation precludes doing this directly.

We have chosen OLogReg, as opposed to say PRanking (Crammer and Singer, 2001), because optimization of the former is more attractive: the objective (likelihood) is smooth and the gradients are continuous. As will become clear shortly, learning our models is not trivial and it is important to use sophisticated off-the-shelf optimizers such as L-BFGS.

For a bag-of-words model, OLogReg learns one weight for each word and a set of thresholds by maximizing the likelihood of the training data. Typically, this is accomplished by using an optimizer like L-BFGS whose interface needs the value and gradient of the likelihood with respect to the parameters at their current values. In the next subsections, we instantiate OLogReg for our sentiment prediction task using a matrix-space word model (2.1 and 2.2) and a bag-of-words model (2.3). The learning formulation of bag-of-words OLogReg is convex therefore we will get the global optimum; in contrast, the optimization problem for matrix-space model is non-convex, it is important to initialize the model well. Initialization of the matrix-space model is discussed in Section 2.4.

2.1 Notation

In the subsequent subsections we will use the following notation. Let n be the number of phrases in the training set and let d be the number of words in the dictionary. Let x^i be the i-th phrase and y^i would be the label of x^i, where y^i takes r different values $y^i \in \{0, \ldots, r-1\}$. Then $|x^i|$ will denote the length of the phrase x^i, and the words in i-th phrase are: $x^i = x^i_1, x^i_2, \ldots, x^i_{|x^i|}$; $x^i_j, 1 \leq j \leq |x^i|$ is the j-th word of i-th phrase; where x^i_j is from the dictionary: $1 \leq x^i_j \leq d$.

In the case of the bag-of-words model, $\Phi(x^i) \in \mathbb{R}^d$ is the representation of the i-th phrase. $\Phi_j(x^i)$ counts the number of times the j-th word from the dictionary appears in the i-th phrase. Given a $w \in \mathbb{R}^d$ it assigns a score ξ_i to a phrase x^i by

$$\xi_i = w^T \Phi(x^i) = \sum_{j=1}^{|x^i|} w_{x^i_j} \tag{1}$$

In the case of the matrix-space model the $\Phi(x^i) \in \mathbb{R}^{|x^i| \times d}$ is the representation of the i-th phrase. $\Phi_{jk}(x^i)$ is 1, if x^i_j is the k-th word in the dictionary, and zero otherwise. Given $u, v \in \mathbb{R}^m$ and a set of matrices $\{W_p \in \mathbb{R}^{m \times m}\}_{p=1}^d$, one for each word, it assigns a score ξ_i to a phrase x^i by

$$\begin{aligned} \xi_i &= u^T \left(\prod_{j=1}^{|x^i|} \sum_{k=1}^{d} W_k \Phi_{jk}(x^i) \right) v \\ &= u^T \left(\prod_{j=1}^{|x^i|} W_{x^i_j} \right) v \end{aligned} \tag{2}$$

where $\prod_{j=1}^{|x^i|} W_{x^i_j} = W_{x^i_1} W_{x^i_2} \cdots W_{x^i_{|x^i|}}$ in **exactly this order**. We choose to map matrices to the real numbers by using vectors u and v from $\mathbb{R}^{m \times 1}$; so that $\xi = u^T M v$, where $M \in \mathbb{R}^{m \times m}$, which is sensitive to the order of matrices[2] , i.e. $u^T M_1 M_2 v \neq$

[2]Care must be taken in choosing way to map matrix to a real

$u^T M_2 M_1 v$.

Modeling composition. A $m \times m$ matrix, representing a word, can be considered as a linear function, mapping from $\mathbb{R}^m$ to $\mathbb{R}^m$. Composition of words is modeled by function composition, in our case composition of linear functions, i.e. matrix multiplication. Note, that unlike bag-of-words model, the matrix-space model takes word order into account, since matrix multiplication is not commutative operation.

2.2 Ordered Logistic Regression

Now we will describe our objective function for OLogReg and its derivatives. OLogReg has $r-1$ thresholds $(\kappa_0, \dots \kappa_{r-2})$, so introducing $\kappa_{-1} = -\infty$ and $\kappa_{r-1} = \infty$ leads to the unified expression for posterior probabilities for all values of k:

$$\begin{aligned} P(y^i = k|x) &= P(\kappa_{k-1} < \xi_i \leq \kappa_k) \\ &= F(\kappa_k - \xi_i) - F(\kappa_{k-1} - \xi_i) \end{aligned}$$

$F(x)$ is an inverse-logit function

$$F(x) = \frac{e^x}{1+e^x}$$

this is its derivative:

$$\frac{dF(x)}{dx} = F(x)(1 - F(x))$$

Therefore the negative loglikelihood of the training data will look like the following (Hardin and Hilbe, 2007):

$$L = -\sum_{i=1}^{n}\sum_{k=0}^{r-1} \ln(F(\kappa_k - \xi_i) - F(\kappa_{k-1} - \xi_i))I(y^i = k)$$

where r is the number of ordinal classes, ξ_i is the score of i-th phrase, I is the indicator function that is equal to 1 – when $y^i = k$, and zero otherwise. We need to minimize the objective L with respect to the following constraints:

$$\kappa_{k-1} \leq \kappa_k, \;\; 1 \leq k \leq r-2 \tag{3}$$

number. For example, one other way to map matrices to the real numbers is to use the determinant of a matrix; however, the determinant is not sensitive to the word order: $det(M_1 M_2) = det(M_1)det(M_2) = det(M_2 M_1)$; which is not desirable for a model that needs to account for word order.

(The constraints are similar to the ones in PRank algorithm). For ease of optimization we parametrize our model via κ_0, and $\tau_j, 1 \leq j \leq r-2$:

$\kappa_{-1} = -\infty$,
κ_0,
$\kappa_1 = \kappa_0 + \tau_1$,
$\kappa_2 = \kappa_0 + \sum_{j=1}^{2} \tau_j$,
$\dots$,
$\kappa_{r-2} = \kappa_0 + \sum_{j=1}^{r-2} \tau_j$
$\kappa_{r-1} = \infty$,

where $\tau_1, \dots, \tau_{r-2}$ are non-negative values, that represent how far the corresponding thresholds are from each other. Then the constraints (3) would be:

$$\tau_j \geq 0, \;\; 1 \leq j \leq r-2 \tag{4}$$

To simplify the equations we can rewrite the negative loglikelihood as follows:

$$L = -\sum_{i=1}^{n}\sum_{k=0}^{r-1} \ln(A_{ik} - B_{ik})I(y^i = k) \tag{5}$$

where

$$A_{ik} = \begin{cases} F(\kappa_0 + \sum_{j=1}^{k} \tau_j - \xi_i), & \text{if } k = 0, \dots, r-2 \\ 1, & \text{if } k = r-1 \end{cases}$$

$$B_{ik} = \begin{cases} 0, & \text{if } k = 0 \\ F(\kappa_0 + \sum_{j=1}^{k-1} \tau_j - \xi_i), & \text{if } k = 1, \dots, r-1 \end{cases}$$

Let's introduce $L_{ik} = -\ln(A_{ik} - B_{ik})I(y^i = k)$ and then the derivative of L_{ik} with respect to κ_0 will be:

$$\begin{aligned} \frac{\partial L_{ik}}{\partial \kappa_0} &= \frac{-[A_{ik}(1-A_{ik}) - B_{ik}(1-B_{ik})]}{A_{ik} - B_{ik}} I(y^i = k) \\ &= (A_{ik} + B_{ik} - 1)I(y^i = k) \end{aligned}$$

For $j = y^i$:

$$\frac{\partial L_{ik}}{\partial \tau_j} = \frac{-A_{ik}(1-A_{ik})}{A_{ik} - B_{ik}} I(y^i = k)$$

For all $j < y^i$:

$$\frac{\partial L_{ik}}{\partial \tau_j} = (A_{ik} + B_{ik} - 1)I(y^i = k)$$

For all $j > y^i$: $\frac{\partial L_{ik}}{\partial \tau_j} = 0$.

The derivative with respect to the score ξ_i is:

$$\frac{\partial L_{ik}}{\partial \xi_i} = (-A_{ik} - B_{ik} + 1)I(y^i = k) \tag{6}$$

2.2.1 Matrix-Space Word Model

Here we show the derivatives with respect to a word. For the OLogReg model with matrix-space word representations, we have:

$$\frac{\partial L}{\partial W_{x_j^i}} = \frac{\partial L}{\partial \xi_i} \cdot \frac{\partial \xi_i}{\partial W_{x_j^i}}$$

The expression for $\frac{\partial L}{\partial \xi_i}$ is given in (6); we will derive $\frac{\partial \xi_i}{\partial W_{x_j^i}}$ from (2). In the case of the Matrix-Space word model each word is represented as an $m \times m$ affine matrix W:

$$W = \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} \tag{7}$$

We choose the class of **affine matrices** since for affine matrices matrix multiplication represents both operations: linear transformation and translation. Linear transformation is important for modeling changes in sentiment - translation is also useful (we make use of a translation vector during initialization, see Section 2.4). In this work we consider $m \geq 3$ since we want the matrix A from (7) to represent rotation and scaling. Applying the affine transformation W to vector $[x, 1]^T$ is equivalent to applying linear transformation A and translation b to x. [3]

Though vectors u and v can be learned together with word matrices W_j, **we choose to fix u and v.** The main intuition behind fixing u and v is *to reduce the degrees of freedom of the model*: different assignments of u, v and W_j-s can lead to the same score ξ, i.e. there exist $\hat{u}$ $\hat{v}$ and $\hat{W}_j$-s different from u, v and W_j-s respectively, such that $\xi(u, v, W)$ would be equal to $\xi(\hat{u}, \hat{v}, \hat{W})$. [4]

[3]
$$\begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} = \begin{pmatrix} Ax + b \\ 1 \end{pmatrix}$$
where A is a linear transformation, b is a translation vector. Also the product of affine matrices is an affine matrix.

[4] The specific choice of u and v leads to an equivalent model for all $\hat{u}$ and $\hat{v}$ such that $\hat{u} = M^T u$, $\hat{v} = M^{-1} v$, where M is any invertible transformation (i.e. $\hat{u}, \hat{v}$ are derived from u,v by applying linear transformations M^T, M^{-1} respectively):

$$\begin{aligned} u^T W_1 W_2 v &= (u^T M)(M^{-1} W_1 M)(M^{-1} W_2 M)(M^{-1} v) \\ &= \hat{u}^T \hat{W}_1 \hat{W}_2 \hat{v} \end{aligned}$$

The derivative of the phrase ξ_i with respect to j-th word W_j would be (for brevity we drop the phrase index and W_j refers to $W_{x_j^i}$ and p refers to $|x_i|$):

$$\begin{aligned} \frac{\partial \xi_i}{\partial W_j} &= \left(\frac{\partial u^T W_1 W_2 \dots W_p v}{\partial W_j} \right) \\ &= \left[(u^T W_1 \dots W_{j-1})^T (W_{j+1} \dots W_p v)^T \right] \\ &= \left[(W_{j-1}^T \dots W_1^T)(u v^T)(W_p^T \dots W_{j+1}^T) \right] \end{aligned}$$

(see Peterson and Pederson(2008)).
In case if a certain word appears multiple times in the phrase, the derivative with respect to that word would be a sum of derivatives with respect to each appearance of a word, while all other appearances are fixed. For example,

$$\left(\frac{\partial u^T W W_1 W v}{\partial W} \right) = u(W_1 W v)^T + (u^T W W_1)^T v^T$$

where W is a representation of a word that is repeated.

So given the expression (6) for $\frac{\partial L}{\partial \xi_i}$, the derivative with respect to each word can be computed. Notice that the update for the j-th word in a sentence depends on the order words, which is in line with our desire to account for word order.

2.2.2 Optimization

The goal of training procedure is for the i-th phrase with p words $x_1 x_2 \dots x_p$ to learn word matrices W_1, W_2, ..., W_p such that resulting ξ_i-s will lead to the lowest negative loglikelihood. The goal of training procedure is to find word matrices W_1, W_2, ... W_p and thresholds κ_0, τ_1, ... τ_{r-2} such that the negative loglikelihood is minimized. So, given the negative loglikelihood and the derivatives with respect κ_0 and τ_j-s and word matrices W, we optimize objective (5) subject to $\tau_j \geq 0$. We use L-BFGS-B (Large-scale Bound-constrained Optimization) by Byrd et al. (1995) as an optimizer.

2.2.3 Regularization in Matrix-Space Model

In order to make sure that the L-BFGS-B updates do not cause numerical issues we perform the following regularization to the resulting matrices. An m by m matrix W_j that can be represented as:

$$W_j = \begin{pmatrix} A_{11} & a_{12} \\ a_{21}^T & a_{22} \end{pmatrix}$$

where $A_{11} \in \mathbb{R}^{m-1 \times m-1}$, $a_{12}, a_{21} \in \mathbb{R}^{m-1 \times 1}$, $a_{22} \in \mathbb{R}$. First make the matrix affine by updating the last row, then the updated matrix will look like:

$$\hat{W}_j = \begin{pmatrix} A_{11} & a_{12} \\ 0 & 1 \end{pmatrix}$$

It can be proven that such a projection returns the closest affine matrix in Frobenius norm.

However, we also want to regularize the model to avoid ill-conditioned matrices. Ill-conditioned matrices represent transformations whose output is very sensitive to small changes in the input and therefore they have a similar effect to having large weights in a bag-of-words model. To perform such a regularization we "shrink" the singular values of A_{11} towards one. More specifically, we first use the Singular Value Decomposition (SVD) of the A_{11}: $U\Sigma V^T = A_{11}$, where U and V are orthogonal matrices, Σ is a matrix with singular values on the diagonal. Then we update singular values in the following way to get $\tilde{\Sigma}$: $\tilde{\Sigma}_{ii} = \Sigma_{ii}^h$, where h is a parameter between 0 and 1. If $h = 1$ then Σ_{ii} remains the same. In the extreme case $h = 0$ then $\Sigma_{ii}^h = 1$. For intermediate values of h the singular values of A_{11} would be brought closer to one. Finally, we recompute $\tilde{A}_{11}$: $\tilde{A}_{11} = U\tilde{\Sigma}V^T$. So, W_j would be :

$$\tilde{W}_j = \begin{pmatrix} \tilde{A}_{11} & a_{12} \\ 0 & 1 \end{pmatrix}$$

2.2.4 Learning in the Matrix-Space Model

We use Algorithm 1 to learn the matrix-space model. What essentially happens is that we iterate two steps: optimizing the W matrices using L-BFGS-B and the projection step. L-BFGS-B returns a solution that is not necessarily an affine matrix. After projecting to the space of affine matrices we start L-BFGS-B from a better initial point. In practice, the first few iterations lead to large decrease in negative loglikelihood.

2.3 Bag-Of-Words Model

In the bag-of-words model the score of the i-th phrase is given in (1). Therefore, the partial derivative with respect to j-th word in i-th phrase $\frac{\partial \xi_i}{\partial w_{x_j^i}}$ is equal to the number c_j of times x_i^j appears in x^i, so:

$$\frac{\partial L}{\partial w_{x_j^i}} = \frac{\partial L}{\partial \xi_i} \cdot c_j$$

Algorithm 1 Training Algorithm for Matrix-Space OLogReg

```
Input: {(x_1, y_1), ..., (x_n, y_n)} //training data
Input: h //projection parameter
Input: T //number of iterations
Input: W, κ_0 and τ_j //initial values
for t = 1, ..., T do
    (W, κ_0, τ_j)=minimize L using L-BFGS-B
    for i = 1, ..., d do
        W_i=Project(W_i, h)
    end for
end for
Return W, κ_0, τ_j
```

Optimization. We minimize negative loglikelihood using L-BFGS-B subject to $\tau_j \geq 0$.

Regularization. To prevent overfitting for bag-of-words model we regularize w. The L_2-regularized negative loglikelihood will consist of the expression in (5) and an additional term $\frac{\lambda}{2}||w||_2^2$, where $||\cdot||_2$ is the L_2-norm of a vector. The derivative of the additional term with respect to w will be:

$$\frac{\partial \frac{\lambda}{2}||w||_2^2}{\partial w} = \lambda w$$

Hence the partial derivative with respect to $w_{x_j^i}$ will have an additional term $\lambda w_{x_j^i}$.

2.4 Initialization

Initialization of bag-of-words OLogReg. We initialize the weight for each word with zero and κ_0 with a random number and τ_j-s with non-negative random numbers. Since the learning problem for bag-of-words OLogReg is convex, we will get the global optimum.

Better Initialization of Matrix-Space Model. Preliminary experiments showed that the Matrix-Space model needs a good initialization. Initializing with different random matrices reaches different local minima and the quality of local minima depends on initialization. Therefore, it is important to initialize the model with a good initial point. *One way to initialize the Matrix-Space model is to use the weights learned by the bag-of-words model.* We use the following intuition for initializing the Matrix-Space model. As noted in Section 2.2.1 applying transformation A of affine matrix W can model a linear

transformation, while vector b represents a translation. Since matrix-space model can encode a vector-space model (Rudolph and Giesbrecht, 2010), we can initialize the matrices to exactly mimic the bag-of-words model. In order to do that we place the weight, learned by the bag-of-words model in the first component of b. Let's assume that w_{x_1} and w_{x_2} are the weights learned for two distinct words x_1 and x_2 respectively. To compute the polarity score of a phrase x_1, x_2 the bag-of-words model sums the weights of these two words: w_{x_1} and w_{x_2}. Now we want to have the same effect in matrix-space model. Here we assume $m = 3$.

$$Z = \begin{pmatrix} 1 & 0 & w_{x_1} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & w_{x_2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & w_{x_1} + w_{x_2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Finally, there is a step of mapping matrix Z to a number using u and v, such that $\xi(Z) = w_{x_1} + w_{x_2}$.We also want vector u and v to be such that:

$$u^T \begin{pmatrix} 1 & 0 & w_{x_1} + w_{x_2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} v = w_{x_1} + w_{x_2} \quad (8)$$

The last equation can help us construct u and v. We also set u and v to be orthogonal: $u^T v = 0$. So, we arbitrarily choose two orthogonal vectors for which equation (8) holds: $u = [1, \sqrt{2}, 1]^T$ and $v = [1, -\sqrt{2}, 1]^T$.[5]

3 Experimental Methodology

For experimental evaluation of the proposed method we use the publicly available Multi-Perspective Question Answering (MPQA)[6] corpus (Wiebe et al., 2005) version 1.2, which contains 535 newswire documents that are manually annotated with phrase-level subjectivity and intensity. We use the expression-level boundary markings in MPQA to extract phrases. We evaluate on positive, negative and neutral opinion expressions that have intensities "medium", "high" or "extreme".[7] The schematic mapping of phrase polarity and intensity values on ordinal sentimental scale is shown in Table 1.

Polarity	Intensity	Ordinal label
negative	high, extreme	0
negative	medium	1
neutral	high, extreme, medium	2
positive	medium	3
positive	high, extreme	4

Table 1: Mapping of combination of polarities and intensities from MPQA dataset to our ordinal sentiment scale.

3.1 Training Details

We perform 10-fold cross-validation on phrases extracted from the MPQA corpus: eight folds for training; one as a validation set; and one as test set. In total there were 8022 phrases. Before training, we extract lemmas for each word. For evaluation we use Ranking Loss: $\frac{1}{n}\sum_i |\hat{y}^i - y^i|$, where $\hat{y^i}$ is the prediction.

Choice of dimensionality m. The reported experiments are done by setting $m = 3$. Preliminary experiments with higher values of m (5, 20, 50), did not lead to a better performance and increased the training time; therefore we did not use those values in our final experiments.

3.2 Methods

PRank. For each of the folds, we run 500 iterations of PRank and choose an early stopping iteration using a model that led to the lowest ranking loss on the validation set; afterwards report the average performance of on a test set.

Bag-of-words OLogReg. To prevent overfitting we search for the best regularization parameter among the following values of λ: 10^i, from 10^{-4} to 10^4. The lowest negative log-likelihood value on the validation set is attained for[8] $\lambda = 0.1$. With this value of λ fixed, the final model is the one with the lowest negative loglikelihood on the training set.

[5] If $m > 3$, u and v can be set using the same intuition.

[6] http://www.cs.pitt.edu/mpqa/

[7] We ignored low-intensity phrases similar to (Choi and Cardie, 2008; Nakagawa et al., 2010).

[8] We pick single λ that gives best average validation set performance, and then use it to compute the average test set performance.

Method	Ranking loss
PRank	0.7808
Bag-of-words OLogReg	0.6665
Matrix-space OLogReg+RandInit	0.7417
Matrix-space OLogReg+BowInit	0.6375†

Table 2: Ranking loss for vector-space Ordered Logistic Regression and Matrix-Space Logistic Regression.
† Stands for a significant difference w.r.t. the Bag-Of-Words OLogReg model with p-value less than 0.001 ($p < 0.001$)

Matrix-space OLogReg+RandInit. First, we initialized matrices with with random numbers from normal distribution $N(0, 0.1)$ and set u and v as in section 2.4, T is set to 25. We run with two different random seeds and three different values for the parameter h: [0.1, 0.5, 0.9] and report the performance of the model that had the lowest likelihood on the validation set. The setting of h that lead to the best model was 0.9.

Matrix-space OLogReg+BowInit. For the matrix-space models we initialize the model with the output of the regularized Bag-of-words OLogReg as described in Section 2.4, T is set to 25. Then we use the training procedure of Algorithm 1. We consider three different values for the parameter h [0.1, 0.5, 0.9] and choose as the model with the lowest validation set negative log-likelihood. The best setting of h was 0.1.

4 Results and Discussion

We report Ranking Loss for the four models in Table 2. The worst performance (denoted by the highest ranking loss value) is obtained by PRank, followed by matrix-space OLogReg with random initialization. Bag-of-words OLogReg obtains quite good performance, and matrix-space OLogReg, initialized using the bag-of-words model performs the best, showing statistically significant improvements over the bag-of-words OLogReg model according to a paired t-test. .

To see what the bag-of-word and matrix-space models are learning we performed inference on a few examples. In Table 3 we show the sentiment scores of the best performing bag-of-words **OLogReg** model and the best performing model based on matrices **Matrix-space OLogReg+BowInit**. By sentiment score, we mean equation (1) of Bag-of-words OLogReg and equation (2) of Matrix-space OLogReg+BowInit.

Phrase	Matrix-space OLogReg+BowInit	Bag-of-words OLogReg
not	-0.83	-0.42
very	0.23	0.04
good	2.81	1.51
very good	3.53	1.55
not good	-0.16	1.09
not very good	0.66	1.13
bad	-1.67	-1.42
very bad	-2.01	-1.38
not bad	-0.54	-1.85
not very bad	-1.36	-1.80

Table 3: Phrase and the sentiment scores of the phrase for 2 models Matrix-space OLogReg+BowInit and Bag-of-words OLogReg respectively. Notice that **relative ranking order what matters**

Here we choose two popular adjectives like 'good' and 'bad' that appeared in the training data, and examine the effect of applying the intensifier 'very' on the sentiment score. As we can see, the matrix-space model learns a matrix that intensifies both 'bad' and 'good' in the correct sentiment scale, i.e., $\xi(good) < \xi(very\ good)$ and $\xi(bad) < \xi(very\ bad)$, while the bag-of-words model gets the sentiment of 'very bad' wrong: it is more positive than 'bad'. We also looked at the effect of combining 'not' with these adjectives. The matrix-space model correctly encodes the effect of the negator for both positive and negative adjectives, such that $\xi(not\ good) < \xi(good)$ and $\xi(bad) < \xi(not\ bad)$. For the interesting case of applying a negator to a phrase with an intensifier, $\xi(not\ good)$ should be less than $\xi(not\ very\ good)$ and $\xi(not\ very\ bad)$ should be less than $\xi(not\ bad)$.[9] As shown in Table 3, these are predicted correctly by the matrix-space model, which the matrix-space model gets right, but the bag-of-words model misses in the case of "bad".

Also notice that since in the matrix-space model

[9] See the detailed discussion in Taboada et al. (2011) and Liu and Seneff (2009).

each word is represented as a function, more specifically a linear operator, and the function composition defined as matrix multiplication, we can think of "not very" being an operator itself, that is a composition of operator "not" and operator "very".

5 Related Work

Sentiment Analysis. There has been a lot of research in determining the sentiment of words and constructing polarity dictionaries (Hatzivassiloglou and McKeown, 1997; Wiebe, 2000; Rao and Ravichandran, 2009; Mohammad et al., 2009; Velikovich et al., 2010). Some recent work is trying to identify the degree of sentiment of adjectives and adverbs from text using co-occurrence statistics. Work by Taboada et. al (2011) and Liu and Seneff (2009), suggest ways of computing the sentiment of adjectives from data, and computing the effect of combining adjective with adverb as multiplicative effect and combining adjective with negation as additive effect. However these models require the knowledge of a part of speech of given words and the list of negators (since the negator is an adjective as well). In our work we propose a single unified model for handling all words of any part of speech.

On the other hand, there has been some research in trying to model compositional effects for sentiment at the phrase- and sentence-level. Choi and Cardie (2008) hand-code compositional rules in order to model compositional effects of combining different words in the phrase. The hand-coded rules are based on domain knowledge and used to *learn* the effects of combining words in the phrase. Another recent work that tries to model the compositional semantics of combining different words is Nakagawa et. al. (2010), which proposes a model that learns the effects of combining different words using phrase/sentence dependency parse trees and an initial polarity dictionary. They present a learning method that employs hidden variables for sentiment classification: given the polarity of a sentence and the *a priori* polarities of its words, they learn how to model the interactions between words with head-modifier relations in the dependency tree.

Some of the previous work looked at MPQA phrase-level classification. Wilson et al. (2004) tackles the problem of classifying clauses according to their subjective strength but not polarity; Wilson et al. (2005) classifies phrases according to their polarity/sentiment but not strength. Our task is different: we classify phrases according to a single ordinal scale that combines both polarity and strength.

Task of predicting document-level star ratings was considered in (Pang and Lee, 2005; Goldberg and Zhu, 2006). In the current work we look at fine-grained sentiment analysis, more specifically we study word representations for use in true compositional semantic settings.

Distributional Semantics and Compositionality. Research in the area of distributional semantics in NLP and Cognitive Science has looked at different word representations and different ways of combining words. Mitchell and Lapata (2010) propose a framework for vector-based semantic composition. They define composition as an additive or multiplicative function of two vectors and show that compositional approaches generally outperform non-compositional approaches that treat the phrase as the union of single lexical items.

Work by Baroni and Zamparelli (2010) models nouns as vectors in some semantic space and adjectives as matrices. It shows that modeling adjectives as linear transformations and applying those linear transformations to nouns results in final vectors for adjective-noun compositions that are close in semantic space to other similar phrases. The authors argue that modeling adjectives as a linear transformation is a better idea than using additive vector-space models. In this work, a separate matrix for each adjective is *learned* using the Partial Least Squares method in a completely unsupervised way. The recent paper by Rudolph and Giesbrecht (2010), described in the introduction, argues for *multiplicative matrix-space* models. In contrast to other work in this area, our work is concerned with a specific dimension of word meaning — sentiment. Our techniques, however, are quite general and should be applicable to other problems in lexical semantics.

6 Conclusions and Future work

In the current work we present a novel matrix-space model for ordinal scale sentiment prediction and an algorithm for learning such a model. The proposed

model learns a matrix for each word; the composition of words is modeled as iterated matrix multiplication. The matrix-space framework with iterated matrix multiplication defines an elegant framework for modeling composition; it is also quite general. We use the matrix-space framework in the context of sentiment prediction, a domain where interesting compositional effects can be observed. The main focus of this work was to study word representations (represent as a single weight vs. as a matrix) for use in true compositional semantic settings. One of the benefits of the proposed approach is that by learning matrices for words, the model can handle unseen word compositions (e.g. unseen bigrams) when the unigrams involved have been seen.

However, it is not trivial to learn a matrix-space model. Since the final optimization problem is non-convex, the initialization has to be done carefully. Here the weights learned in bag-of-words model come to rescue and provide good initial point for optimization procedure. The final model outperforms the bag-of-words based model, which suggests that this research direction is very promising.

Though in our model the order of composition is the same as the word order, we believe that a linguistically informed order of composition can give us further performance gains. For example, one can use the output of a dependency parser to guide the order of composition, similar to Nakagawa et al. (2010). Another possibility for improvement is to use the information about the scope of negation. In the current work we assume the scope of negation to be the expression following the negation; in reality, however, determining the scope of negation is a complex linguistic phenomenon (Moilanen and Pulman, 2007). So the proposed model can benefit from identifying the scope of negation, similar to (Councill et al., 2010).

Also we plan to consider other ways to initialize the matrix-space model. One interesting direction to explore might be to use non-negative matrix factorization (Lee and Seung, 2001), co-clustering techniques (Dhillon, 2001) to better initialize words that share similar contexts. The other possible direction is to use existing sentiment lexicons and employing a "curriculum learning" strategy (Bengio et al., 2009; Kumar et al., 2010) for our learning problem.

Acknowledgments

This work was supported in part by National Science Foundation Grants BCS-0904822, BCS-0624277, IIS-0968450; and by a gift from Google. We thank the anonymous reviewers, and David Bindel, Nikos Karampatziakis, Lillian Lee and Cornell NLP group for useful suggestions and insightful discussions.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1183–1193, Morristown, NJ, USA. Association for Computational Linguistics.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09. ACM.

R. H. Byrd, P. Lu, and J. Nocedal. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, pages 1190–1208.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Isaac G. Councill, Ryan McDonald, and Leonid Velikovich. 2010. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, NeSp-NLP '10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koby Crammer and Yoram Singer. 2001. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2010. Was it good? It was provocative. learning the meaning of scalar adjectives. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, July 11–16. ACL.

I. S. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*.

Andrew B. Goldberg and Jerry Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*.

James W. Hardin and Joseph Hilbe. 2007. *Generalized Linear Models and Extensions*. Stata Press.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *EACL*, pages 174–181.

Alistair Kennedy and Diana Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2, Special Issue on Sentiment Analysis)):110–125.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems 23*. NIPS.

D. Lee and H. Seung. 2001. Algorithms for non-negative matrix factorization. In *NIPS*.

Jingjing Liu and Stephanie Seneff. 2009. Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 161–169, Singapore, August. Association for Computational Linguistics.

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

Saif Mohammad, Cody Dunne, and Bonnie Dorr. 2009. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 599–608, Singapore, August. Association for Computational Linguistics.

Karo Moilanen and Stephen Pulman. 2007. Sentiment composition. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007)*, pages 378–382, September 27-29.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124.

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

K. B. Petersen and M. S. Pedersen. "2008". *The Matrix Cookbook*. "Technical University of Denmark", "oct". "Version 20081110".

Livia Polanyi and Annie Zaenen. 2004. Contextual lexical valence shifters. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 675–682, Athens, Greece, March. Association for Computational Linguistics.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 907–916, Morristown, NJ, USA. Association for Computational Linguistics.

Mostafa Shaikh, Helmut Prendinger, and Ishizuka Mitsuru. 2007. Assessing sentiment of text by semantic dependency and contextual valence analysis.

Maite Taboada, Julian Brooke, Milan Tofiloskiy, and Kimberly Vollz. 2011). Lexicon-based methods for sentiment analysis. In *Computational Linguistics*.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 777–785, Los Angeles, California, June. Association for Computational Linguistics.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation (formerly Computers and the Humanities)*, 39(2/3):164–210.

Janyce M. Wiebe. 2000. Learning subjective adjectives from corpora. In *In AAAI*, pages 735–740.

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2004. Just how mad are you? In *AAAI*. AAAI.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Empirical Methods in Natural Language Processing (EMNLP)*.