Simple Algorithms For Sentiment Analysis On Sentiment Rich, Data Poor Domains.

Prathusha K Sarma
University of Wisconsin-Madison
kameswarasar@wisc.edu

William A Sethares
University of Wisconsin-Madison
sethares@wisc.edu

Abstract

Standard word embedding algorithms learn vector representations from large corpora of text documents in an unsupervised fashion. However, the quality of word embeddings learned from these algorithms is affected by the size of training data sets. Thus, applications of these algorithms in domains with only moderate amounts of available data is limited. In this paper we introduce an algorithm that learns word embeddings jointly with a classifier. Our algorithm is called SWESA (Supervised Word Embeddings for Sentiment Analysis). SWESA leverages document label information to learn vector representations of words from a modest corpus of text documents by solving an optimization problem that minimizes a cost function with respect to both word embeddings and the weight vector used for classification. Experiments on several real world data sets show that SWESA has superior performance on domains with limited data, when compared to previously suggested approaches to word embeddings and sentiment analysis tasks.

1 Introduction

Word embedding algorithms learn vector representations for words that are useful to quantify semantic relationships between words in a given text. Additionally, word embeddings are used to initialize several algorithms for sentiment analysis, sentence encoding etc. Currently popular embedding algorithms such as word2vec (Mikolov et al., 2013b; Le and Mikolov, 2014), GloVe (Pennington et al., 2014), are based off neural network methods and have achieved tremendous success in various word, sentence/document level evaluation tasks. These algorithms thrive on the large volumes of training data sets when learning high quality word embeddings. However, there is an increase in application domains where getting large amounts of data is not always possible. Furthermore, in some of these domains, representing words from off-the-shelf word embeddings such as ones obtained from training word2vec, GloVe on Wikipedia or common-crawl may not be efficient. This is because the sentiment expressed by a word in such datasets could be somewhat different from the sentiment expressed by the same word when it appears in Wikipedia, common-crawl. A concrete example of such domains is given in the next paragraph. *In such cases off-the-shelf word embeddings are not very useful and better techniques are required to learn word embeddings and classifiers for sentiment analysis.*

The goal of this paper is to build classification algorithms for sentiment analysis on small, domain specific data sets that are sentiment rich. Such an algorithm is not limited to sentiment classification and easily generalizes to other text classification problems as well. An example of a small sized and sentiment rich data set is the Substance Use Disorder (SUD) data set (Mohr et al., 2013), (Moore et al., 2011) obtained from digital health intervention treatments. These treatments aim to predict relapse risk by analyzing the content of participants' text messages. Though forum moderators can monitor and provide support when participants are struggling, considerable labor is involved in reviewing and deciding the risk level of each text message. Text data obtained from these discussion forums is rich in sentiments such as 'determination,' 'pleasure,' 'anger,' 'fear' and by analyzing discussion messages for sentiments such as ‘anger’, ‘fear’ it is possible to develop efficient algorithms that can automatically tag if a message is positive (i.e. benign) or negative (indicative of relapse).

One approach to this problem would be to use off-the-shelf word embeddings to learn document embeddings and then run a classification algorithm on the obtained embeddings. While this approach does decently well it fails to capture and exploit the semantics of within domain words. For example, consider words such as ‘alcohol,’ ‘holiday,’ ‘party.’ Such words are typically neutral or positive sentiment in the Wikipedia corpus. However, with data sets such as SUD, these words are indicative of a moderate/strong negative sentiment. As a result using off-the-shelf word embeddings from GloVe, word2vec embeddings trained on Wikipedia corpus for building a sentiment analyzer will result in a sentiment analyzer that performs poorly. An alternative is to obtain word embeddings by training word2vec or GloVe on the target dataset, and then using the resultant embeddings for sentiment analysis. Unsurprisingly this approach succeeds only when one has the luxury of large datasets. However, in the SUD application, as mentioned above, the amount of data that is available is limited and hence this approach is not a successful alternative. In fact we argue that in the presence of limited, but sentiment rich data, it is better to learn word embeddings in a supervised manner. This way the resulting embeddings tend to be polarity-aware and are better suitable for downstream tasks such as sentiment analysis. Our contributions are as follows,

1. We introduce an algorithm (Section 3) that jointly learns word embeddings as well as a sentiment analyzer (classifier) by solving a bi-convex optimization problem. Our algorithm is called Supervised Word Embedding for Sentiment Analysis (SWESA) . This is an iterative algorithm that minimizes a cost function for both a classifier and word embeddings under unit norm constraint on the word vectors.

2. SWESA uses document labels for learning word embeddings. Using document labels within SWESA helps us overcomes the problem of small-size training data and allows learning of polarity-aware word embeddings. Via a thorough empirical evaluation (Section (4)) we show that our algorithm outperforms classifiers built by re-training off-the-shelf word embeddings such as word2vec, GloVe. We also compare SWESA against an algorithm that uses convolutional neural networks to represent sentences (Kim, 2014) for sentiment analysis, and a sentiment analysis algorithm based on recurrent neural networks (Socher et al., 2013). On the A-CHESS dataset where data is limited but rich in sentiment, SWESA outperforms all algorithms by at least 12% on the precision.

3. To demonstrate the fact that the word embeddings learned by SWESA are better than embeddings learned from unsupervised learning algorithms we investigate the polarity of various word embeddings. We show that the embeddings learned from SWESA perform well on antonym task. For example, ‘Awful/Good’ is the antonym pair returned via SWESA as opposed to ‘Awful/Could’ obtained via word2vec. SWESA learns such antonym pairs, using document polarities, and as a byproduct of our optimization based formulations, independent of an antonym pairs training data set.

2 Related Work

Modern embedding algorithms such as word2vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) are neural network based algorithms that learn word embeddings in an unsupervised fashion. These algorithms exploit word co-occurrence statistics in order to learn word embeddings. Due to this, typically these algorithms thrive on large training data sets in order to learn generic word embeddings. Also these models learn word embeddings in an unsupervised fashion. However, using labeled data can often help with learning sentiment-aware word embeddings more appropriate to the corpus at hand In their work (Maas et al., 2011) propose a probabilistic model that captures semantic similarities among words across documents. This model leverages document label information to improve word vectors to better capture sentiment of the contexts in which these words occur. While it may seem that this model is similar to SWESA, it is not so because in their model (Maas et al., 2011) first learn word embeddings and then fit a classifier in two different objective functions. On the other hand SWESA is a single bi-convex objective. The probabilistic model used by (Maas et al., 2011) is similar to that in Latent Dirichlet

Allocation (LDA) (Blei et al., 2003) in which each document is modeled as a mixture of latent topics. In (Maas et al., 2011), word probabilities in a document are modeled directly assuming a given topic. Yet another model that makes use of word co-occurence features in a supervised learning task is that of (Aga et al., 2016). In this model word co-occurences and context are used to learn word embeddings through matrix factorization algorithms. However, the classification task is independent of the process used to learn word embeddings.

(Tang et al., 2014) propose a supervised neural network based model to classify Twitter data. The proposed algorithm learns sentiment specific word vectors, from tweets making use of emoticons in text to guide sentiment of words used in the text instead of annotated sentiment labels. The Recursive Neural Tensor Network (RNTN) proposed by (Socher et al., 2013) classifies sentiment of text of varying length. To learn sentiment from long text, this model exploits compositionality in text by converting input text into the Sentiment Treebank format with annotated sentiment labels. The Sentiment Treebank is based on a data set introduced by Pang and Lee (Pang and Lee, 2005). This model performs particularly well on longer texts by exploiting compositionality as opposed to a regular bag of features approach. A popular algorithm for sentiment analysis is the CNN based approach proposed by (Kim, 2014). This algorithm takes as input word embedding and learns a CNN based sentence composition on which sentiment analysis is performed. Finally we would like to mention that in some domains, the vocabulary in the domain is manually labeled. For example, SentiStrength (Thelwall et al., 2010), LIWC (Pennebaker et al., 2001) projects provide manually labeled data. However, such efforts are not scalable and need considerable human expertise.

Notation: Throughout this paper we shall denote word vectors as $\mathbf{w}_j \in \mathbb{R}^k$, for $j = 1, \ldots, V$, where V indicates the size of the vocabulary. The matrix of word vectors is $\mathbf{W}$ where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_V] \in \mathbb{R}^{k \times V}$. The classifier to be learned is represented by $\boldsymbol{\theta} \in \mathbb{R}^k$, weights of word vectors $\mathbf{w}_j$ in document i are contained in the vector $\boldsymbol{\phi}_i \in \mathbb{R}^V$, and the document label of the i^{th} document is indicated by y_i, document i is represented as $d_i = \mathbf{W}\,\boldsymbol{\phi}_i$. Let $\Phi = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_N] \in \mathbb{R}^{V \times N}$ be the matrix containing weight vectors $\boldsymbol{\phi}_i$ and vector $\mathbf{y} = [y_1, y_2, ...y_N]$ be the vector containing document labels.

3 Supervised Word Vectors for Sentiment Analysis

Given a collection of documents $d_1, d_2, \ldots d_N$ with binary sentiments $y_1, y_2, \ldots, y_N$ respectively, the aim is to learn a classifier that when given a new, previously unseen document d can accurately estimate the sentiment of the document. There could be class imbalance in the training data and so the algorithm should explicitly account for such a class imbalance. We approach this problem by introducing a new algorithm called SWESA. SWESA simultaneously learns word vector embeddings and a classifier, by making use of document polarity/sentiment labels. Representation of documents within SWESA is motivated by the fact that in short texts like “I am sad”, “I am happy”, polarity of the sentence hinges on the words “sad” and “happy”. As a result, by learning polarity aware word embeddings, good vector representations for documents can be achieved. For instance, in the above example, the distance between the vectors $(\mathbf{w}_I + \mathbf{w}_{am} + \mathbf{w}_{sad})$ and $(\mathbf{w}_I + \mathbf{w}_{am} + \mathbf{w}_{happy})$ would capture dissimilarities in sentiment of these two documents while at the same time reflecting similarities in sentence structure.

Text documents in this framework are represented as a weighted linear combination of words in a given vocabulary. Weights can be either the term frequencies (tf) of words within each document or term frequency-inverse document frequency (tf-idf). Weights provided as input to SWESA for experiments described in Section (4) are term frequencies. We choose this weighting scheme to mimic the concept of local context used in the word2vec family of algorithms. Global co-occurrence information can be leveraged by using tf-idf for weighting words in documents. Such an approach in not entirely unheard off in sentiment analysis tasks, where word embeddings are considered as features for a classification algorithm (Labutov and Lipson, 2013).

SWESA aims to find vector representations for words, which when combined using weights, in a bag-of-words framework provide us with embeddings for text documents. These embeddings should be such that applying a nonlinear transformation f to the product $(\theta^\top\,\mathbf{W}\,\phi)$ results in a binary label y indicating the polarity of document. Mathematically we assume that,

$$\mathbb{P}[Y = 1|d = \mathbf{W}\,\boldsymbol{\phi}, \boldsymbol{\theta}] = f(\boldsymbol{\theta}^\top\,\mathbf{W}\,\boldsymbol{\phi}) \tag{1}$$

for some function f. In order to solve for $\boldsymbol{\theta}$ and $\mathbf{W}$, a regularized negative likelihood minimization problem is solved. This optimization problem is as (1) and can be solved as a minimization problem with objective function,

$$J(\boldsymbol{\theta}, \mathbf{W}) \overset{\text{def}}{=} \frac{-1}{N}\Big[\mathrm{C}_+ \sum_{y_i=+1} \log \mathbb{P}(Y = y_i|\,\mathbf{W}\,\boldsymbol{\phi}_i, \boldsymbol{\theta}) + \mathrm{C}_- \sum_{y_i=-1} \log \mathbb{P}(Y = y_i|\,\mathbf{W}\,\boldsymbol{\phi}_i, \boldsymbol{\theta})\Big] + \lambda_{\boldsymbol{\theta}} ||\,\boldsymbol{\theta}\,||_2^2. \tag{2}$$

This optimization problem can now be written as

$$\min_{\substack{\boldsymbol{\theta}\in\mathbb{R}^k,\\ \mathbf{W}\in\mathbb{R}^{k\times V}}} \quad J(\boldsymbol{\theta}, \mathbf{W}) \tag{3}$$
$$\text{s.t. } ||\,\mathbf{w}_j\,||_2 = 1\;\forall j = 1, \ldots V.$$

The vector $\boldsymbol{\phi}_i$ is a vector of weights, corresponding to the different words, for document d_i. As mentioned previously, for testing SWESA term frequencies of different words in a certain document i are used in $\boldsymbol{\phi}_i$. $\lambda_\theta > 0$ is the regularization parameter for the classifier θ, C_+ is the cost associated with misclassifying a document from the positive class and C_- is the cost associated with misclassifying a document from the negative class. Following the heuristic suggested by (Lin et al., 2002), $\mathrm{C}_+ = \frac{\mathrm{N}_-}{N}$ and $\mathrm{C}_- = \frac{\mathrm{N}_+}{N}$, where N_+ is the number of positive documents in the corpus and N_- is the number of negative documents in the corpus. This scheme is particularly useful when dealing with data sets with imbalanced classes. When using a balanced data set $\mathrm{C}_+ = \mathrm{C}_-$. Sentiment in a given document is captured by the document label y_i, which in this framework is a binary label that capture sentiments such as 'positive/negative' or 'threatening/benign' depending on the data set.

The unit norm constraint in the optimization problem shown in (3) is enforced on word embeddings to discourage degenerate solutions of $\mathbf{w}_j$. For example in the absence of this constraint, the optimal $\mathbf{w}_j^*$ is typically a vector of zeros. Note that this optimization problem is bi-convex, but it is not jointly convex in the optimization variables. Algorithm 1 shows the algorithm that we use to solve the optimization problem in (3). This algorithm is an alternating minimization procedure that initializes the word embedding matrix $\mathbf{W}$ with $\mathbf{W}_0$ and then alternates between minimizing the objective function w.r.t. the weight vector $\boldsymbol{\theta}$ and the word embeddings $\mathbf{W}$.

Algorithm 1 Supervised Word Embeddings for Sentiment Analysis (SWESA)

```
Require: $\mathbf{W}_0$, $\Phi$, $\mathrm{C}_+$, $\mathrm{C}_-$, $\lambda_\theta$, $0 < k < V$, Labels: $\mathbf{y} = [y_1, \ldots, y_N]$, Iterations: $T > 0$,
1: Initialize $\mathbf{W} = \mathbf{W}_0$.
2: for $t = 1, \ldots, T$ do
3:     Solve $\boldsymbol{\theta}_t \leftarrow \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \mathbf{W}_{t-1})$.
4:     Solve $\mathbf{W}_t \leftarrow \arg\min_{\mathbf{W}} J(\boldsymbol{\theta}_t, \mathbf{W})$.
5: end for
6: Return $\boldsymbol{\theta}_T, \mathbf{W}_T$
```

3.1 Logistic regression model

The optimization problem in (2) assumes a certain probability model and minimizes the negative log-likelihood under norm constraints. While, the specific goal of the user might dictate an appropriate choice of probabilistic model, for a large class of classification tasks such as sentiment analysis, the logistic regression model is widely used. In this section we assume that the probability model of interest is the logistic model. Under this assumption the minimization problem in Step 3 of Algorithm 1 is a standard logistic regression problem [1]. Many specialized solvers have been devised for this problem and in this implementation of SWESA, a standard off-the-shelf solver available in the scikit-learn package in Python is used. In order to solve the optimization problem in line 4 of Algorithm 1 a projected stochastic gradient descent (SGD) with suffix averaging (Rakhlin et al., 2011) is used. In suffix averaging the last few iterates obtained during stochastic gradient descent are averaged. Suffix averaging guarantees that the noise in the iterates is reduced and has been shown to achieve almost optimal rates of convergence for minimization of strongly convex functions. For experiments in Section 4 we set $\tau = 50$.

Gradient updates for $\mathbf{W}$ given $\boldsymbol{\theta}$ are of the form

$$\nabla J(\boldsymbol{\theta}, \mathbf{W}) = \frac{1}{N}\Big[\sum_{y_i=+1} \frac{-\,\mathrm{C}_{+}\, y_i\, \boldsymbol{\theta}\, \boldsymbol{\phi}_i^\top}{1+e^{y_i(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}_i)}} + \sum_{y_i=-1} \frac{-\,\mathrm{C}_{-}\, y_i\, \boldsymbol{\theta}\, \boldsymbol{\phi}_i^\top}{1+e^{y_i(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}_i)}}\Big]. \tag{4}$$

Algorithm 2 implements the SGD algorithm (with stochastic gradients instead of full gradients) for solving the optimization problem in step 4 of Algorithm 1.

Algorithm 2 Stochastic Gradient Descent for $\mathbf{W}$

```
Require: $\boldsymbol{\theta}, \gamma, \mathbf{W}_0$, Labels: $\mathbf{y} = [y_1, \ldots, y_N]$, Iterations: N, step size: $\eta > 0$, and suffix parameter:
    $0 < \tau \leq N$.
1: Randomly shuffle the dataset.
2: for $t = 1, \ldots, N$ do
3:     Set $C_t = C_+$ if $y_t = +1$, $C_t = C_-$ if $y_t = -1$.
4:     $\widetilde{\mathbf{W}}_{t+1} = \mathbf{W}_t - \frac{\eta C_t}{1+e^{y_i(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}_i)}} \times (-y_i\, \boldsymbol{\theta}\, \boldsymbol{\phi}_i^\top)$
5:     $\mathbf{W}_{t+1,j} = \mathbf{W}_{t+1,j} / \|\, \mathbf{W}_{t+1,j} \,\|_2 \; \forall j = 1, 2, \ldots, V$
6:     $\eta \leftarrow \frac{\eta}{t}$
7: end for
8: Return $\mathbf{W} = \frac{1}{\tau} \sum_{t=N-\tau}^{N} \mathbf{W}_t$
```

Convergence of SWESA: At a high level, SWESA can be seen as a variation of the supervised dictionary learning problem (SDL). Within SDL (Mairal et al., 2009) given labeled data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, and unlabeled part of the data that lies in a d dimensional space, the goal is to learn a dictionary D of size $d \times k$, $(k >> d)$ such that each $x_i = Dz_i$ where z_i is a sparse encoding of x_i w.r.t. dictionary D. Further, the label is generated by a linear classifier w.r.t z_i, i.e. $y_i = \mathbf{W}^\top z_i$. The learning problem is to estimate the dictionary, the codes of each data point and the classifier. SWESA can be roughly mapped to the SDL by considering dictionary D of size $k \times V$, where each column corresponds to a word embedding. However, there are significant differences between SWESA and SDL. Despite these differences, with suitable modifications convergence analysis for SWESA can be performed. Extensive literature on alternative minimization and their convergence guarantees for SDL and related problems such as matrix completion (Jain et al., 2013) exist. A full theoretical analysis is out of scope of this paper and is left for future work. A brief description of differences between SWESA and SDL can be found in Appendix A.

3.2 Initialization of W

Two different initialization procedures are used to obtain $\mathbf{W}_0$. The first method uses the Latent Semantic Analysis (Dumais, 2004) (LSA) procedure to form the matrix of word vectors $\mathbf{W}_0$ from the given corpus of text documents. The second method uses the word2vec (w2v) algorithm to form word vector matrix $\mathbf{W}_0$ from the corpus. Since word2vec is ideally trained on a dataset that is much larger in size than the test datasets in Section 4, it is expected that the initialization vectors obtained by retraining word2vec on the test data sets will not be as suitable as the LSA intialization. However, as expected SWESA improves the performance of any initial guess parameters.

[1]A bias term, γ can be trivially introduced in the logistic regression model.

4 Experimental Evaluation and Results

We now describe in brief the data sets and the algorithms that we use to evaluate SWESA.

4.1 Data Sets

We conduct our experiments on four small sized data sets out of which 3 are balanced and 1 is imbalanced. All four data sets are drawn from different domains and hence have different vocabularies and contexts. The four data sets are:

- **Yelp:** This data set consists of 1000 restaurant reviews labeled 'positive' or 'negative'. There are a total of 2049 distinct word tokens in this data set.
- **IMDB:** This data set consists of 1000 reviews of movies with labels 'positive' or 'negative'. There are a total of 3075 distinct word tokens.
- **Amazon:** This data set contains 1000 product reviews with labels 'positive' or 'negative'. This data set has 1865 distinct word tokens.
- **A-CHESS:** This is a proprietary data set[2] and is obtained from an intervention treatment aimed at alcohol disorder. Text is obtained via the discussion forums part of the A-CHESS mobile app. There are a total of 2500 messages with 8% of the messages indicating relapse risk or 'threat'. Typical messages from this data set are of the form, *"I've been clean for 7 months now, but I still feel I may not make it"*. Such a message is marked as 'threat'. Positive or 'benign' messages are of the form, *"30 days and sober! I feel great!!"*. This labeling is human moderated. The goal is to automate this process. After removing special characters, this data set consists of 3400 distinct word tokens.

The first three data sets are obtained from (Kotzias et al., 2015).

4.2 Baselines

We choose two types of baselines that cover both neural and non-neural network style algorithms.
Standard baselines:

1. **Two-Step (TS):** The TS algorithm is a family of algorithms where we perform two steps (i) First, word embeddings are trained in an unsupervised manner on the target dataset. These word embeddings are combined together with appropriate weights (as is done in SWESA) to obtain document embeddings. (ii) In the second step the document embeddings obtained from the first step are used in a logistic regression classifier to obtain a classification model. To implement our first step we use two types of word embeddings, namely LSA word embeddings, and word2vec (re-w2v) obtained by re-training LSA, and word2vec algorithms on our target datasets. We use an open source implementation in gensim[3] to get word2vec embeddings.
2. **Naive Bayes classifier:** The classic Naive Bayes classifier for sentiment classification based on the Bag-of-words features, optimized in NLTK toolkit in Python is used.

Neural network based baselines:

1. **Recursive Neural Tensor Network (RNTN):** RNTN is an RNN proposed by (Socher et al., 2013) that learns compositionality from text of varying length and performs classification in a supervised fashion with fine grained sentiment labels. Since SWESA is aimed at binary classification, RNTN is also used in a binary classification framework. RNTN has been shown to perform better than the previously proposed Recursive Auto Encoder (RAE) by (Socher et al., 2011) and hence we limit our comparisons to RNTN only.

[2]Center for Health Enhancement System Services at UW-Madison.
[3]`https://radimrehurek.com/gensim/models/word2vec.html`

2. **CNN based sentence classification:** (Kim, 2014) propose a Convolutional Neural Network (CNN) based architecture that takes as input word embeddings and uses a CNN based architecture to learn sentence embeddings. These sentence embeddings are then used for classification. This algorithm belongs to a large class of algorithms that take as input word embedding and learn sentence embeddings for other tasks. While it seems that SWESA is similar to these algorithms, the significant difference between the two lies in the application domain of SWESA. SWESA is geared towards small sized datasets, whereas neural network based algorithms are data intensive and can extract useful relationships from very large amounts of data.

Note that neural network based baselines are used in two modes,

- **Pre-trained (Pr-tr):** In this framework both RNTN and CNN-static (CNN-S), CNN-non static (CNN-NS) are initialized with word2vec embeddings (w2v) and are trained on the Pang and Lee data set (Pang and Lee, 2005). This data set consists of roughly 10k text documents and is roughly 10 times the size of the data sets considered in experiments in Section 4.4. Note that in these experiments, word embeddings used are obtained from word2vec trained on the Google news data set and do not make use of any labels from the test data sets considered.

- **Re-trained (Re-tr):** In this framework, RNTN is retrained on the data sets described in subsection 4.1. Further, CNN-static and non static are used in the following ways,

  1. CNN-static and CNN-non static are initialized with re-trained w2v (re-w2v) embeddings. The training set is still the Pang and Lee data set used by the authors (Kim, 2014). We expect the performance of this baseline to be poorer when compared to CNN-static and non static initialized with pre-trained word embeddings. This baseline is used to investigate the sensitivity of CNNs to initial input word embeddings.
  2. CNN-static and CNN-non static are initialized with pre-trained word2vec embeddings but trained on train/dev sets obtained from the data sets described in subsection 4.1. We expect this data set to be the least well performing data set due to the size of the training data. This baseline is used to illustrate the limitations of small sized training data sets on neural network algorithms.

4.3 Dimensionality of word embeddings and hyperparameters

Dimensions of word embeddings and other hyperparameters such as regularization on the logistic regression classifier are determined via 10 fold cross validation. Pre-trained RNTN[4] and CNN-static/non static[5] are used with the training sets and parameters as described by the authors of both works. Procedure for retraining RNTN as well as for fine tuning word embeddings for use in CNN-non static follow the methods described in (Socher et al., 2013) and (Kim, 2014) respectively.

4.4 Results from classification tasks

Performance metrics reported are average Precision and average AUC. AUC is only reported for model that provide prediction probabilities in addition to predicted label. For the A-CHESS data set the minor class, i.e 'threat' owing to the large imbalance is of more importance. This is because, the system can accept few false positives, but the risk of misclassifying a 'threat' message as 'benign' has a larger impact. Owing to this, on this data set, the best performing classifier will be one that achieves maximum precision. Note that the objective of these results is to show how well SWESA performs in applications with small sized data sets as compared to i) completely re-training a neural network with a smaller training set, ii) or initializing a neural network with word embeddings learned on small data sets and iii) against a neural network that uses pre-trained word embeddings and larger training sets.

SWESA against pre-trained neural nets initialized with pre-trained word embeddings: On the balanced Yelp, Amazon and IMDB data sets, pre-trained RNTN, CNN-static and non static perform the

[4]`https://nlp.stanford.edu/sentiment/code.html`
[5]`https://github.com/yoonkim/CNN_sentence`

best. From table 1, observe that the best performing baseline CNN-NS achieves an average precision of 87.98, 87.15 and 87.77 respectively on the three balanced data sets. This result is *not surprising*, given that the pre-trained RNTN and CNNS are i)initialized with pre-trained word embeddings and ii)trained on a data set with roughly 10 times more training data points than within SWESA. However, note that on the imbalanced A-CHESS data set RNTN fails to perform any classification. This can be attributed to the fact that RNTN is a dependency parser and on varying length text like within SWESA, it is very hard to capture dependencies. CNN-static and CNN-non static achieve poor precision owing to the class imbalance which is not accounted for when training them.

SWESA against re-trained neural nets: CNN-static (re-w2v) and CNN-non static (re-w2v) perform comparably well to SWESA on the balanced data sets. From table 1 notice that on the balanced Yelp, Amazon and IMDB data sets, SWESA achieves an average precision of 78.35, 80.36 and 77.27 respectively while CNN-non static (re-w2v) achieves an average precision of 78.10, 80.40 and 75.85 respectively. These results suggest that SWESA is a suitable alternative to neural network architectures that are sensitive to initializing word embeddings. This result is particularly useful for data sets that are not large enough to enable learning of high quality word embeddings. Note that on the A-CHESS data set CNN-static and non static perform poorly.

SWESA against neural nets initialized with re-trained word embeddings: re-trained RNTN performs very poorly on the three balanced data sets and fails to classify the A-CHESS test set. On the other hand while CNN-static (re-w2v) and CNN-non static (re-w2v) perform comparably well to SWESA on the balanced data sets, both algorithms perform poorly on the A-CHESS data set. CNN-static and non static achiever average precision of 15.62 and 18.98 respectively on the A-CHESS data set while SWESA achieves an average precision of 35.80. This is not a surprising result because the training data to RNTN and CNNs is much smaller than the Pang and Lee Data set.

Polarity of word embeddings. The objective of SWESA is to perform effective sentiment analysis by learning embeddings from text documents with sentiment labels. Since, word embeddings are learned via a joint optimization framework one can expect that the resulting word embeddings are polarity aware. To test our hypothesis, we investigate if one can predict the antonym of a word. That is, given a word v whose word embedding is $\mathbf{w}_v$ we determine the antonym to be that word a with word embedding $\mathbf{w}_a$ such that the cosine similarity between $\mathbf{w}_a, \mathbf{w}_v$ is minimized over all possible choices of a. Figure 1 shows a small sample of word embeddings learned on the Amazon data set by SWESA and word2vec. The cosine similarity (angle) between the most dissimilar words is calculated and owing to the assumptions on word embeddings, words are depicted as points on the unit circle.

From figure 1 it is evident that a supervised algorithm like SWESA projects document level polarity onto word level embeddings while an unsupervised algorithm like word2vec that learns embeddings of words via virtue of word co-occurrences will fail to embed polarity. It is important to notice that SWESA learns word polarities by using document polarities, and these word polarities are useful for antonym tasks. Unlike classical antonym tasks where examples of known antonym pairs are provided, in our setup no such pairs are provided, and yet SWESA was able to do a good job discovering antonym pairs. For example the most dissimilar word to given word ‘Excellent’ is ‘Poor’ when learned via SWESA as opposed to ‘Work’ when learned via word2vec. From these examples, the advantage of SWESA is particularly towards exploiting strong polarity differences in words such as ‘Excellent’ and ‘Poor’, however not much success is noted towards less polarized synonyms such as ‘Wet’ and ‘Dry’. However, with finer grained sentiment labels we would expect subtler antonym pairs to be distinguished by SWESA. This will be actively pursed in future extensions of SWESA, particularly when used to perform multi-class classification tasks.

Thus, word antonym pairs (w_a, w) can be obtained by calculating cosine similarities. Since the aim of experiments in section 4 was to perform document level sentiment analysis and observations on word dissimilarities is a consequence of this task, extensive experiments validating word antonym pairs have not been performed. This is left for future work and only a qualitative analysis of learned word embeddings is discussed here.

Data Set	Method	Avg Precision	Avg AUC
Yelp	SWESA (LSA)	78.09±2.8	86.06±2.4
	SWESA (re-w2v)	**78.35±4.6**	**86.93±3.5**
	TS (LSA)	76.27±3.0	83.05±5.0
	TS (re-w2v)	65.22±4.4	69.08±3.5
	Naive Bayes	57.07±3.3	61.16±4.5
	RNTN (Pre-tr)	83.31±1.1	-
	RNTN (retrained)	51.15±4.3	-
	CNN-S (Pre-tr)	86.45±0.8	-
	CNN-NS (Pre-tr)	87.98±0.5	-
	CNN-S (re-w2v)	76.89±2.5	-
	CNN-NS (re-w2v)	78.10±1.5	-
	CNN-S (Re-tr)	68.55±1.2	-
	CNN-NS (Re-tr)	72.80±0.5	-
Amazon	SWESA (LSA)	80.31±3.3	87.54±4.2
	SWESA (re-w2v)	**80.36±2.8**	**87.19±3.3**
	TS (LSA)	77.32±4.6	85.00±6.2
	TS (re-w2v)	71.09±6.2	77.09±5.3
	Naive Bayes	72.54±6.4	61.16±4.5
	RNTN (Pre-tr)	82.84±0.6	-
	RNTN (retrained)	49.15±2.1	-
	CNN-S (Pre-tr)	87.08±1.0	-
	CNN-NS (Pre-tr)	87.15±0.8	-
	CNN-S (re-w2v)	78.85±1.2	-
	CNN-NS (re-w2v)	80.40±1.0	-
	CNN-S (Re-tr)	70.15±1.8	-
	CNN-NS (Re-tr)	72.86±2.5	-
IMDB	SWESA (LSA)	76.40±5.2	81.08±7.6
	SWESA (re-w2v)	**77.27±5.4**	**81.04±6.8**
	TS (LSA)	70.36±5.5	77.54±6.8
	TS (re-w2v)	56.87±7.6	59.34±8.9
	Naive Bayes	73.31±5.6	48.40±2.9
	RNTN (Pre-tr)	80.88±0.7	-
	RNTN (retrained)	53.95±1.9	-
	CNN-S (Pre-tr)	85.54±0.6	-
	CNN-NS (Pre-tr)	87.77±0.5	-
	CNN-S (re-w2v)	72.46±2.0	-
	CNN-NS (re-w2v)	75.85±2.1	-
	CNN-S (Re-tr)	62.07±1.5	-
	CNN-NS (Re-tr)	70.16±1.2	-
A-CHESS	**SWESA (LSA)**	**35.80±2.5**	**83.80±3.1**
	SWESA (re-w2v)	35.40±2.0	83.40±2.6
	TS (LSA)	32.20±3.2	**83.80±3.1**
	TS (re-w2v)	23.60±2.4	68.00±1.2
	Naive Bayes	30.30±3.8	45.23±3.3
	RNTN (Pre-tr)	-	-
	RNTN (retrained)	-	-
	CNN-S (Pre-tr)	18.75±3.2	-
	CNN-NS (Pre-tr)	23.02±2.8	-
	CNN-S (re-w2v)	20.12±2.4	-
	CNN-NS (re-w2v)	25.45±1.8	-
	CNN-S (Re-tr)	15.62±2.7	-
	CNN-NS (Re-tr)	18.98±2.2	-

Table 1: This table shows results from a standard sentiment classification task on all four data sets. Results from SWESA are in boldface and results from pre-trained models are in blue. No AUC values are reported for models where classification probabilities of classifiers is not available.

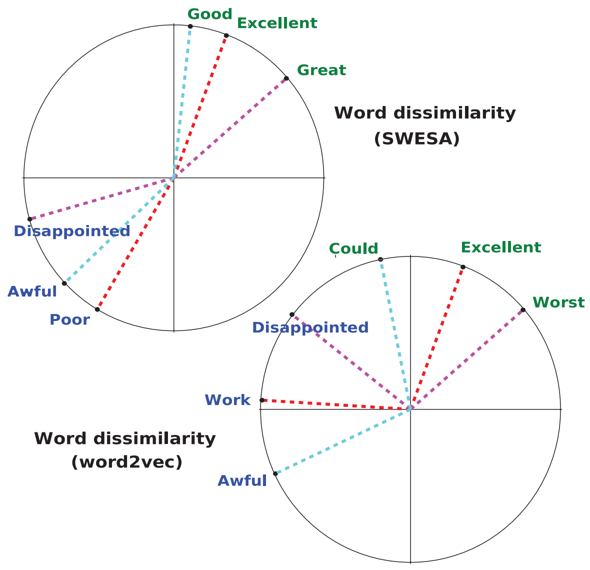

Figure 1: *This figure depicts word embeddings on a unit circle. Most dissimilar word pairs are plotted based on the cosine angle between the respective word embeddings learned via SWESA and word2vec.*

5 Discussions and Conclusions

In this paper we provide a simple optimization based framework, called SWESA that jointly learns word embeddings and a classifier for the task of sentiment analysis. Through extensive experimentation we show that SWESA outperforms both non-neural network algorithms (naive Bayes, LSA) as well as state-of-the-art neural network algorithms based on CNNs and RNNs. As a byproduct of our optimization formulation we show that the word embeddings learned by SWESA are polarity aware and perform very well on antonym tasks, even without being explicitly trained for such tasks. Particularly, strongly polarized word embeddings are easily distinguished. This results indicates that when using finer grained sentiment labels, word polarized along other scales such as dimension, etc can be determined. Our contributions strongly emphasize the point that on domains where data is limited, i.e few thousands of points, but sentiment rich, data size is not a handicap and simple algorithms can do a better task than more involved neural network based algorithm. In the future, we shall investigate modifications of our framework for tasks other than sentiment analysis.

References

[Aga et al.2016] Rosa Tsegaye Aga, Lucas Drumond, Christian Wartena, and Lars Schmidt-Thieme. 2016. Integrating distributional and lexical information for semantic classification of words using mrmf. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2708–2717.

[Blei et al.2003] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

[Dumais2004] Susan T Dumais. 2004. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230.

[Jain et al.2013] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. 2013. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674. ACM.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

[Kotzias et al.2015] Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606. ACM.

[Labutov and Lipson2013] Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *ACL (2)*, pages 489–493.

[Le and Mikolov2014] Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

[Lin et al.2002] Yi Lin, Yoonkyung Lee, and Grace Wahba. 2002. Support vector machines for classification in nonstandard situations. *Machine learning*, 46(1-3):191–202.

[Maas et al.2011] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

[Mairal et al.2009] Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R Bach. 2009. Supervised dictionary learning. In *Advances in neural information processing systems*, pages 1033–1040.

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

[Mikolov et al.2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Mohr et al.2013] David C Mohr, Michelle Nicole Burns, Stephen M Schueller, Gregory Clarke, and Michael Klinkman. 2013. Behavioral intervention technologies: evidence review and recommendations for future research in mental health. *General hospital psychiatry*, 35(4):332–338.

[Moore et al.2011] Brent A Moore, Tera Fazzino, Brian Garnet, Christopher J Cutter, and Declan T Barry. 2011. Computer-based interventions for drug use disorders: a systematic review. *Journal of substance abuse treatment*, 40(3):215–223.

[Pang and Lee2005] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

[Pennebaker et al.2001] James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

[Rakhlin et al.2011] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. 2011. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*.

[Socher et al.2011] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

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

[Tang et al.2014] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565.

[Thelwall et al.2010] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the Association for Information Science and Technology*, 61(12):2544–2558.

Appendix A: Differences between SDL and SWESA.

The significant differences between SDL and SWESA are i) Input to SDL is a labeled dataset, with each data point already represented as a vector. This allows for a definition of reconstruction error within algorithms designed for SDL. In contrast, SWESA has labeled unstructured data without a direct vector representation, and the aim is to learn vector representations for such data. As a result the notion of reconstruction error used in SDL does not apply to SWESA and hence the optimization formulation used is significantly different from the one used in SDL. ii) In SDL sparse encoding of each data point is to be learned, whereas in SWESA this sparse encoding is known and is proportional to the number of times a word appears in the document. (iii) Finally, in SDL the classifier is a high-dimensional vector that acts on the latent codes.