Sentiment Classification Based on Supervised Latent n-gram Analysis

Dmitriy Bespalov
NEC Labs America
CS Dept, Drexel University
db59@drexel.edu

Bing Bai
NEC Labs America
Princeton, NJ, USA
bbai@nec-labs.com

Yanjun Qi
NEC Labs America
Princeton, NJ, USA
yanjun@nec-labs.com

Ali Shokoufandeh
Computer Science Dept.
Drexel University
Philadelphia, PA, USA
as79@drexel.edu

ABSTRACT

In this paper, we propose an efficient embedding for modeling higher-order (n-gram) phrases that projects the n-grams to low-dimensional latent semantic space, where a classification function can be defined. We utilize a deep neural network to build a unified discriminative framework that allows for estimating the parameters of the latent space as well as the classification function with a bias for the target classification task at hand. We apply the framework to large-scale sentimental classification task. We present comparative evaluation of the proposed method on two (large) benchmark data sets for online product reviews. The proposed method achieves superior performance in comparison to the state of the art.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms

Algorithms, Experimentation

Keywords

sentiment analysis, supervised embedding, deep learning

1. INTRODUCTION

Sentiment analysis (SA) or polarity mining refers to identifying and extracting subjective information from natural language text. The problem of automatic sentiment analysis has received significant attention in recent years, largely due to the explosion of online social-oriented content (e.g., user reviews, blogs, etc). As one of its main applications, sentiment classification targets to rate the polarity of a given text accurately towards a label or a score, predicting whether the expressed opinion in the text is positive, negative, or neutral.

In this paper, we will explore the use of high order n-grams for classifying the sentiment orientation of a given text at article level. The motivation is that longer phrases tend to be less ambiguous in terms of their polarity. The authors in [10] pointed out that a discriminating classifier combined with high order n-grams as features can achieve comparable, or better SA performance than state of the art on large-scale data sets. Indeed, e.g., while term "good" is likely a positive sentiment, "not good" or "not very good" are less likely to appear in positive comments. Needless to say, models such as bag-of-unigrams or bag-of-bigrams will possibly fail to deal with "not good" and "not very good", respectively. The other facet of SA is the degree of positivity or negativity of long phrases. As one such example, consider the compounding effect of the phrase "... terrible terrible terrible ..." in contrast to three dispersed "terrible" in a text. The use of n-gram features can provide a remedy for such scenarios [10]. Unfortunately, this comes at a very high computational cost associated with modeling n-grams for $n \geq 3$. This is due to the extremely large parameter space associated to n-grams. For instance, assuming English word dictionary $\mathcal{D}$ of size $|\mathcal{D}|$, then bigram representation of text relates to $|\mathcal{D}|^2$ free parameters, while trigram relates to $|\mathcal{D}|^3$ free parameters. This will become intimidating even for a dictionary of moderate size. When the number of training samples is limited, it can easily lead to overfitting. If unigram dictionary has the size $|\mathcal{D}| = 10,000$, we have $|\mathcal{D}|^2 = 10^8$ free parameters or $|\mathcal{D}|^3 = 10^{12}$ that need to be estimated, which is far too many for a small corpora. As more and more web-scale sentimental data sets become available, large corpora with sentimental labels could easily be accessible for researchers. The availability of large datasets also encourages us to explore n-grams. To the best of our knowledge, there are only a few experimental evaluations involving large-scale data sets [12, 8].

This paper partly motivated our idea and we believe that potentially, using higher order n-grams is beneficial in capturing sentiments in the text. For example, term *good* commonly appears in positive reviews, but "not good" or "not very good" are less likely to happen in positive comments. If using bag-of-unigram, "not" is separated from "good", which does not have the ability to describe the "not good" combination. Similarly using just bag-of-bigrams,

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

the model can not represent the short pattern "not very good" either. Another example is: if a product review uses the phrase "Terrible Terrible Terrible", it contains more negative opinion than three "Terrible" separately occur in the text. Building n-gram features is a method to remedy this issue [10], but it is computationally very difficult to model n-grams (for $n >= 3$) raw features directly.

We utilize a multi-level embedding strategy to project n-grams into a low-dimensional latent semantic space where the projection parameters are trained in a supervised fashion together with the sentiment classification task.

In this paper, we present a novel approach for using high-order n-grams for sentimental classification problem. We will also devise a new embedding mechanism of n-grams, called "latent n-grams", that will enable us to deal with the curse of dimensionality. Next, the proposed embedding of n-grams to low-dimensional latent semantic space will be tied to a classifier, trained for sentiment analysis tasks. Using a deep neural network allows one to learn the parameters for the latent space and the classifier jointly in one unified discriminative framework. It is worth noting that performing dimensionality reduction in the original feature space is a common practice for various classification methods. However, as we discuss in Section 3, methods that bias parameters of the embedding towards specific classification task has not received much attention until recently. The proposed multi-layer embedding and classification model provides two major advantages. First, the latent model greatly reduces the dimensionality and computational efficiency in comparison to raw n-gram features. Moreover, the parameters of latent space are learned using supervised signals arising from the sentiment classification. Two clear advantages to use the proposed method involves that (1), the system utilizes an embedding space to greatly reduce the dimensionality of n-gram, and is thus much easier to model than n-grams raw features. (2), The n-gram embeddings are learned using supervised signals with the main sentiment classification task which means they are optimized for the task and require little human labors in feature engineering.

We evaluate the performance of the proposed method along with several state of art baselines using two typical tasks relevant for SA: (1) binary classification, predicting binary (positive or negative) sentiment of the text; and (2) multi-score sentiment classification, predicting a range of scores on Likert-scale (e.g. 1-5 stars) that reflect both polarity and strength of the sentiment in the text. The presented empirical evidence demonstrate the superior performance of the proposed system on two major publically available benchmark datasets: Amazon[1] and TripAdvisor[2].

The rest of this paper is organized as follows. Section 2 describe our method in details. Section 3 provides an overview of related work. In Section 4 we present the experimental results, and conclude the paper in Section 5.

2. METHOD

Sentiment classification can be stated as the general task of classifying an input text-sequence into certain types or as rating the input text with a certain score. Feature extraction and feature-based representation are critical to the effectiveness of sequence analysis, since text sequences cannot be readily described as feature vectors. Traditional text categorization methods use feature vectors indexed by all possible words (e.g., the so-called "bag-of-words") of a given dictionary to represent text documents. The bag-of-words strategy treats articles as an unordered set of features (words), for which the critical word ordering information is not preserved.

[1] http://times.cs.uiuc.edu/~wang296/Data/TripAdvisor.tar.gz

[2] http://www.cs.jhu.edu/~mdredze/datasets/sentiment/unprocessed.tar.gz

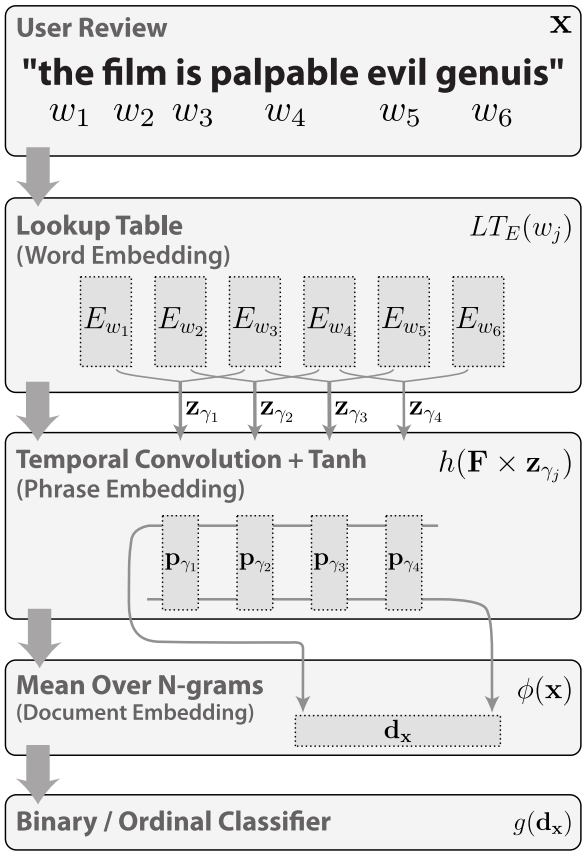

Figure 1: The overall structure of the proposed system.

To take word ordering into account, text articles can be considered as bags of short sequences of words with feature vectors corresponding to all possible word n-grams (n adjacent words from vocabulary $\mathcal{D}$). In the sequel, we will refer to this as "bag-of-ngram" (BON). [20, 10] showed that introducing 2-grams into the bag representation significantly improves the performance of sentiment classification. However, adding higher order n-grams, for $n \geq 3$, into BON model can be prohibitively expensive, since the dimensionality of a BON vector grows exponentially as a function of n.

There are two possible approaches to remedy the curse of dimensionality associated with n-gram representation. First is the feature selection approach. To this ends, pre-selected n-gram patterns or short phrases (e.g. with certain semantic meaning) will be utilized for modeling the sequences. Another possible approach is the use of the dimensionality reduction to approximate the original space – e.g., latent semantic indexing (LSI), an embedding based on term-document matrix.

In this paper, we will use a supervised embedding strategy. We represent a document using its n-gram embedding, which in turn is built upon its word embedding. The step utilizes the supervised signal and acts as feature learning for sentiment classification. Next, the document representation is fed into a perceptron classifier to learn a function mapping toward sentiment labels. The embedding feature learning and the sentiment classification are trained jointly in an end-to-end fashion, which can be illustrated using a multi-layer perceptron (MLP) network structure, as illustrated in Figure 1.

2.1 Embedding of n-Grams

Formally, the basic bag-of-word representation for text applies a mapping $\phi(\cdot)$ to strings of variable length into a feature space of fixed dimension. It is in this space that a standard classifier such as linear perceptron or support vector machine can be applied. Let $\mathcal{D}$ denote the underlying dictionary and $\mathcal{S}$ denote the set of all finite length sequences of words from $\mathcal{D}$. We use $|.|$ to denote the cardinality of a set. We will also assume the aforementioned mapping $\phi : \mathcal{S} \rightarrow \mathbf{R}^M$ maps words sequences in $\mathcal{S}$ to finite dimension feature space. The sentiment labels will form a set $\mathcal{Y} = \{1, ..., K\}$, e.g., $K = 2$ denotes sentiment classes such as "positive" or "negative". We will also denote a labeled training-set with training labels from $\mathcal{Y}$ as $\{\{(\mathbf{x}_i, y_i)\}_{i=1,...,L} | x_i \in \mathcal{S}, y_i \in \mathcal{Y}\}$. An input text sequence of length N will be denoted as $\mathbf{x} = (w_1, \ldots, w_N)$, where $w_j \in \mathcal{D}$. Let Γ denote the vocabulary of n-grams in the corpus, and $\gamma_j = (w_j, w_{j+1}, \ldots, w_{j+n-1})$, where j indicates the j-th position in $\mathbf{x}$. Given the bag-of-unigram representation, $\phi(\mathbf{x})$ maps the input $\mathbf{x}$ in a natural way as a (sparse) vector of dimensionality $M = |\mathcal{D}|$. Similarly, In a bags-of-ngrams $\phi(\mathbf{x})$ maps $\mathbf{x}$ to a $M = |\Gamma|$-dimensional representation, with $|\Gamma| = O(|\mathcal{D}|^n)$.

Motivated by the fact that individual words carry significant semantic information, we learn a mapping of each word into a real-valued vector space, referred to as an "embedding". Specifically, each word $w_j \in \mathcal{D}$ is embedded into an m-dimensional feature space using a *lookup table* $LT_E(\cdot)$ defined as

$$\begin{aligned} LT_E(w_j) &= E \times (0, \ldots, 0, \underset{\text{at index } w_j}{1}, \ldots, 0)^\top \\ &= E_{w_j} = [E_{1,w_j}, E_{2,w_j}, \ldots, E_{m,w_j}]^\top \end{aligned}$$

where $E \in \mathbf{R}^{m \times |\mathcal{D}|}$ is a matrix with word embedding parameters to be learn. Here, $E_{w_j} \in \mathbf{R}^m$ is the embedding of the word w_j in the dictionary $\mathcal{D}$ and m denotes the target word embedding dimensionality. It is important to note that the parameters of E are automatically *trained* during the learning process using backpropagation.

In our framework, formation of n-grams will be carried through a sliding window of length n. As illustrated in Figure 1, setting $n = 3$, the first n-gram is (w_1, w_2, w_3), second n-gram will be (w_2, w_3, w_4), etc. Given an n-gram of adjacent words, we represent its embedding based on the embedding of individual words it contains. Specifically, given $\gamma_j = (w_j, w_{j+1}, \ldots, w_{j+n-1})$, define $\mathbf{z}_{\gamma_j} = [E_{w_j}^\top, E_{w_{j+1}}^\top, \ldots, E_{w_{j+n-1}}^\top]^\top$ as an operator concatenating embeddings of words in the n-gram γ_j, resulting in an nm-dimensional vector $\mathbf{z}_{\gamma_j}$. The embedding of the γ_j can then be defined as

$$\mathbf{p}_{\gamma_j} = h(\mathbf{F} \times \mathbf{z}_{\gamma_j}),$$

where projection matrix $\mathbf{F} \in \mathbb{R}^{M \times nm}$ maps the vector $\mathbf{z}_{\gamma_j}$ into a M-dimensional latent space, and $h(\cdot) = \tanh(\cdot)$[3].

Finally, we use the n-gram embeddings to form a vector representation for each input text document. Formally, the document representation will be defined as

$$\phi(\mathbf{x}) \equiv \mathbf{d}_\mathbf{x} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{p}_{\gamma_j} \tag{1}$$

where $\mathbf{d}_\mathbf{x} \in \mathbb{R}^M$ and $\mathbf{x} = (w_1, \ldots, w_N)$. In other words, $\mathbf{d}_\mathbf{x}$ is the centroid of the vector associated with n-grams of the document $\mathbf{x}$. Intuitively, the sentiment of a document is related to the aggregated polarity of all its n-grams; that is, the more positive n-grams that exist in the document, it is more likely for it to express a positive opinion. While there are many possibilities for aggregation function, mean value provides a good summarization of the document's sentiment in this latent space. As suggested by [9], one can also use the $\mathbf{max}$ function that selects the maximum value along each latent dimension. Our empirical evidence indicate that the $\mathbf{mean}$ function is a more appropriate choice for the SA task.

Another fundamental reason for this formulation is that the number of phrases in the sentence is variable depending on the sentence length n. Thus, we need a function to compress the information from these phrases into a fixed length document embedding vector.

2.2 Sentiment Classification

We will consider two types of sentiment classification in this paper. First, similar to most prior approaches, we will consider binary classification setting that classifies a document either as a positive or negative. Next, we try to predict the text polarity toward its star-scale score directly using ordinal regression. It is our believe that this setting will become more prominent in the future since most of the online evaluation system are utilizing Likert-scale (e.g. 1-5 stars).

Given the document representation $\mathbf{d}_\mathbf{x}$ defined as above, the binary sentiment classification learns a function by optimizing the following loss measure:

$$\mathcal{L} = \sum_{\mathbf{x} \in S} (\text{sgn}(g(\mathbf{d}_\mathbf{x})) - y_\mathbf{x}), \tag{2}$$

where $\text{sgn}(g(\cdot)) \in \{1, -1\}$ are the prediction obtained from the classifier, and $y_\mathbf{x} \in \{1, -1\}$ is the label of the document $\mathbf{x}$. We chose a linear function for g, where

$$g(\mathbf{d}_\mathbf{x}) = \mathbf{a} \cdot \mathbf{d}_\mathbf{x} + b,$$

Linear classifiers have proven record of achieving the state of the art performance in previous sentiment classification results [19].

The Likert-scale classification problem belongs to the more general class known as "Ordinal Classification" [1], where the class labels have orderings. Utilizing this ordinal information in the classification will achieve better performance than treating each class separately without the order. There exist different methods to tackle ordinal classification or regression. Here, we present a simple marginal ordinal loss based strategy. We aim to learn a function g that optimizes the following loss:

$$\begin{aligned} \mathcal{L} = \sum_{\mathbf{x} \in S} \{ & \sum_{1 \le l \le y_\mathbf{x}} \max(0, 1 - g(\mathbf{d}_\mathbf{x}) + \beta_l) \\ + & \sum_{y_\mathbf{x} < l \le L} \max(0, 1 - \beta_l + g(\mathbf{d}_\mathbf{x})) \}. \end{aligned} \tag{3}$$

In this case, we are handling a L-likert-scale system. We have a set of boundaries β_l for each class $l \in [1, L]$. These boundaries are in ascending order, i.e $\beta_i < \beta_j, \forall i < j$. The function $g(\cdot)$ outputs a score for a document vector $\mathbf{d}_\mathbf{x}$. The discriminative training estimates the parameters of function $g(\cdot)$ and class boundaries $\beta_i, i \in [1, L]$ by minimizing the loss function $\mathcal{L}$.[4] The final classifier $g(\mathbf{d}_\mathbf{x})$ is defined as:

$$\underset{l \in [1,L]}{\arg\min} (\beta_{l-1} < g(\mathbf{d}_\mathbf{x}) < \beta_l).$$

The ordinal classification can be related to Rank Loss used in information retrieval society. Given a set of objects $x_i \in X, i \in$

[3] the non-linear function $\tanh(\cdot)$ converts unbounded range of the input into $[-1, 1]$

[4] For simplicity, we define $\beta_0 = -\infty$, for the first class that does not need a lower boundary for $g(\cdot)$

$[1, |\mathcal{S}|]$, their corresponding labels $y_i \in Y, i \in [1, |\mathcal{S}|]$, and a scoring function, the rank loss can be defined as:

$$\mathcal{L} = \sum_{i,j \text{ s.t. } y_i > y_j} \max(0, 1 - g(\mathbf{d}_{\mathbf{x}_i}) + g(\mathbf{d}_{\mathbf{x}_j})).$$

The objective is to learn function $g(\cdot)$ while minimizing the loss function $\mathcal{L}(\cdot)$. Clearly the score $g(\cdot)$ will try to put the objects in the order just like their labels. This order is consistent with the ordinal classification objective.

2.3 Embedding and Sentiment Classification within a Unified Model

We use the multi-layer perceptron network to implement the above modules in a unified framework. As illustrated in Table 1, the overall system can be represented in a 6-layer network architecture ($T = 6$):

level	layer	params
f_1	lookup table (word embedding)	$\mathbf{E}$
f_2	temporal convolution (n-gram embedding)	$\mathbf{F}$
f_3	transfer layer $\tanh$ (n-gram embedding)	-
f_4	mean function (document representation)	-
f_5	binary/ordinal classifier	$\mathbf{a}, b, \boldsymbol{\beta}$
f_6	loss function	-

Table 1: Parameters learned at each level.

We take advantages of the simple and efficient backpropagation process to train this layered network. The stacked layers in our network can be written in a more general form of multi-level functions:

$$l_{\mathbf{x}} = \mathbf{f}_T(\mathbf{f}_{T-1}(...(\mathbf{f}_1(\mathbf{x}))...)),$$

where $l_{\mathbf{x}}$ denotes the loss on a single example $\mathbf{x}$, and the exact loss function f_T is defined in Equation 2 and Equation 3. For a layer $\mathbf{f}_i, i \in [1, T]$, the derivative for updating its parameter set $\boldsymbol{\theta}_i$ is using the delta rule:

$$\frac{\partial l}{\partial \boldsymbol{\theta}_i} = \frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_i} \times \frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\theta}_i},$$

and the first factor on the right can be recursively calculated:

$$\frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_i} = \frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_{i+1}} \times \frac{\partial \mathbf{f}_{i+1}}{\partial \mathbf{f}_i}.$$

Note that $\mathbf{f}$ and θ are usually vectors (the scalar loss $l = \mathbf{f}_T$ can be treated as an all 1's vector), so $\frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_{i+1}}$ and $\frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\theta}_i}$ are Jacobian matrices, and "$\times$" is matrix multiplication.

We can use stochastic gradient descent (SGD) method to to perform training of parameters [7]. For a set of training samples, instead of calculating true gradient of the objective on all training samples, SGD calculates gradient and updates accordingly on each training sample. SGD is proved to be scalable and more efficient than batch-mode gradient descent methods, especially when dealing with large-scale datasets. The training procedure is presented in Algorithm 2.3. The 3rd column in Table 1 specify the parameters that will be learned during this end-to-end learning process in each layer.

Algorithm 1 End-to-End Training procedure for the proposed system

```
for j = 1 to MaxIter do
  if converge then
    break
  end if
  x, y ← random sampled data point and label
  calculate loss l(x; y)
  cumulative ← 1
  for i = 6 to 1 do
    ∂l/∂θ_i ← cumulative * ∂f_i/∂θ_i
    θ_i ← θ_i − λ ∂l/∂θ_i
    cumulative ← cumulative * ∂f_{i+1}/∂f_i
  end for
end for
```

We also want to point out that when using "deep learning" network, unsupervised pre-training of lower-layers is a common practice [3] and shows to improve the results. We adopt a similar strategy and use the projection matrix learned via LSI (using the term-doc matrix on a large unlabeled text set) to initialize the word embedding - E matrix, i.e. unsupervised pre-training, in our experiment.

3. RELATED WORK

Our proposed framework is closely related to the following areas and research topics.

3.1 Sentiment Analysis

An in-depth survey of the earlier methods for sentiment analysis has been presented in [20]. We will primarily focus on related works on polarity classification with similar objective tasks. The main approaches classify the polarity of a given text at either the word, sentence or paragraph, or document levels. In the most intuitive model, the polarity of an article can be related to the sentiment orientation of its words [13]. Latent semantic analysis has been used in [23] to to calculate the semantic orientation of the extracted words according to their co-occurrences with the seed words, such as "excellent" and "poor". The polarity of the article is then determined through averaging the sentimental orientation of its corresponding words. Instead of limiting the sentiment analysis at the word level, the mainstream research community performs sentiment classification at the article level. Various methods based on this principle have been proposed. These methods can be contrasted in terms of features they use: utilizing either unigram features [20] and/or filtered bigrams [6]. It is claimed that unigram and bigram features beat other features in the evaluation of several small-scale benchmark data sets [10]. It should also be noted that [19] investigated the use of several inverse document frequency (IDF) weighting schemes as features. They observed that such features improve the accuracy of sentiment classification. The third noteworthy trend is based on capturing existing substructures and their ordering in the text. In a recent paper, [22] used an HMM-based model to describe the dependency between local substructures to improve sentiment prediction. Similarly, [8] proposed a learning method for local content modeling (aspect-sentiment sense) using large-scale unsupervised data sets.

3.2 Latent Topic/Concept Models

There exists a number of word-level embedding methods that are able to capture semantic similarity between word pairs. One of

the earliest but widely used approaches is Latent Semantic Indexing (LSI) [11]. LSI applies SVD to term-document co-occurrence matrix, producing a low-dimensional representation for both documents (including unseen ones) and words, and enables efficient computation of semantic similarity between them. LSI is considered the pioneering work that inspired methods such as probabilistic LSI (pLSI) [14] and Latent Dirichlet Allocation (LDA) using a generative probabilistic framework [5].

There are also supervised variants that try to compute embedding by fitting to the labeled data. Such approaches have been applied to a variety of information retrieval tasks such as link prediction, cross-lingual retrieval, and image annotations [2, 25]. Another variant of supervised embedding method was recently introduced by [21] that learns (128 bit) hash-coding for term-frequency vectors of documents in the corpus, enabling semantic similarity-based hashing.

3.3 String Classification Using String Kernel

Broadly speaking, the proposed method is applicable to the general string classification problem. A variety of methods have been proposed to tackle the string classification problem, including generative (e.g., HMMs) or discriminative approaches, such as string kernel-based machine learning methods [15, 26]. The key idea of basic string kernels is to map text strings of variable length into a vectorial feature space of fixed length. In this space a standard classifier such as a support vector machine (SVM) can then be applied. As SVMs require only inner products between examples in the feature space, rather than the feature vectors themselves, one can define a *string kernel* which *implicitly* computes an inner product in the feature space. String kernels are designed so that the high similarity value between two text documents indicates that they have numerous n-grams in common.

We note that our proposed method implements a variation of inexact matching between n-grams which is also a critical component in the string kernel research, usually tackled by using different families of mismatch in the string kernel [16].

3.4 Deep Learning for Natural Language Processing

Lately, "deep learning" research grows to bring in attentions. As pointed out by [3], recent results have suggested that in order to learn the kind of complicated functions that can represent high-level abstractions (e.g. in natural language, or vision), one would need deep architectures. Each layer in the architecture represents features at a different level of abstraction, defined as a composition of lower-level features. Statistical language modeling is a key topic in natural language processing (NLP), where the difficulty is the curse of dimensionality, especially when modeling joint distribution between many discrete random variables. [4, 18] proposed a language model based on the multi-layered neural networks, which tries to model a distributed representation for each word and the probability function for word sequences, simultaneously. Later, [9] utilized a single multi-layered convolutional neural network architecture to handle multiple classic NLP tasks at the same time. The proposed unified framework provides an end-to-end system that, given a sentence, outputs a host of language processing predictions.

Our proposed method is motivated by the above approach and uses a multi-layer "deep" neural network to combine the n-gram embedding and sentiment classification in a single framework. It is our belief that the word embedding and n-gram embedding are feature learning in some sense.

4. EXPERIMENTS

4.1 Datasets

We have evaluated the performance of the proposed method on two sentiment dataset, Amazon and TripAdvisor that were introduced by [6] and [24], respectively. The Amazon dataset contains customer reviews of 25 various categories including apparel, automotive, baby, DVDs, electronics, magazines, tools and hardware, etc. The TripAdvisor data set contains customer reviews for various hotels across the globe. In addition to the overall reviewer's sentiment rating, this corpus contains scores for various aspects such as rooms, location, cleanliness, etc. However, sentiment scores for the specific aspects are absent from a significant number of the reviews in the TripAdvisor data set, so In our experiments we only consider overall ratings for this data set. These are considered some of the largest SA data sets currently available. We note that, our approach requires training for a large number of parameters in comparison to a simple shallow BOW model. Consequently, the advantage of our model becomes more obvious on larger datasets (we will show the effect in section 4.3.3).

Both datasets contain user-generated reviews where an overall sentiment for each review is quantified with an integer 1 through 5 (a.k.a the 5 stars scale). A sentiment score of 1 star corresponds to the lowest (negative) sentiment, while the score of 5 stars corresponds to the highest (positive) sentiment. For each dataset, we create a *balanced* version of the data that contains equal number of positive (4 and 5 stars) and negative (1 and 2 stars) reviews. TripAdvisor data also contains neutral reviews (3 stars) that do not exceed the number of positive nor negative reviews, so we add the neutral reviews to the balanced version of the data as well. In addition, we drop very small number of reviews from TripAdvisor with overall sentiment scores of zero stars, since the meaning of this sentiment rating is ambiguous. Statistics of both datasets can be found in Table 2.

We split balanced datasets with $70\%/30\%$ ratio into training and testing sets, respectively. We sample positive, negative, and (optionally) neutral reviews separately. This results in equal number of positive and negative reviews in both training and testing sets. Furthermore, in the case of Amazon dataset, each of the 25 categories is handled separately, resulting in balanced version of the data and splitting it into training/testing sets. This is done to obtain equally split sets in terms of polarity for each category, as the number of reviews varies significantly across the categories (e.g., from couple of hundreds to tens of thousands). Finally, once training and testing sets are constructed, we sample the training sets to obtain subsets required for validation. We use $20,000$ and $3,000$ reviews for validation from the Amazon and TripAdvisor datasets, respectively. The datasets used in our experiments are made available online. [5]

4.2 Evaluation Methodology

We have evaluated the performance of sentiment prediction using binary and ordinal classifications described in Section 2. We also present comparative results between our method and linear classifiers on bag-of-word features (1-gram and 2-gram). The latter approaches are considered to achieve the state of the art performance on sentiment prediction [19].

First, we outline the baseline procedure producing 1-gram and 2-gram features. We follow the method used by [6] to limit vocabulary size to $10,000$ terms with highest mutual information (MI), shared with the binary labels (positive or negative). As was men-

[5] `http://mst.cs.drexel.edu/datasets/CIKM2011`

	Amazon	TripAdvisor
$\star$	103,953	15,152
$\star\star$	80,278	20,040
$\star\star\star$	0	25,968
$\star\star\star\star$	48,086	15,141
$\star\star\star\star\star$	136,145	20,051
Train	237,900	64,445
Test	110,562	28,907
Validate	20,000	3,000
Total	368,462	96,352
$\|\mathcal{D}\|$	448,146	158,997
$\|\Gamma_1\|$	54,334	54,000
$\|\Gamma_2\|$	127,337	127,000

Table 2: Selected statistics for Amazon and TripAdvisor data sets. The table lists the number of reviews for each star rating, as well as for training, testing and validation sets separately. In addition, total number of reviews in each data set is provided. Original dictionary size ($|\mathcal{D}|$) for each data set is listed along with 1-gram (Γ_1) and 2-gram (Γ_2) vocabulary sizes. All of the numbers listed are obtained from the *balanced* versions of the datasets, containing equal number of positive and negative reviews.

tioned earlier, Amazon dataset contains reviews for 25 categories of consumer products. As a result, the MI-based procedure for building 1- and 2-gram vocabularies for the Amazon is performed on each category separately. In the interest of fairness, the MI-based procedure is performed only on the training data. Then $10,000$ terms for each of the 25 categories are concatenated together to form the final vocabulary that contains $54,334$ unique terms. This vocabulary is then used to filter all three sets of data; training, testing, and validation, for both Supervised Latent n-gram Analysis (SLNA) input and BOW input.

In addition to selecting single terms with highest mutual information to form 1-gram vocabulary Γ_1, we perform the same procedure on both unigrams and bigrams to form 2-gram vocabulary Γ_2, yielding $127,337$ features. For the TripAdvisor corpus, we also use MI-based procedure to build 1-gram and 2-gram vocabularies. However, since there is only one category for this corpus, we simply limit the vocabulary size to match the size of the corresponding vocabulary obtained from the Amazon training set. Table 2 lists 1- and 2-gram vocabulary sizes for both sentiment datasets.

We train the SVM classifier with linear kernel using LibLinear SVM tool[6]. Note that the size of the training data, as well as its dimensionality makes it impractical to use any non-linear kernel for the SVM classifier. This was previously pointed out by [19]. The SVM classifier is obtained using the entire training data, and its performance is evaluated on whole testing set. For each term, we used two weighting schemes, regular TF-IDF and Δ-IDF (we have implemented $a\Delta(t')n$ variant). According to [19], Δ-IDF achieves state of art performance on a small subset of the Amazon dataset. The optimal value for the cost constant C was chosen using validation set with simple parameter grid search on $C = \{2^{-7}, 2^{-6}, \ldots, 2^6, 2^7\}$. We found that for both datasets, setting $C = 32$ and $C = \frac{1}{32}$ for TF-IDF and Δ-IDF weighting schemes, respectively, yield the best validation performance.

The training of the perceptron classifiers is different under two representations: BOW linear or SLNA. Specifically, we perform training and validation of the perceptrons simultaneously in batches, where each batch consists of training for $100,000$ iterations followed by the validation testing on the entire set. Each iteration consists of one positive and one negative reviews selected at random from the appropriate set. We stop the training either after 250 batches were performed ($25,000,000$ training iterations), or when the running average validation error over the last 50 batches increased for the three consecutive batches. Following the training procedure, the classification error rate (reported in all of our experiments) is computed on the entire testing set using trained neural network. In our experiments, we set the dimensionality of the word embedding (Level 1 in Table 1) to $m = 50$, dimensionality of the n-gram embedding to $M = 30$ (Level 2 in Table 1), and size of the window to $n = 5$ words (reducing the window size increased error rate). Dimensionality of the word embedding ($m = 50$) was selected to match dimensionality of the LSI-based embedding, and is a common choice for the latter. Furthermore, reducing the dimensionality of the n-gram embedding to $M = 30$ slightly improved classification accuracy in our preliminary evaluations. Thus, we chose to fix this parameter throughout all experiments presented in Section 4.3.

[6] http://www.csie.ntu.edu.tw/~cjlin/liblinear/

4.3 Empirical Results

4.3.1 Binary Classification

The first experiment that we performed was to evaluate the performance of the proposed Latent n-gram Analysis method for a binary classification task – predicting polarity of the review (positive or negative). Binary classification setup was chosen, since it is the most common measure of performance for the sentiment prediction task. The binary classification error rates for Amazon and TripAdvisor datasets can be found in Table 3.

Method	Amazon	TripAdvisor
BOW Prc 1g	10.96	8.27
BOW SVM 1g	11.10	8.89
BOW Prc 2g	7.59	**7.37**
BOW SVM 2g	7.45	7.47
BOW SVM Δ-IDF 1g	10.91	8.74
BOW SVM Δ-IDF 2g	7.39	7.96
BOW LSI SVM 1g	21.40	24.18
SLNA	9.84	8.92
SLNA LSI	**7.12**	8.33
SLNA LT-FIX	15.4	-

Table 3: Average error rate under binary classification.

The results indicate that for Amazon dataset, SLNA initialized with LSI embedding achieved better performance in comparison to both 1-gram and 2-gram, including the SVM classifier using Δ-IDF. However, on TripAdvisor data, SLNA does not outperform the BOW methods. Our hypothesis is that this is due to the size limitation of the training data (60k) for the model. This will be further confirmed in Section 4.3.3. As a comparison of SLNA to an unsupervised embedding method, we also include results of a linear SVM trained on BOW LSI embeddings (BOW LSI SVM), and also a SLNA with fixed lookup table: i.e., classifier and n-gram embeddings are trained, but word embeddings are fixed to LSI embeddings (SLNA LT-FIX). Note that for BOW LSI SVM method, LSI-based document embedding was obtained in transductive setting (i.e., using entire balanced dataset), and the best error rate for 5-fold cross validation on each set is reported (cost parameters used: $C = \{2^{-7}, 2^{-6}, \ldots, 2^6, 2^7\}$). It is clear from Table 3

1g	vocabulary with unigrams
2g	vocabulary with unigrams and bigrams
BOW	bag-of-words text representation
Prc	perceptron classifier
SVM	support vector machine classifier
SLNA	Supervised Latent n-gram Embedding
LSI	lookup table is initialized with LSI-based latent vocabulary
BOW LSI	50-dimensional LSI document embedding for BOW
LT-FIX	SLNA structure initialized by LSI, except that we don't update word embedding in training
RL	The neural network for ordinal classification is initialized with the network pre-trained for margin ranking loss

Table 4: Acronym legend.

that neither of these two methods reaches the level of SLNA. For a description of acronyms, please refer to Table 4.

4.3.2 *Rank Loss and Ordinal Classification*

Table 5 shows the results on rank loss.

Method	Amazon	TripAdvisor
BOW Prc 2g	12.5	16.0
SLNA LSI	**10.2**	**13.7**

Table 5: Average error when using margin ranking loss. For TripAdvisor dataset, the experiments were performed on the data containing neutral reviews.

Clearly, SLNA outperforms BOW approaches on both Amazon and TripAdvisor datasets. This is different from the results we observed for binary classifications. We believe this behavior can be attributed to the fact that pair-wise training examples generates more training data than simple binary classification. Furthermore, the rank loss is consistent with the ordinal classification results presented in Table 6, where SLNA yields the best performance in all scenarios. We are not aware of any prior art reporting results for ordinal classification on these two datasets, so direct comparisons is not possible. For the sake completeness, we calculated mean squared error (MSE) based on predictions with ordinal classification. We obtained MSE $= 1.03$ on Amazon dataset using SLNA LSI RL model, which is considerably better than previously reported MSE ≈ 1.5 on a small subset of Amazon dataset [17].

Method	Amazon	TripAdvisor
BOW Prc 2g	37.8	44.9 (52.1)
BOW Prc 2g RL	35.8	41.5 (51.6)
SLNA LSI	30.7	42.7 (51.4)
SLNA LSI RL	**28.2**	**39.6 (49.2)**

Table 6: Ordinal classification average error rate. The error rate for 5 star classification (i.e., including neutral reviews) are provided in parentheses.

4.3.3 *Effect of the training set size*

In table 2, we train Amazon data on the binary classification task, while varying the size of the training data. We can see that as we increase the size of the training set, the gap between SLNA and BOW diminishes and eventually SLNA out-performs 2-gram BOW.

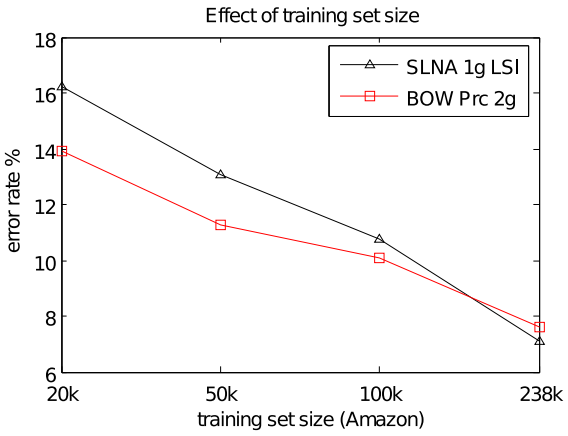

Figure 2: Testing error on Amazon with different training set size.

5. CONCLUSIONS

It is our belief that an n-gram model, combined with latent representation, will produce an more suitable embedding for document-level classification tasks, such as sentiment polarity prediction. Our proposed embedding combines n-gram representation with a latent model to produce a simple and efficient embedding for short segments of text. Our experimental evaluation indicates that for binary classification and regression tasks, our method with 1-gram (i.e., single words) dictionary of size $|\mathcal{D}|$ achieves accuracy of BOW representation with a 1- and 2-gram dictionary of size $|\mathcal{D}|^2$. Furthermore, we show that in a multi-class experimental setting (i.e., predicting star values for reviews) our embedding outperforms the baseline methods based on BOW with 2-gram dictionary.

One potential limitation associated with the proposed method is its need of large enough training set for model training. However, due the increasing availability of large scale SA datasets become available, this is not considered a major shortcoming. Moreover, our study indicates the model trained on mixed categories has excellent performance on both mixed and separated categories. These are all indications of great potential of proposed method in sentiment analysis. This model is not restricted to the task of sentiment classification and can potentially be applied for other document classification applications.

6. ACKNOWLEDGMENTS

Ali Shokoufandeh and Dmitriy Bespalov gratefully acknowledges the support from the National Science Foundation Grant #0803670 under the IIS Division and Office of Naval Research Grant ONR-N000140410363.

7. REFERENCES

[1] A. Agresti. *Analysis of Ordinal Categorical Data*. John Wiley and Sons Inc., 2010.

[2] B. Bai, J. Weston, R. Collobert, D. Grangier, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Supervised semantic indexing. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 187–196. ACM, 2009.

[3] Y. Bengio. *Learning Deep Architectures for AI*. Now Publishers Inc., Hanover, MA, USA, 2009.

[4] Y. Bengio, R. Ducharme, P. Vincent, and D. D. E. R.

Operationnelle. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2000.

[5] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[6] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *In ACL*, pages 187–205, 2007.

[7] L. Bottou. Stochastic learning. In O. Bousquet and U. von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.

[8] S. Brody and N. Elhadad. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 804–812, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[9] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*, 2008.

[10] H. Cui, V. Mittal, and M. Datar. Comparative experiments on sentiment classification for online product reviews. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1265–1270. AAAI Press, 2006.

[11] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of The American Society for Information Science*, 41(6):391–407, 1990.

[12] G. Ganu, N. Elhadad, and A. Marian. *Beyond the stars: improving rating predictions using review text content*. 2009.

[13] V. Hatzivassiloglou and K. R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98, pages 174–181, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[14] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM Press New York, NY, USA, 1999.

[15] P. Kuksa, P.-H. Huang, and V. Pavlovic. Scalable algorithms for string kernels with inexact matching. In *NIPS*, 2008. Spotlight Presentation. Acceptance rate: 123/1022 (12

[16] C. S. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. In *NIPS*, pages 1417–1424, 2002.

[17] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation with multiple sources. In *NIPS'08*, pages 1041–1048, 2008.

[18] F. Morin. Hierarchical probabilistic neural network language model. aistatsâĂŹ05. In *In AISTATS*, pages 246–252, 2005.

[19] G. Paltoglou and M. Thelwall. A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1386–1395, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[20] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[21] R. Salakhutdinov and G. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50:969–978, July 2009.

[22] C. Sauper, A. Haghighi, and R. Barzilay. Incorporating content structure into text analysis applications. In *EMNLP'10*, pages 377–387, 2010.

[23] P. Turney and M. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21:315–346, 2003.

[24] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 783–792, New York, NY, USA, 2010. ACM.

[25] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: Learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.

[26] J. Weston, C. Leslie, E. Ie, D. Zhou, A. Elisseeff, and W. S. Noble. Semi-supervised protein classification using cluster kernels. *Bioinformatics*, 21(15):3241–3247, 2005.