A BIGRAM EXTENSION TO WORD VECTOR REPRESENTATION

ADRIAN SANBORN AND JACEK SKRYZALIN

1. Background

GloVe is an algorithm which associates a vector to each word such that the dot product of two words corresponds to the likelihood they appear together in a large corpus ([PSM14]). GloVe vectors achieve state-of-the-art performance on word analogy tasks ($v(\text{king}) - v(\text{man}) + v(\text{woman}) \approx v(\text{queen})$), but they are limited to capturing meanings of individual words. In our project, we develop "biGloVe," a version of GloVe that learns vector representations of bigrams. Using the full English Wikipedia text as our training corpus, we compute 1.2 million bigram vectors in 150 dimensions.

To evaluate the quality of our biGloVe vectors, we apply them to two machine learning tasks. The first task is a 2012 SemEval challenge where one must determine the semantic similarity of two sentences or phrases. We used logistic regression using as features the cosine similarity of the average sentence (bi)GloVe vectors and found slightly better performance in one challenge when GloVe and biGlove were combined, but generally, the usage of biGloVe vectors did not increase performance. Second, we applied biGloVe vectors to classify the sentiment of movie reviews, training with naive Bayes using bag-of-words, SVMs, and random forests. We found that naive Bayes or an SVM with GloVe vectors performed the best.

Applications of biGloVe vectors were hindered by insufficient bigram coverage, despite training 1.2 million vectors. At the same time, examination of nearest neighbors revealed that biGloVe vectors were indeed capturing semantic relationships unique to bigrams, suggesting that the method has promise. Training new vectors on a much larger corpus such as Common Crawl is likely to improve performance of biGloVe vectors in tasks.

2. Bigram Word Vectors

The GloVe Algorithm: GloVe associates to each word w a vector $v(w) \in \mathbb{R}^n$, where typically n takes values between 50 and 300. This collection of vectors is trained first by computing X_{ij}, the number of times word i occurs in the context of word j, within some large corpus of text C (we used the English Wikipedia):

$$X_{ij} = \sum_{c,d \in [1,m]} \frac{\mathbb{1}\{C[c] = w_i\} \cdot \mathbb{1}\{C[d] = w_j\} \cdot \mathbb{1}\{0 < |c-d| \leq 15\}}{|c-d|},$$

where $C[k]$ denotes the kth word in C, m is the number of words in C, and w_i represents the ith word in the lexicon obtained by collecting all words which occur in C. Note that the word $C[k]$ cannot occur in the context of itself, that we consider the context of a word to be all words that appear at most 15 places before or after the word in question, and we weight contextuality by the inverse distance between the words in the article.

Let $w_i \in W$ be the ith word in the set of words W. GloVe uses gradient descent to minimize over all possible assignments of $v(w_i), \tilde{v}(w_i) \in \mathbb{R}^n$ and $b_i, \tilde{b}_j \in \mathbb{R}$ the quantity

$$J = \sum_{i,j=1}^{|W|} {X_{ij}}^{3/4} \left(v(w_i) \cdot \tilde{v}(w_j) + b_i + \tilde{b}_j - \log X_{ij} \right)^2 .$$

biGloVe: We develop a technique called biGloVe (**bi**gram **GloVe**). biGloVe associates to each *bigram* b a vector $v(b) \in \mathbb{R}^n$ (we choose $n = 150$ for this project). We hypothesize that by associating a vector to each *bigram* we can capture new bigram-specific meanings.

To train biGloVe, we download all articles in English Wikipedia, removed all non-article content (i.e., pictures, urls, captions, metadata), converted all letters to lowercase, and removed all numbers. With some experimentation, we noticed that the bigram frequency was strongly biased towards bigrams containing one significant word (e.g., "removal", "tendency") and one insignificant word (e.g., "the", "of"). In order to increase the information density of our bigrams, we removed 68 common stop words which do not add to the meaning of a sentence. Finally, we trained biGloVe via the same gradient descent algorithm as GloVe, except that the cooccurrence X_{ij} is calculated via:

$$X_{ij} = \sum_{c,d \in [1,m-1]} \frac{\mathbb{1}\{C[c] = b_i\} \cdot \mathbb{1}\{C[d] = b_j\} \cdot \mathbb{1}\{0 < |c-d| \leq 7\}}{|c-d|},$$

where $C[k]$ now denotes the kth bigram in C (which consists of the kth and $(k+1)$th word in C), and where b_i represents the ith bigram in the lexicon obtained by collecting all bigrams which occur in C. We performed fifty iterations of stochastic gradient descent, which took around 40 hours on a 16 CPU node with 64GB memory.

Bigram Vector Space. We trained vectors for any bigram occurring more than 150 times in the corpus, resulting in 1,240,128 vectors. To probe the semantic structure captured in these vectors, we computed the ten nearest neighbors for several hundred of the most frequent bigrams, measured by cosine similarity of the corresponding vectors, $v_1 \cdot v_2 / \|v_1\| \|v_2\|$. We observed that many neighbors captured new bigram-specific meaning. For example, nearest neighbors of "united nations" included "security council", "secretary general", "human rights"; nearest neighbors of "st louis" included "kansas city" and "saint louis"; and nearest neighbors of "major league" included "professional baseball" and "baseball player".

3. Semantic Similarity Prediction

We use GloVe and biGloVe to produce feature vectors for a semantic similarity task. The data, downloaded from `http://ixa2.si.ehu.es/sts/data/trial.tgz`, was used for the SemEval-2012 task 17 challenge. The challenge contains 750 pairs of sentences in each of three groups, annotated with a numerical label ranging from 0, indicating that the sentences have no overlapping meaning, to 5, indicating that the sentences have the exact same meaning.

We used logistic regression using a small feature vector, minimizing over all θ and c the quantity $\frac{1}{2}\theta^\top\theta + C\sum_{i=1}^{n} \log(\exp(-y_i(X_i^\top\theta + c)) + 1)$ where we set $C = 10{,}000$ for weak regularization. We considered four features: the lengths of the two sentences, and the cosine similarity between the two sentences using the GloVe/biGlove vectors obtained by averaging the GloVe/biGloVe vectors of all words/bigrams in the sentences. We tested three versions: one just using GloVe, one just using biGloVe, and one using both GloVe and biGloVe. The percentage of correct similarity rankings is given below:

	Group 1		Group 2		Group 3	
	Train success	Test success	Train success	Test success	Train success	Test success
GLV	0.347	0.306	0.713	**<u>0.716</u>**	0.684	**<u>0.403</u>**
bGLV	0.220	0.189	0.289	0.295	0.620	0.085
GLV+bGLV	0.325	**<u>0.341</u>**	0.736	**<u>0.718</u>**	0.702	0.361

Figure 1. Success rates using various algorithms and feature vectors for semantic similarity

We find that biGloVe vectors alone perform poorly, suggesting insufficient coverage of bigram vocabulary. However, adding biGloVe to GloVe did improve performance slightly for one group of sentences.

4. Sentiment Analysis

The Data: The data, downloaded from `http://nlp.stanford.edu/sentiment/` ([SPW+13]), consists of 11855 movie reviews with a train/dev/test split of 8544/2210/1101. Each movie review is assigned a score from 0 to 1. The interval $[0, 1]$ is split evenly into fifths, and each fifth is assigned a sentiment (i.e., very-negative, negative, neutral, positive, and very-positive). It is our goal to correctly categorize each review into its designated category. We record the percentage of reviews that have been classified correctly.

We also consider a two-category system in which we consider reviews with assigned scores in the interval $[0, 0.4]$ to be "negative" and those with assigned scores in the range $[0.6, 1]$ to be "positive". We apply all algorithms to both the five-category and two-category schemes; algorithmic adaptations necessitated by having a non-binary classification scheme are detailed below.

Before applying all algorithms, we filter all text as done in bigram vector training. We consider three featurization schemes: bag-of-words, GloVe, and biGloVe. The bag-of-words (BOW) model associates to each review $\mathtt{r}$ a sparse vector $\phi(\mathtt{r})$ indexed on the words in the article. The $\mathtt{w}$th entry of $\phi(\mathtt{r})$ is the number of times the word $\mathtt{w}$ appears in $\mathtt{r}$. To construct GloVe and biGloVe feature vectors, we average all word / bigram vectors corresponding to words in the review.

Naive Bayes: We use a multinomial naive Bayes (NB) model with Laplace smoothing. We use the bag-of-words feature vectors only with this algorithm. To train the model with training set T_{train}, we calculate the values

$$p(\text{word } x \mid \text{category} = c) = \frac{1 + \sum_{k=1}^{|T_{train}|} \mathbb{1}\{c = \text{category}(\mathtt{r}_k)\} \cdot \phi(\mathtt{r}_k)[x]}{|\text{vocabulary}| + \sum_{k=1}^{|T_{train}|} \mathbb{1}\{c = \text{category}(\mathtt{r}_k)\} \cdot n_k}$$

$$p(\text{category } c) = \frac{\sum_{k=1}^{|T_{train}|} \mathbb{1}\{c = \text{category}(\mathtt{r}_k)\}}{|T_{train}|},$$

where $\mathtt{r}_k$ denotes the kth review in T_{train}, category$(\mathtt{r})$ gives the category associated to review $\mathtt{r}$, and n_k gives the number of words in the kth review $\mathtt{r}_k$. Given a test review $\mathtt{r}$ with BOW feature vector $\phi(\mathtt{r})$, we predict the category with the highest posterior probability.

Support Vector Machines: We experimented with a variety of support vector machines (SVMs) and found that a ν-SVM (as constructed in [SSWB00]) with a Gaussian kernel demonstrated optimal performance. The parameter ν is an upper bound on the fraction of training errors and a lower bound on the proportion of support vectors.

Concretely, given a training set T_{train} consisting of pairs $\left(\phi^{(i)}, y^{(i)}\right)$, where $y^{(i)} = \pm 1$, we solve the following optimization problem:

$$\text{maximize} \qquad W(\alpha) = -\frac{1}{2} \sum_{i,j=1}^{|T_{train}|} \alpha_i \alpha_j y^{(i)} y^{(j)} K(\phi^{(i)}, \phi^{(j)})$$

$$\text{subject to} \qquad 0 \leq \frac{1}{\alpha_i} \leq \frac{1}{|T_{train}|} \qquad \sum_{i=1}^{|T_{train}|} \alpha_i y^{(i)} = 0 \qquad \sum_{i=1}^{|T_{train}|} \alpha_i \geq \nu = \begin{cases} 0.5 & \text{bag-of-words used} \\ 0.6 & \text{*GloVe used} \end{cases}$$

where $K(\phi, \psi) = \exp\left[-\gamma \|\phi - \psi\|^2\right]$, where $\gamma = \begin{cases} 0.05 & \text{bag-of-words used} \\ 0.2 & \text{*GloVe used} \end{cases}$.

Given a new feature vector ϕ, we assign the category determined by

$$\text{sign}\left(\sum_{i=1}^{|T_{train}|} \alpha_i y^{(i)} \left(K(\phi, \phi^{(i)}) - \frac{1}{|SV|} \sum_{\phi_{sv} \in SV} K(\phi_{sv}, \phi^{(i)}) \right) \right),$$

where SV is the set of all support vectors.

For the case where we have 5 classes, we use the "one-vs-one" approach: we construct 10 SVMs to separate each pair of classes. Upon receiving a new feature vector ϕ, we test ϕ against each SVM and tally the number of times that ϕ is assigned to each class. We classify ϕ according to the category to which it is most frequently assigned. If "ties" occur, ϕ is assigned the class based on the classification provided by the furthest hyperplane.

Random forests: To train our random forests (RF), we create 250 decision trees (as outlined in [Bre01]). We create the decision trees via the CART algorithm, minimizing Gini impurity. To test our random forest, we output the most common decision of the 250 decision trees.

Results: We provide the computed success rates for the various (learning algorithm, feature vector) pairs. The best test success rates for each classification scheme are bold and underlined.

		Two sentiments		Five sentiments	
		Train success	Test success	Train success	Test success
NB	BOW	0.944	**<u>0.818</u>**	0.807	0.399
SVM	BOW	0.987	0.813	0.955	0.401
	GLV	0.850	0.793	0.741	**<u>0.415</u>**
	bGLV	0.916	0.652	0.842	0.295
	GLV+bGLV	0.990	0.763	0.980	0.382
RF	GLV	1	0.769	0.999	0.397
	bGLV	0.932	0.643	0.882	0.317
	GLV+bGLV	1	0.762	0.999	0.403

FIGURE 2. Success rates using various algorithms and feature vectors for sentiment analysis

One of the most important messages gleaned from these results is that naive Bayes, which is both simple and easily implemented, is also highly effective. Also important is that ν-SVMs can improve success rates over regular SVMs: using a ν-SVM produced success rates around 4% higher than those of a standard SVM using GloVe features (using ν-SVMs with the GloVe feature model produced higher success rates than naive Bayes!), and success rates around 17% higher than those of a standard SVM using BOW features (standard SVM results not shown). It can be concluded that ν-SVMs allow us to harness the power of SVMs when the data is sparse and/or very high dimensional.

Interestingly, we found that neural networks generally produced relatively low success rates (success rates not shown here). This comes as a surprise: other experiments not reported here involving the 20 Newsgroups dataset showed that using neural networks with the GloVe feature model can rival naive Bayes on document classification tasks.

5. Discussion

We effectively implemented biGloVe and trained bigram vectors on the full English Wikipedia text, and the resulting vectors captured new bigram-specific meaning. However, when our biGloVe vectors were used in semantic similarity and sentiment analysis tasks, they performed poorly on their own and did not substantially improve performance in combination with GloVe vectors.

Because the number of bigrams needed is on the order of the square of the English vocabulary size, achieving ample vocabulary coverage is a particular challenge in training biGloVe vectors. Upon closer inspection, despite training over 1.2 million bigrams, we found that algorithm performance was hindered by low vocabulary coverage of bigrams. For example, over 50% of bigrams in the sentiment analysis movie reviews were not in our biGloVe dictionary. Furthermore, expanding the vocabulary by lowering the corpus co-occurrence threshold is not sufficient to achieve significantly greater coverage: training a vector for any bigram occurring more than 50 times in the Wikipedia corpus (rather than

150) would produce about 3.7 million vectors but only reduce the percentage of missing bigrams from 59% to 49% (see figure).

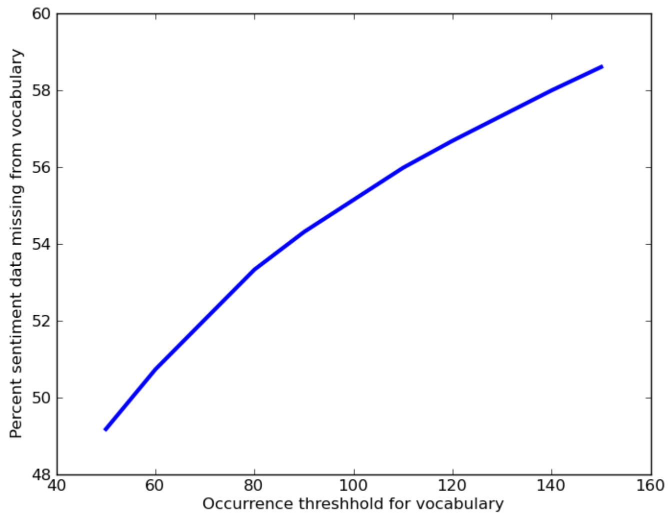

Figure 3. Bigram coverage of movie review text as a function of training corpus occurrence threshold

By comparing the movie review text and biGloVe vocabulary, it also became evident that some not-uncommon colloquial bigrams were missing from the vocabulary. It appears that the formal nature of the Wikipedia text also biased the biGloVe vocabulary away from the text in our applications.

Although our implementation of biGloVe didn't perform as well as expected, we nonetheless feel that the algorithm is sound, and that examination of biGloVe neighbors suggests that the model still has potential. As such, as a next step, we would like to train biGloVe on a much larger corpus, ideally the combination of Wikipedia, Common Crawl, and Gigaword 5. We estimate that a vocabulary of around 10 million biGloVe vectors should provide sufficient coverage for our applications, and we would expect then to see improved performance when biGloVe is combined with GloVe. Additionally, the semantic space of bigrams is more complex than the space of individual words, and we expect that biGloVe performance will continue to improve as the vector dimension is increased, at least to 300 dimensions, and perhaps to 1000 dimensions.

Once second-generation biGloVe vectors have been trained, we would also like to experiment with other statistical methods for using the vectors in applications. For example, instead of averaging all word vectors in order to create a vector for the entire review, it would be interesting to instead analyze the *distribution* of all word vectors in a review. Using more advanced statistical techniques, one might then be able to correlate certain distributions of word vectors with certain sentiment classes.

References

[Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, 2014.

[SPW$^+$13] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. 2013.

[SSWB00] Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. *Neural Comput.*, 12(5):1207–1245, May 2000.

E-mail address: `asanborn@stanford.edu`

E-mail address: `jskryzal@stanford.edu`