Predicting Verb Syntax from Distributional Information: An Analysis Using GLoVe and VerbNet

Kyle Sargent, Melissa Kline, Idan Blank, Evelina Fedorenko

September 8, 2017

Abstract

Using VerbNet frames as class labels, we attempt to predict VerbNet frames for verbs using the corresponding GLoVe vectors of those verbs, in the style of a multi-label classification problem. We then nvestigate the quality of these predictions using the AUROC benchmark, and go on to investigate the dimensionality of the syntax data encoded in GLoVe.

To be added: Clustering verbs in VerbNet and GloVe; clustering frames in VerbNet

1 Introduction

Levin (1993) developed the notion of a verb class. This classification was extended and enriched by Verbnet, a database of about 8000 verb/class instances. The website of VerbNet succintly describes a verb class:

> "Each verb class in VN is completely described by thematic roles, selectional restrictions on the arguments, and frames consisting of a syntactic description and semantic predicates with a temporal function, in a manner similar to the event decomposition of Moens and Steedman (1988)."

We are interested in demonstrating that GLoVe encodes syntax information; thus, the most natural thing to try and predict are the frames a verb fits into. It is worth noting that in VerbNet, whether or not a verb fits in a frame is true or false, with no gray area - in practice, such judgments by humans are much fuzzier.

Any verb in VerbNet belongs to some number of classes, and each class expresses some number of the 291 frames. Suppose we number the frames $f_1 ... f_{291}$. The notion of a "binary frame vector" associated to a verb is as follows: Suppose a verb vhas associated binary frame vector v, then we say $v_i = 1$ if v belongs to frame i in any of its class instantiations, and 0 otherwise.

On the other hand, for most of the verbs in VerbNet, we have GLoVe vectors obtained from crawling Twitter, which are publicly available from Stanford NLP. A GLoVe vector for a word is a 100-dimensional vector of real numbers that encodes its distributional information.

1.1 Summary of the Prediction Task

We can reduce from the problem of recovering verb syntax from GLoVe to learning a good approximate mapping from the 100-dimensional GLoVe space to the 291-dimensional space of binary frame vectors. In the field of machine learning, this is known as a multi-label classification problem.

To avoid over-fitting, we will randomly shuffle the set of 3709 verbs for which we have both GLoVe vectors and frame vectors into a training set of 80% of the verbs, and a validation set of 20% of the verbs.

One of the simplest possible models was chosen for classification: a logistic regression model. Since a logistic regression model is a binary classifier, and we are interested in 291 classification problems, we simply concatenate and separately train 291 independent logistic classifiers to predict the frame vector of a verb from its GLoVe vector.

An advantage of a logistic regression model is that its transparency will allow us to directly examine the weights on each dimension of GLoVe - in short, it may be able to find which dimensions of GLoVe are most diagnostic, for syntax purposes.

To ensure robust predictions, positive training examples in both training and testing sets, and to simplify the problem of disentangling the data into training and testing sets such that this latter condition is satisfied, all frames with 5 or fewer positive training examples have been excluded from our prediction task. Since there are 232 frames with more than 5 positive training examples, we restrict our initial investigation to these frames (for the time being).

1.2 Discussion of Results

Since we are evaluating hundreds of independently trained binary classifiers, the Area Under the Receiver Operating

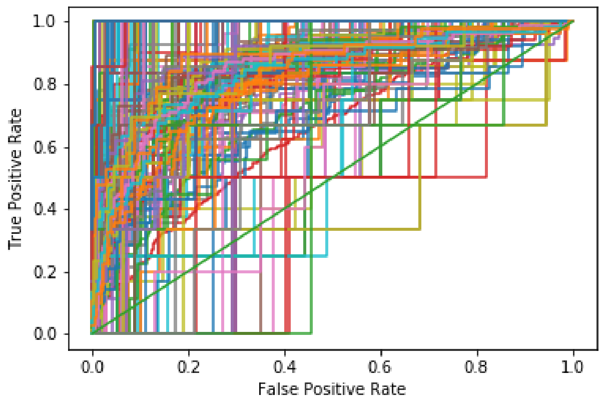

Figure 1: 232 frames with minimum 5 positive training examples, AUC average = .8431

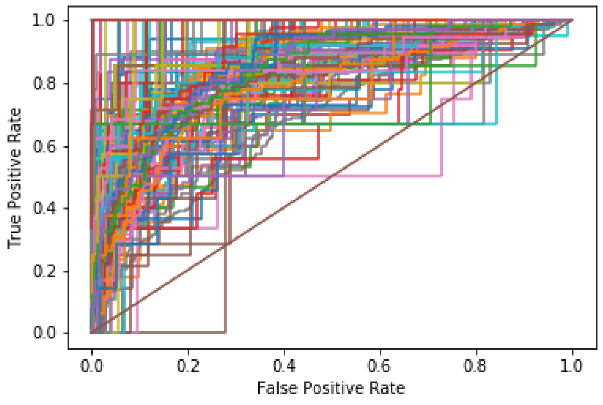

Figure 2: 155 frames with minimum 15 positive training examples, AUC average = .8421

Curve (AUROC) seems a fair benchmark for our model. The receiver operating curve (ROC) of a binary classifier is essentially the true positive rate of the classifier, plotted against its false positive rate. That is, the ROC is the probability it will correctly classify a positive example as a function of the probability it will incorrectly classify a negative example. When these points are joined they form a curve, and the area under it, or AUROC, measures the effectiveness of a binary classifier by giving a number between .5 and 1.0 that measures the strength of the tradeoff between false positives and true positives - at .5, a model is no better than a random classifier. At 1.0, the model classifies all training examples perfectly.

After fitting the 232 logistic regressions on the training data, we now consider their performance on the testing data. We can plot the receiver operating curves atop one another as in figures 1-4. They appear to vary widely, but there is no mistaking that the average predictive strength is well above a random classifier. However, some spuriously trained models do quite poorly - take a look at the curves that fall below the random classifier benchmark line $y = x$ in figures 1-4, but especially in figures 1 and 2.

It is easy to see that, on average, predicting frames from GLoVe vectors using a logistic regression classifier does better than random chance. But there is a lot of variance in how good the model is for different frames. Let's see what happens when we raise the threshold for positive training examples.

It's clear from the above plots that model stability and performance goes up as a function of the number of positive training examples. It is encouraging that the AUC seems to stabilize as well. Consider a scatterplot of AUROC against the number of positive training examples, in figure 5.

As training examples (the X-axis) increases, the spread in AUC tends to decrease, all the while stabilizing at around

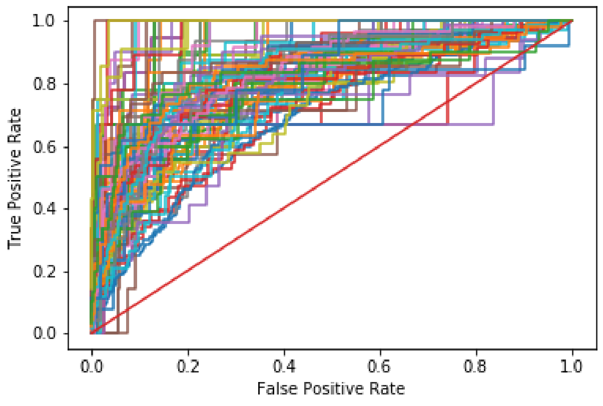

Figure 3: 93 frames with minimum 30 positive training examples, AUC average = .8409

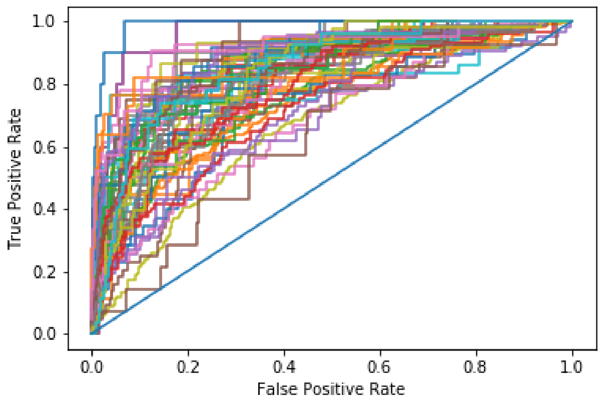

Figure 4: 60 frames with minimum 50 positive training examples, AUC average = .8111

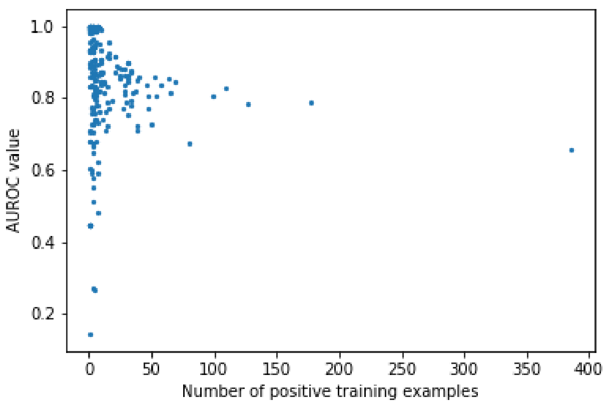

Figure 5:

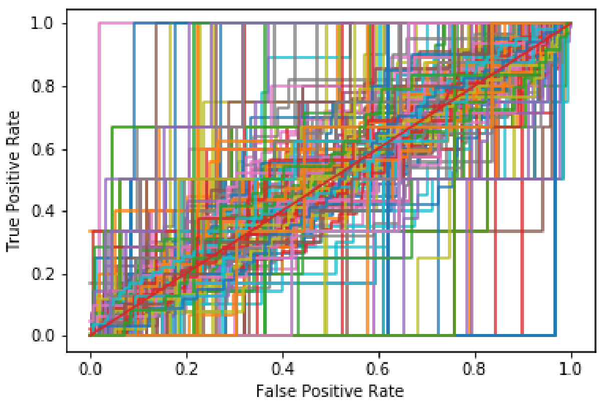

Figure 6:

.75, confirming our hypothesis.

It seems clear that our model is predicting syntax reasonably well from GLoVe vectors. How can we be sure we have not overfitted a model? For one, the training data was thrown away after model fitting, and all the results you see above are on the test set. This alone is reason to believe we have not overfitted to our data.

Suppose we scrambled the assignment of verbs to frames. Our model could still pick up on the distributional information of frames (e.g. this frame occurs 30% of the time, so with a false positive tolerance of 70% or more we should predict it always), but the GLoVe vector observations for a verb would be meaningless. We would expect such a model to do poorly, and this is confirmed by examining figure 6, which is a plot of the AUROC curves for 232 frame classifiers that are trained on the scrambled data.

These models perform no better than random chance. Indeed, in the fashion of figure 5 we can also take each classifier's AUC as a function of the number of positive training

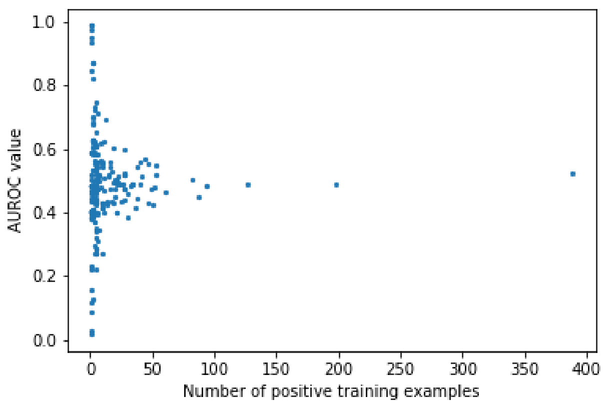

Figure 7:

examples. This is shown in figure 7. In this case it's clear

$$\lim_{\#+\text{training examples}\to\infty} AUC = .5$$

Which is as we expect for a collection of fundamentally random (uninformed) classifiers. This is good because it tells us GLoVe vectors give us frame information when (and only when) this information is correctly paired, which is another confirmation of our hypothesis.

2 Variance in GLoVe Explained by Syntax

It is clear now that the 100-dimensional GLoVe space encodes some syntax information. This leads us to a natural follow-up - are all dimensions of GLoVe equally important in predicting syntax? It may be possible to reduce the number of dimensions significantly via some basis change and projection, and still obtain comparable performance on our prediction task.

Suppose it were possible to project GLoVe onto a new orthonormal basis $b_1 ... b_n$, where the basis vectors were arranged in order of their usefulness in predicting syntax. Then we could use information about how our predictions degraded as a function of the number of basis vectors $n \leq$ 1100 to determine whether the "syntax-space" of GLoVe has inherently lower dimensionality. As it happens, there is a way to get such a basis that is reasonably principled.

2.1 Syntactically Diagnostic Axes of GLoVe

Recall that we are predicting L frame labels from an input space of dimension $D = 100$. For a single label, logistic regression over the 100 dimensional GLoVe set is formulated such that, given n input data $\{\vec{x}\}_{i=1}^{n} \in \mathbb{R}^{100}$ in the glove space and corresponding labels $\{\vec{y}_i\}_{i=1}^{n} \in \{0, 1\}$, we learn a weight vector β_1 and bias β_0 that gives optimal prediction probabilities

$$P(\vec{x}_i \text{ is labeled } 1) = (1 + e^{-(\beta_0 + \vec{x}_i \cdot \beta_1)})^{-1}$$

In other words, it is easy to see that we predict $\vec{x}_i$ to have label 1 if

$$\beta_0 + \beta_1^T x_i > 0$$

And we predict label 0 otherwise, because the probability will be less than $\frac{1}{2}$. But the takeaway here is that each frame has a weight vector β_1 on the space of GLoVe that measures, inuitively, how important each dimension is.

Suppose we ran PCA on the space of weight vectors β_1^i, where the number of data points is precisely the number of frames i, $i \in \{1...L\}$. Mathematically, this gives an orthonormal transformation U such that for any weight vector β_1^i,

$$\beta_1 \approx U^T(U(\beta_1 - \mu)) + \mu$$

Where μ is the mean of the data $\{\beta_1^i\}_{i=1}^L$. Notice that intuitively, this means

$$\begin{aligned} P(\vec{x}_i \text{ is labeled } 1) &= (1 + e^{-(\beta_0 + \vec{x}_i \cdot \beta_1)})^{-1} \\ &\approx (1 + e^{-z})^{-1} \end{aligned}$$

Where we have defined

$$z := \beta_0 + \vec{x}_i \cdot (U^T(U(\beta_1 - \mu)) + \mu)$$

Simply regrouping parentheses, we obtain

$$z := \beta_0 + (\vec{x}_i U^T)(U(\beta_1 - \mu)) + \mu)$$

The key fact is that is a linear function of $\vec{x}_i U^T$, which is in fact a reduced-dimension representation of $\vec{x}_i$. Moreover, this linear function will be implicitly learned via a logistic regression if we transform our input data $\vec{x}_i$ by left-multiplication of U^T, and since our choice of U was to minimize reconstruction loss on the weight vectors β_1, we should be able to get prediction accuracy close to that of when we used our original data set.

As in the previous section, we split our data into 80% training and 20% testing. The difference here is that we first train a logistic regression to learn the weights β_1 and then the optimal transformation $U^T \in \mathbb{R}^{100 \times n}$. Then we transform our data by left-multiplication of U^T, and train another logistic regression with the transformed data and the same labels.

Then we can compare performance of our regression from section 2 on the validation data to the performance of our current regression on the validation data of reduced dimension. Suppose we restrict to the 183 frames with at least 10 positive training examples. Consider the following plot of the average AUROC over all 183 frame classifers as a function of the number of dimensions onto which we have projected GLoVe.

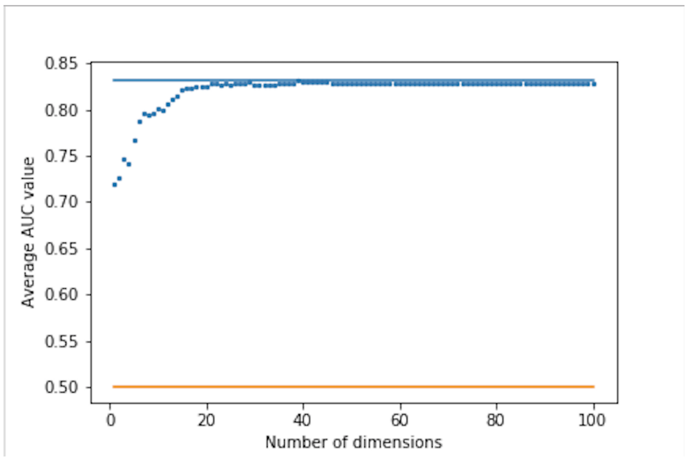

Figure 8: Orange line is performance of the baseline random predictor, blue line is predictor trained on full-dimensional GLoVe data

As you can see, we can project GLoVe onto a space of dimension 50 and perfectly replicate the performance of our baseline predictor trained on the whole GLoVe space. Moreover, projecting GLoVe onto the single most syntactically predictive dimension still yields an average AUC of .713. In short, there is strong evidence to suggest the syntactically diagnostic variance in GLoVe is an extremely small part of the overall variance in the data.