Predicting syntactic sentence frames from GLoVe vectors for verbs

Kyle Sarent[1], Melissa Kline[2], Idan Blank[2], and Evelina Fedorenko[3,4]

[1]Harvard University
[2]Department of Brain & Cognitive Sciences, Massachusetts Institute of Technology
[3]Massachusetts General Hospital
[4]Harvard Medical School

Abstract

IN OUTLINE FORM

(0) While computational models of semantics that are based on purely distributional information have done well (CITES), semantic theories predict a role for frame-level representations, and these may explain why verb semantics are so hard. What are these frames and where do they come from?

(1) Glove predicts frames pretty well!

(2) Ooh :. glove contains much more than just the frames, aka frames explain a tiny amount of GLOVE variance.

(2a) Glove predicts frames, but frames don't predict glove. This is sensible from a linguistic preference.

(3) So, what are human judgments? Some kind of combination of large distributional information (GLOVE) and structured semantic knowledge (VerbNet)? Maybe so! Here is SimVerb, which tells us directly how similar ppl find verbs. Verbnet thinks those same verbs are X similar, while Glove thinks they are Y similar. These 3 similarity matrices compare LIKE SO, suggesting a possible role for explicit frame representations in human language, and that NL researchers should take further looks at VerbNet.

1 Background

1.1 Verbs and argument structure

Nouns are different from verbs Verbs appear in certain frames There are lots of theories as to why this is, or how to describe them, but the basic insight is that clustering by semantics

Class: ROLL-51.3.1	Class: PUSH-12.1.1
Bill **rolled** the door open	Bill **pushed** the door open
The drawer **rolled** open	~~The door **pushed** open~~
~~Bill **rolled** at the door~~*	Bill **pushed** at the door
bounce, float, slide, twist	*jostle, nudge, pull, squeeze*

Figure 1: Two verb classes from VerbNet (Palmer, CITATION), showing a subset of frames that are used to identify the classes, and examples of verb in that class.) No semantic criteria are used to generate the classes, but the verbs nevertheless share aspects of meaning. *Note that this sentence is ungrammatical in the *conative reading* (Bill attempted to roll the door but didn't move it, in parallel with "Bill pushed at the door.") The only grammatical interpretation is the *locative reading* where Bill himself rolls towards the door.

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 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.

1.2 Distributional models of verb semantics

IDAN/KYLE WRITE - can be short, focus on what's different about this - primarily, we treat verbs just like any other words. Describe/cite what Glove is. This is the section to head off objections about sense/POS disambiguation as well - say whether we're using a Glove version that does, acknowledge any limitation, and move on.)

On the other hand, for most of the verbs in VerbNet, we have GLoVe vectors. For this project, I wanted to make use of distributional information from Wikipedia and Twitter - thus, GLoVe vectors from both were concatenated to form a 400-dimensional vector of real numbers that encodes its distributional information.

2 Predicting VerbNet frames from GLoVe, and vice versa

2.1 Summary of the prediction task - VerbNet from GLoVe

We can reduce from the problem of recovering verb syntax from GLoVe to learning a good approximate mapping from the 400-dimensional to the 291-dimensional space of binary frame vectors. In the field of machine learning, this is known as a multi-label classification problem.

Well randomly shuffle the set of verbs for which we have both GLoVe vectors and frame vectors into a training set of 2949 verbs, and a validation set of 737 verbs, i.e. the standard 80/20 training/validation split for most machine learning problems. By learning a mapping from glove vectors to frames on the training data and checking its precision (see part 2) on the validation data, we will attempt to demonstrate that we can in fact successfully recover syntax information from GLoVe.

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 investigation to these frames (for the time being).

2.1.1 Discussion of Results

Since we are evaluating hundreds of independent binary classifiers, the Area Under the Receiver Operating Curve (AUROC) seems a fair benchmark for our model. The receiver operating curve of a binary classifier is essentially the true positive rate of the classifier, plotted against its false positive rate. That 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 below. They appear 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 $y = x$.

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 training ex-

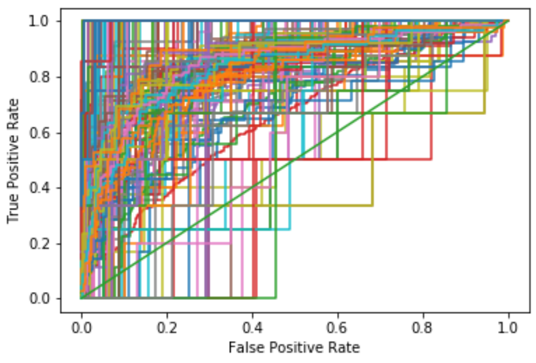

Figure 2: 232 frames with minimum 5 training examples, AUC average = .8431

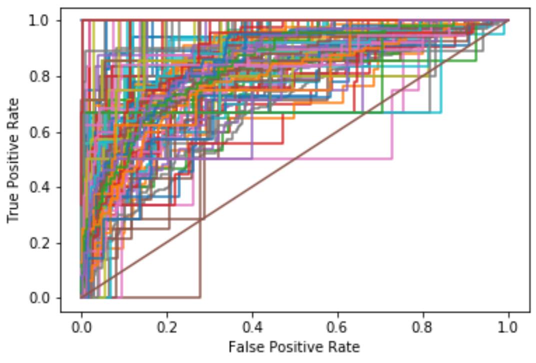

Figure 3: 155 frames with minimum 15 training examples, AUC average = .8421

amples.

It's clear from the above plots that model stability and performance goes up as a function of the number of training examples. What's encouraging is that AUC seems to stabilize as well. Let's take a look at a scatterplot of AUC against the number of training examples.

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

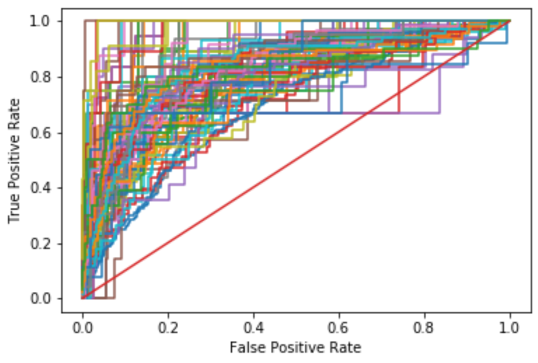

Figure 4: 93 frames with minimum 30 training examples, AUC average = .8409

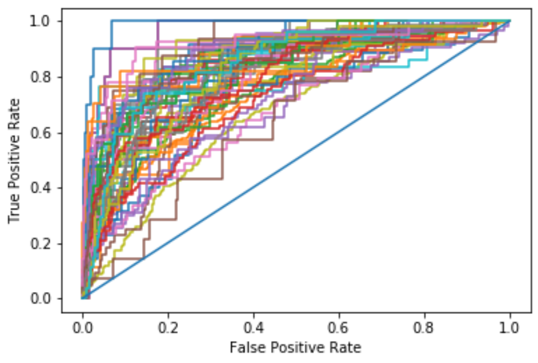

Figure 5: 60 frames with minimum 50 training examples, AUC average = .8111

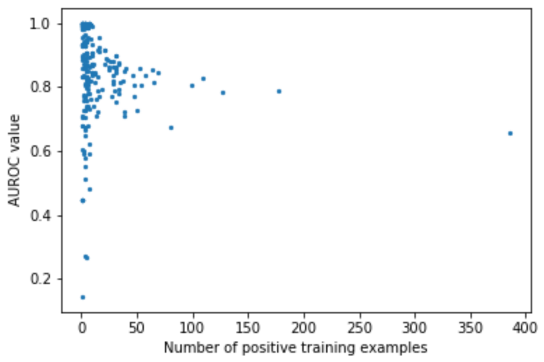

Figure 6:

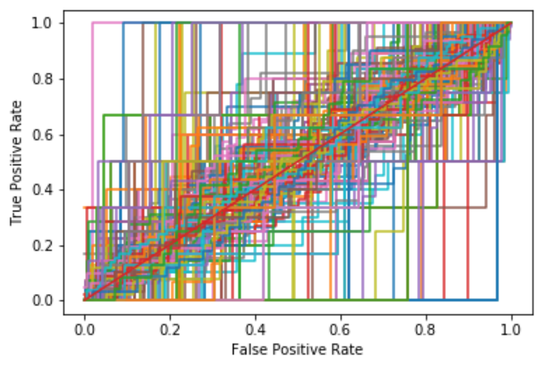

Figure 7:

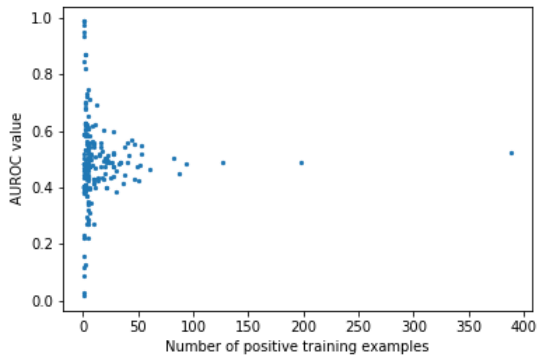

Figure 8:

tolerance of 70% or more we should predict it always), but the GLoVe vector observations for a verb would be meaningless. How does such a model do? Let us examine their collective ROC plots in figure 7.

These models perform no better than random chance. Indeed, it's clear from figure 8 that in this case that

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

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.

2.2 Dimensionality reduction of word embeddings

It is possible to make predictions from our word embeddings that are nearly as good using only a fraction of the dimensionality. Given data which

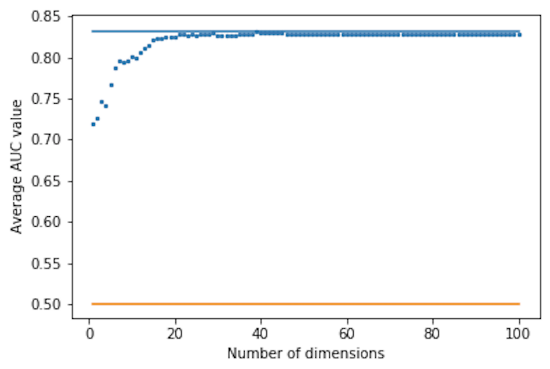

Figure 9: Orange line is performance of the baseline random predictor, blue line is predictor trained on full-dimensional GLoVe data

we have split into 80% training and 20% testing, we can use the training data to discover a transformation designed to preserve syntax information, that maps our word embeddings from $\mathbb{R}^{100}$ to $\mathbb{R}^d$ where $d < 100$ (details in appendix). We then re-train our logistic regressions on the reduced-dimension training data and explore its performance on the reduced-dimension test set.

Suppose we restrict to the 183 frames most represented in our set of verbs, and ask how performance of our frame classifiers degrades as a function of the number of original dimensions of our word embeddings that we use. Consider figure 9, a plot of the average AUROC over the 183 frame classifers as a function of the number of dimensions onto which we have projected our word embeddings.

3 Modeling human similarity judgments from VerbNet and GLoVe

Acknowledgments

The acknowledgments should go immediately before the references. Do not number the acknowledgments section. Do not include this section when submitting your paper for review.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery* 28(1):114–133. https://doi.org/10.1145/322234.322243.

Association for Computing Machinery. 1983. *Computing Reviews* 24(11):503–512.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1–11. https://doi.org/10.18653/v1/P16-1001.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Mary Harper. 2014. Learning from 26 languages: Program management and science in the babel program. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, page 1. http://aclweb.org/anthology/C14-1001.

A Appendix

In this section we describe our method of reverse-PCA for discovering syntactically diagnostic axes of the space of word embeddings.

Consider our prediction task: recall that we are predicting L frame labels from an input space of dimension $D = 100$. For a single label, logistic regression over our data of word embeddings is formulated such that, given n input data $\{\vec{x}\}_{i=1}^{n} \in \mathbb{R}^{100}$ 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 word embeddings 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 from our training set, and then recover the transformation $U^T \in \mathbb{R}^{100 \times n}$ by running PCA on those weights. Then we transform our training AND testing data by left-multiplication of U^T, and train another logistic regression with the transformed data and the same labels.