CS 229 Project: Using Vector Representations to Augment Sentiment Analysis Training Data

Andrew McLeod*, Lucas Peeters†

Abstract

While the accuracy of supervised sentiment classification algorithms has steadily increased in recent years, acquiring new human-labeled sentiment data remains expensive. In this paper, we explore the effectiveness of increasing the training data set size for sentiment classification algorithms by adding unlabeled phrases whose sentiments are inferred by their proximity to labeled training phrases. This study was carried out in the context of two softmax classifiers, each trained on movie review phrases with sentiments labeled between 1 (very negative) and 5 (very positive), using pre-trained vectors to represent individual words. We find that augmenting the training data of these softmax classifiers can improve the classification accuracy of one of our models by up to 5%, while the other neither improves nor deteriorates.

Introduction

Sentiment analysis is an active field of machine learning research in which the goal is to find the opinion or emotion expressed by a text with regard to a specific entity [2]. This analysis can be carried out at the level of individual words, whereby one associates a sentiment to each word separately, or at the level of full sentences or even texts. Methods aimed at performing the task of assigning sentiments to full sentences will then commonly contain two key components: a way to represent and predict the sentiments of individual words, and a way to represent and predict the sentiments of phrases using these word representations.

In this project, we set out to improve the predictions made by these types of sentiment classifiers in situations where the amount of labeled sentiment data is limited. This is attempted by representing words and phrases as vectors and using the notion of distance in these vector spaces to infer the sentiments of the nearest unlabeled neighbors of the training phrases; these newly labeled phrases can thereby be added to the original training data. We tested this idea on two classifiers—one in which the vectors representing phrases were constructed by averaging the vectors corresponding the phrase's constituent words, and another in which these word vectors were concatenated.

Data

Both of our classifiers were trained and tested on the Stanford Sentiment Treebank (SST), a database of movie review phrases. This database is comprised of 11,855 full sentences parsed into a total of 215,154 unique phrases, all of which have been annotated by three human judges with a sentiment ranging from 0 to 1 [3]. Following previous work on this data set, we discretize the sentiment into five evenly spaced bins, where 1 is very negative and 5 is very positive. The data is split into 8,544 training, 1,101 development, and 2,210 test sentences—however, due to the fact that some phrases are found in multiple sentences, individual phrases can be found in one, two, or all three splits. For the same reason, some phrases are repeated within the same split; accordingly, accuracies can either be reported on the full (redundant) splits, or on just the unique phrases occurring in them, leading to very different reported numbers. Unless otherwise noted, the accuracies reported in this paper were computed using the redundant splits so that direct comparison could be made with the results reported in [3].

We chose to represent the words within these phrases using vectors trained by GloVe [4], an unsupervised learning algorithm for obtaining such vectors. GloVe is able to capture significant semantic and linguistic relationships—for instance, the difference vector between 'man' and 'woman' is found to be approximately parallel to the difference vector between 'king' and 'queen'—making it well-suited to our task. The vectors we used were pulled from a publicly available set of fifty-dimensional vectors trained on Wikipedia articles.

*ajmcleod@stanford.edu
†lpeeters@stanford.edu

Methods

The softmax classifiers

Both of our classifiers are trained using a softmax training algorithm. Denoting our phrase (or word) vectors as x_k, we optimize θ_l for each sentiment bin $l \in \{0, 1, 2, 3, 4\}$ by maximizing our objective function

$$J(\theta) = -\frac{1}{m} \sum_k \sum_l \left(\mathbb{1}(s_k = l) \log \left(\frac{\exp(\theta_l \cdot x_k)}{\sum_m \exp(\theta_m \cdot x_k)} \right) \right) + \lambda |\theta|^2 \quad (1)$$

where the k sum runs over all training phrases, and s_k is the known sentiment for each training phrase x_k. The regularization term $\lambda|\theta|^2$ is proportional to the square of the Euclidean norm of θ, defined as the vector formed by concatenating the five θ_l vectors. We carried out this optimization using the Adaptive Gradient Algorithm (AdaGrad) [1], which reduces the step size of gradient descent for a given parameter the more that parameter is updated. Convergence is assumed to be attained when the Euclidean norm of the change in θ after one batch step is smaller than a user-defined fraction of the norm of θ itself. Once our θ_l are trained, the predicted sentiment $\tilde{s}_k$ of the vector x_k is

$$\tilde{s}_k = \underset{l}{\text{argmax}} \ \exp(\theta_l \cdot x_k) \quad (2)$$

where l again runs over our five sentiment bins.

While the pre-trained GloVe vectors already capture some of the semantic relations between the words occurring in the SST, they are not optimized for this particular task. Thus, we initialize our word vectors to their pre-trained GloVe values (after normalization), but then adjust them through backpropagation. That is, we maximize our objective function with respect to these word vectors as well as θ, thereby training both sets of vectors simultaneously. These word vector optimizations were also carried out using AdaGrad.

Word Vector Averaging Model

Our first classifier constructs phrase vectors by averaging over all word vectors in a phrase. Phrases are thereby represented in the same fifty-dimensional space as individual words, and both phrases and words are used to train a single set of model parameters θ_l. Word vector backpropagation is only carried out when training on words (not on phrases), and words that are found in the SST but not in the GloVe database are initialized to a random unit vector. Training was carried out both on the full SST training set, and on restricted training sets with sizes logarithmically distributed between 1 and 1000. When training on these restricted training sets, we augment the training data with the k nearest neighbors of each training phrase (pulled from the remainder of the training set), labeling these neighbors with the same sentiment as the original training phrase.

Word Vector Concatenation Model

Conversely, our second classifier represents phrases of different length in different spaces by setting phrase vectors equal to the concatenation of their constituent word vectors. Due to the different resultant vector lengths, a separate set of softmax parameters θ_l is trained on the phrases of each length. This is done by first training the phrase length-one softmax using word vector backpropagation, and then computing the k nearest neighbors of all word vectors in our current (possibly restricted) training set (where k may be 0). A softmax is then trained on each set of phrases with length greater than one as well as their nearest neighbors, where the nearest neighbors of a phrase are constructed by replacing each word in the phrase with its k nearest neighbors (never replacing more than one word at a time). This gives us a $(k \times L)$-fold increase in our training set size for phrases of length L.

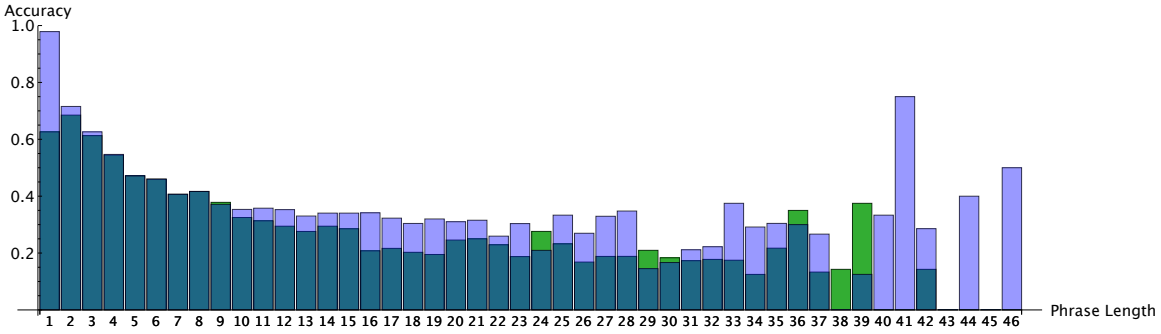

Figure 1: Comparison of the accuracy achieved by our vector averaging and vector concatenation approaches as a function of phrase length (without adding nearest neighbors). Green represents the vector averaging model, blue the vector concatenation model, and dark turquoise their overlap.

Base Model Performance

Before testing the effect of augmenting our training data, we ran both classifiers on the full SST training data to compare our baseline results to previously reported accuracies achieved on this data set. This comparison is found in Table 1. Although neither of our models is able to match the best classifier reported in [3], our concatenated vector model did outperform all non-neural net models reported therein (including Naive Bayes and SVMs). Figure 1 compares the vector averaged and concatenated vector models for each phrase length separately. We see there that the concatenated vector model mainly attains higher accuracy on the phrases of length 10 to 20. For very long phrase length (beyond 30), the number of training phrases is limited and thus the statistics are not very reliable.

Model	Accuracy
Vector Averaged	68.8%
Concatenated Vector	77.3%
Recursive Neural Tensor Network	80.7%

Table 1: The accuracy achieved by our two classifiers when trained on the full SST training set without adding nearest neighbors, compared to the highest accuracy reported in [3], which was attained using a Recursive Neural Tensor Network.

.

Nearest Neighbor Results

The effects of adding nearest neighbors to the training set of our two classifiers was tested by running a large number of trials with variable training set sizes and number of nearest neighbors. The restricted training set was randomly selected from the SST training phrases for each trial, and the phrases within it were required to be unique.

Word Vector Averaging Model

The accuracy achieved by our vector averaging model as a function of training set size is shown for 0, 5, and 10 nearest neighbors in Figure 2. Gains of approximately 2% are achieved by adding 5 nearest neighbors, but only marginal improvements are made by going from 5 to 10 nearest neighbors. Note that these accuracies were achieved by testing on only the unique phrases in the relevant data split, not on the full (redundant) set; the 68.8% reported in Figure 1 (on the redundant set) corresponds to 50.1% on the set of unique phrases. The relative accuracy for 0 through 10 nearest neighbors is also plotted in Figure 3. We see there that our classifiers with augmented training sets do up to 5% better than our baseline ($k = 0$) model for training sets with 100 or fewer training examples, but that this improvement is mitigated to below 2% once training set sizes grow to around 1000.

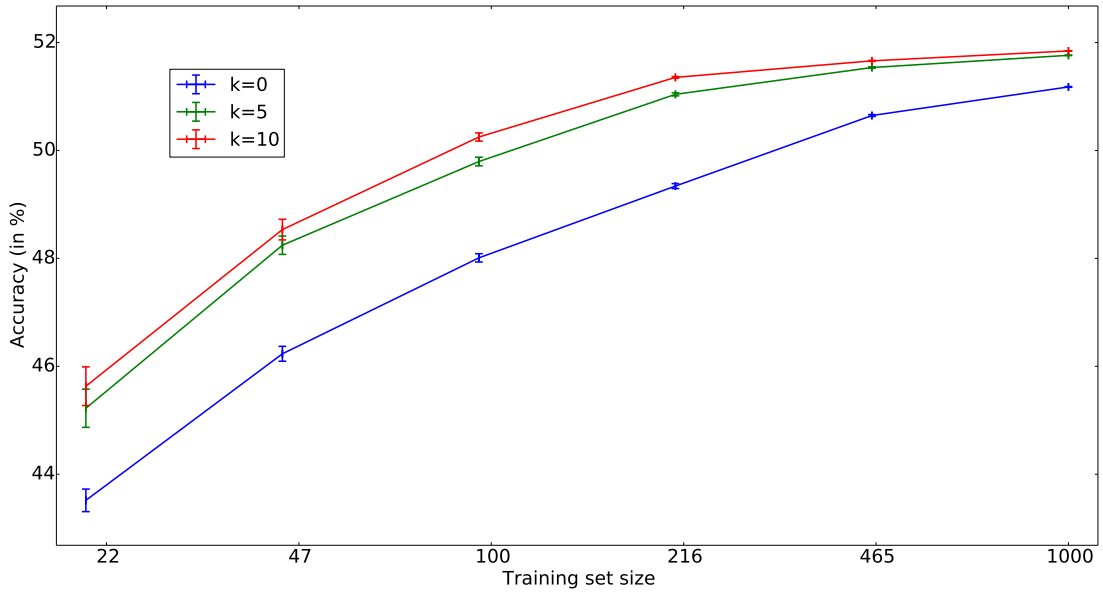

Figure 2: Accuracy attained by the vector averaged model on the unique phrases in the relevant data split as a function of training set size, for $k = 0, 5$ and 10 nearest neighbors. The plotted accuracies represent values averaged over 300 trials, and the error bars denote the standard error of the mean of these trials. Note that the obtained accuracies are obtained using unique phrases, which severely influences the accuracy, as discussed in the text.

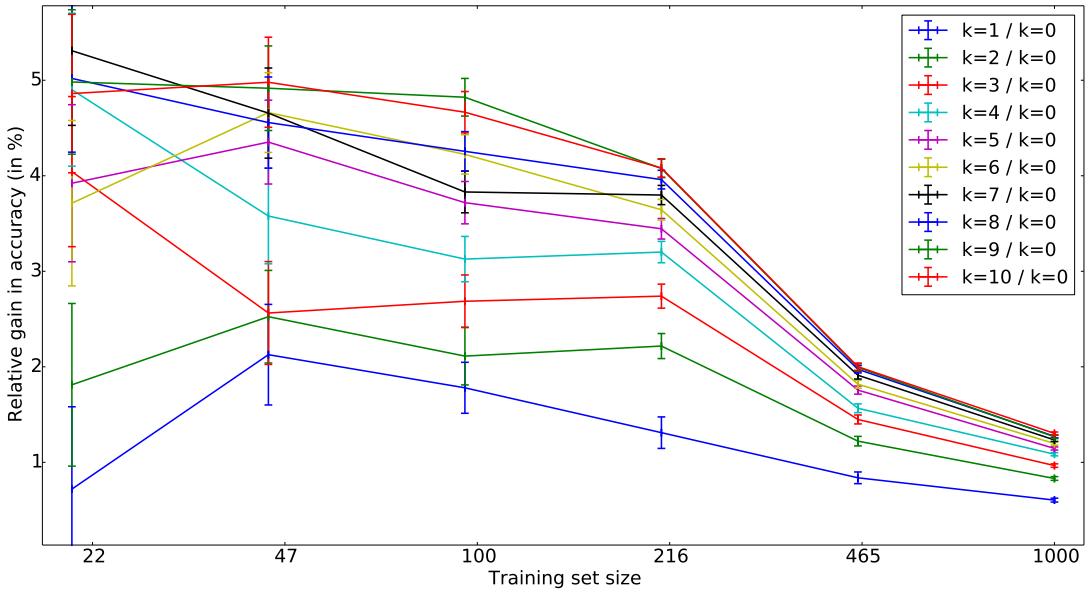

Figure 3: Relative gain in accuracy as a function of training set size for $k = 0$ through $k = 10$. Plotted values were averaged over 300 runs, and the error bars denote the standard error of the mean.

Word Vector Concatenation Model

A similar set of trials was run for our vector concatenation model. However, unlike the vector averaged model, no statistically significant increase or decrease in accuracy was observed.

Improving the Sentiment Classification of Nearest Neighbors

By adding the k nearest neighbors to our training sets in the above manner, we increase our training set size by a factor of k (or more, in the concatenated model). The reason our gains remain limited despite this k-fold increase is that phrases are only loosely clustered by sentiment; assigning the nearest neighbor of each phrase in the SST training set (using a Euclidean metric) only achieves an accuracy of 35.0%. To improve on our above results, we therefore need to improve our ability to predict nearest neighbor sentiments. Our first attempt to do this was to compare the confusion matrices of nearest neighbor assignments for different metrics on this space. Some of the results of this comparison are found in Table 2. Given the accuracy achieved by each metric for each sentiment label, we can maximize our overall accuracy by choosing the best-performing metric as a function of training phrase sentiment. However, doing this only boosts the overall nearest neighbor sentiment prediction accuracy to 36.0%. A second approach consists of training a new set of five softmax classifiers to predict the sentiments of nearest neighbors—one for each sentiment label (of the labeled training phrase). The input used to train these classifiers was a concatenation of the original training vector and the neighboring vector. Training these classifiers on all phrases in the SST training split using the five nearest neighbors of each phrase, and testing on the five nearest neighbors of phrases in the SST development split, we achieve an overall nearest-neighbor classification accuracy of 62%.

Sentiment	L_2^{NN}	L_1^{NN}	L_∞^{NN}	Training Set Fraction
5	0.046	0.047	0.048	0.0490
4	0.176	0.178	0.190	0.185
3	0.509	0.513	0.508	0.509
2	0.210	0.204	0.199	0.203
1	0.059	0.056	0.055	0.055

Table 2: The accuracies achieved by the L_1, L_2, and L_∞ norms when assigning the same sentiment to the nearest neighbor of vectors in the SST training set. These accuracies can be compared to the fraction of the training set associated with each sentiment label.

Future Work

There are a few directions in which this work can be taken. A new set of trials should be run using our recently developed nearest neighbor sentiment classifiers in conjunction with our vector averaged model to see if perfor-

mance improves. We can also try adding different types of nearest neighbors to our concatenated vector model, for instance by selecting nearest neighbor phrases from the remainder of the SST training data like is done in the vector averaged model. Finally, this line of research into augmented training data sets can be continued on to neural net models, which will increase the overall accuracy of our classifiers.

Conclusions

We have shown that, for small training set sizes, the use of nearest-neighbor vectors allows for an increase in phrase sentiment classification accuracy by several percent with respect to our baseline vector averaging method. We did not find that this result extended to our vector concatenation method, at least for the types of nearest neighbors constructed in this model. Without adding nearest neighbors, our concatenation model outperforms all reported methods on this dataset except for neural network methods (notably the Recursive Neural Tensor Network, which achieves 80.7%) [3]. Whether or not it would be possible to increase the accuracy of neural network methods by augmenting their training data with nearest neighbors remains a question for future work.

Acknowledgments

We would like to acknowledge Richard Socher for advising us on this project.

References and Notes

[1] J. Duchi *et al.*, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*, JMLR **12**, p. 2121-2159 (2011).

[2] R. Feldman, Comm. ACM **56** (4), p. 82-89 (2013).

[3] R. Socher *et al.*, *Reasoning With Neural Tensor Networks for Knowledge Base Completion*, Adv. NIPS 26, 2013.

[4] J. Pennington *et al.*, *GloVe: Global Vectors for Word Representation*, Proc. EMNLP, 2014.