# Unsupervised Recovery of Tree Metrics from Learned Language Representations

**Benjamin Newman** [1]

## Abstract

Deep contextualized word embedding models likely take advantage of traditional linguistic structure to make predictions, but there are few clear ways to measure what structure these models know. Current supervised approaches might add information that the models don't have access to while unsupervised approaches deal only with whether the models learn specific linguistic phenomena. In this work we attempt to bridge this gap by designing an unsupervised structural probe to using a variant of the EM algorithm. While our probe was not too successful, its failures point to future directions to further develop the idea of an unsupervised probe.

## 1. Introduction

The introduction of deep, contextualized word embedding models, like ELMo, BERT and GPT2, has led to increased performance on a number of downstream NLP tasks from question answering to summarization. Because these improvements are widespread, the representations these models learn likely include information that is relevant to understanding language in general, such as understanding grammatical information (from here on referred to as "syntax").

As such, many have attempted to determine the extent to which these models encode syntactic information. This body of work enumerates a number of properties that deep language models' representations do capture resulting in the general consensus that these models are capable of predicting syntax without explicit syntactic signal (e.g. parses) during training. We hypothesize that these models are able to learn syntax because the structure helps them in making their predictions, but it is currently unclear how and to what extent syntax and other similar tree structures are encoded in the representations.

Prior work has shown that models can be trained to extract syntactic structure from these learned representations, but the problem with these trained models, or "probes" is that it can be difficult to disentangle how much of the syntactic information comes from the deep representation and how much comes from process of training the probe (Hewitt & Manning, 2019). To this end, in this work, we aim to design an unsupervised structural probe that tries to find tree structure from the representations alone. We can see if the tree structures recovered correspond to common linguistic structures like syntax, or semantics relations, or if there is other structure recovered entirely.

We base our probe on the supervised structural probe put forth by Hewitt & Manning (2019). In brief, their probe searches for a subspace of the word representation space where distances between representations in the subspace correspond to distances between those representations' words in a dependency tree. Instead of relying on actual dependency trees for labels, we use a variant of the EM algorithm in which we sample a number of trees that fit the distances, weight them by the quality of the fit, and using the weighted average as the gold label to update the probe parameters. In order to do this, we need a good way to extract a tree from a set of distances, so we explore that first. In summary, the goal of our first experiment is to determine a good method for decoding a tree from a distance matrix, and the goal of our second experiment is to use this decoding strategy to design an unsupervised structural probe.

## 2. Related Work

Supervised probes are usually small models like logistic regression or shallow neural networks that are trained to pick out some kind of structural information from a learned representation. These models are given one (or more, depending on the task) of the hidden states of the deep models as input, and are trained to predict attibutes like part-of-speech, dependency depth, constituency label, or even sentence length (Belinkov et al., 2017; Blevins et al., 2018; Tenney et al., 2019; Conneau et al., 2018; Hupkes et al., 2018). Some probes are more ambitious, and attempt to reconstruct higher level linguistic features and concepts. For example, (Hewitt & Manning, 2019) attempt to extract an entire syntactic tree from the linear subspace of the hidden state vectors space. Having this structure is useful because it provides information that can be incorporated into word representations for languages that lack the data to trian deep models or can be used to reason over when doing semantic

tasks. In a similar vein, Andreas (2019) probes for compositionality by searching for arbitrary pairs of embeddings that can be composed to create other embeddings.

While these probes have been successful, there is the concern that the data they are trained on are picking up on signal that isn't just structural. If probes are too powerful, they can potentially pick up on word-identity information stored in the embeddings and memorize mappings from words to say part of speech tags (Hewitt & Liang, 2019). One way to avoid these problems is to use unsupervised probes. Some work has focused on providing anecdotal evidence such as looking at a few of the model activations and seeing if they correspond to any human-recognizable syntactic division like parenthesis matching, number, or the end of the sentence subject (Dalvi et al., 2019; Lakretz et al., 2019). These are less than ideal though because they involve experts sifting through a lot of data and finding stories that fit it. Another approach is to find a linguistic task that networks ought to be able to accomplish if they have learned certain structures, and will fail at if they haven't. One example of this is giving grammaticality judgements: Marvin & Linzen (2018) collect pair of sentences differing by a single word, one grammatical and one ungrammatical, and see how often language models assign higher probabilities to the grammatical one compared to the ungrammatical one. These unsupervised methods give a sense of whether the model has learned a particular structure (i.e. syntax), but they do not reveal how this structure is latent in the models. As such, our goal is to extract linguistic structure from model embeddings in an unsupervised manner.

## 3. Structural Probe

The basis of our unsupervised probe will be the structural probe described in Hewitt & Manning (2019). In their work, they learn a projection matrix, $B$, such that the squared-L2 norms of differences between projected model representations, corresponded to distances in a dependency tree.

In short, the method is as follows Given a sentence, $s$, represent it as a list of words:

$$s = [w_1, w_2, \ldots, w_\ell].$$

Also given is a gold distance metric, $d$, defined on these words in the form of a dependency tree from training data. The distance between two words according to the dependency parse is the number of edges in the tree between words.

Next, we can take some large pre-trained contextual word embedding model, $M$, (e.g. ELMo), and use it to generate hidden representations for each word, $h_i \in \mathbb{R}^{\text{ⲕ}}$:

$$M(s) = [h_1, h_2, \ldots, h_\ell].$$

Now, following the intuition that hidden representations for words that are syntactically related should be "close" under some norm in some dimension, we search for a metric $d_B$ close to our gold metric, $d$ (i.e. for all $i, j$, $|d_B(h_i, h_j) - d(w_i, w_j)|$ is small). We define a positive semi-definite projection matrix, $B \in \mathbb{R}^{n \times k}$ with $k \leq n$, to pick out which dimensions to focus on, and define $d_B(h_i, h_j)$ to be the norm of the difference between the projected representations:

$$d_B(h_i, h_j) = ||Bh_i - Bh_j||_2^2.$$

Finally, we use gradient descent to search for the projection matrix that minimizes the difference between these two metrics over all sentences:

$$B = \arg\min_B \sum_s \frac{1}{\ell^2} \sum_{i,j}^{\ell} |d_B(h_i, h_j) - d(w_i, w_j)|$$

This gives us a metric $d_B$. To extract a tree, we consider the weighted fully-connected graph on all $\ell$ words in the sentence, where the weights are the distances according to $d_B$. We then need some way to decode a tree out from these distances—in a sense find the tree that best fits the distance. This would mean that the number of edges between words in the true distances should correspond to the distance defined by $d_B$. In the original work, (Hewitt & Manning, 2019) do this by taking the minimum spanning tree of the fully connected graph. That said, this might not be the best way to get the tree. In our first experiment, we explore the best way to decode a tree from a distance matrix. In our second experiment, we use this optimal decoding scheme to predict syntax in an unsupervised manner.

## 4. Experiment 1: Decoding Trees from Distance Matrices

Our goal for our first experiment is to decompose a distance matrix, $D$, into a tree that explains some structure in the data the model was trained on. This will be an unsupervised decomposition so as to not bias our search for a type of structure (i.e. syntactic structure) in particular.

### 4.1. Method

Formally, given a distance matrix $D$, we want to output an unweighted tree, $T$ such that the distances between nodes in $T$ are close to those in $D$. This is a well studied-problem, particularly in the field of phylogenetics where many are interested in decoding out evolutionary trees from differences in DNA sequences between species (Warnow, 1998). There are two main approaches.

The first approach assumes that the embeddings the distances are calculated between represent all the vertices in the tree. If the distance matrix has $n$ elements, the tree has $n$ vertices. This is the approach that (Hewitt & Manning, 2019) take and is related to finding dependency parses in linguistic data. We take two approaches here. First is finding the minimum spanning tree of the graph induced by the distance matrix. This approach takes the perspective that the best tree will minimize the total distance in the tree. A second approach is to write an integer linear program to find the tree whose edges optimally describe local distances. The integer linear program we solve optimizes over variables $x_{i,j} \in \{0, 1\}$, where 1 represents an edge in the tree between words $i$ and $j$, and a 0 represents no such edge. We constrain the edges to form a tree using the constraints described by (Finkel & Manning, 2008). We seek to maximize:

$$\max_{x_{ij}} \sum_{i,j} x_{ij}(1 - d_{ij}).$$

The second approach assumes that the embeddings the distance are calculated between represent the *leaves* of the tree, and the structure derives from some higher level process. This corresponds to constituency parsing in linguistic data. The natural way of doing this is to use hierarchical clustering techniques–similar to k-means–such as agglomorative clustering.

To evaluate our tree decoding algorithms we can look at how well they are able to identify trees out of mixtures of trees. For simplicity, we start by considering trees on $n = 5$ elements (there are 125 possible trees). We consider the distance matrices imposed by trees in three settings: a single tree, a pair of trees (with randomly chosen weights, $\lambda$), and five trees (with randomly chosen weights, $\lambda$). We have 125 distance matrices that we aim to decompose for each setting. We choose to evaluate our predicted trees in two ways. First we compute a *weighted edge accuracy* representing how many of the edges predicted by the decoding algorithm are in the tree(s) used to generate the data weighted by how much the original tree contributes to the original distance matrix:

$$\frac{\sum_i \lambda_i |\text{gold}_i \cap \text{predicted}|}{n \sum_i \lambda_i}.$$

We also consider the Frobenius norm of the difference between the true distance matrix and the distance matrix induced by the predicted tree:

$$||D - D^{(T)}||_F.$$

This gives us an analogue of distortion—how different distances predicted by the decoded tree are from the true distances.

Weighted Edge Accuracy

| | single tree | two trees | five trees |
|---|---|---|---|
| MST | 1.000 | 0.400 | 0.124 |
| ILP | 1.000 | 0.324 | 0.124 |
| AGG | - | - | - |

*Table 1.* The weighted edge accuracies of decoding the distance matrix with the minimum spanning tree (MST), a locally optimal integer linear program (ILP), and agglomorative hierarchical clustering (AGG)

Average Frobenius Norm

| | single tree | two trees | five trees |
|---|---|---|---|
| MST | 0.000 | 3.695 | 12.666 |
| ILP | 0.000 | 3.914 | 12.666 |
| AGG | 9.673 | 9.108 | 7.116 |

*Table 2.* The average norm of the difference between the predicted tree's distance matrix and the data distance matrices. Again, the comparison is between the minimum spanning tree (MST), a locally optimal integer linear program (ILP), and agglomorative hierarchical clustering (AGG).

In both of these evaluations, we are only considering a single predicted tree, but both of these methods can be extended to handle cases where we predict multiple trees.

### 4.2. Results

We can see that the minimum spanning tree approach and the integer linear programming approach perform almost identically for both metrics. The weighted edge accuracy drops very quickly as soon as more edges are added. The average norm also increases a lot at five trees, though this is likely because the distances from adding five trees are actually larger, so the difference from distances in single tree is large. This is similarly why the hierarchical clustering seems to improve as the number of trees increases—the number of edges between leaves in the hierarchical clustering is larger than in the MST and ILP case. Note that the hierarchical clustering does not give us edges between the vectors the distances are calculated over, so we cannot evaluate on the average edge overlap score.

## 5. Experiment 2: An Unsupervised Probe

Now that we have a sense of good ways to decode a tree from a distance matrix, we are going to use these methods to generate unsupervised labels to train the unsupervised probe.

### 5.1. Methods

Recall that in the structural probe procedure, our gold distance metric, $d$, is given by gold labels from the data. The unsupervised setting of this probe then entails trying to predict the most likely $d$ given the data. To this we use a variant of the EM algorithm.

The starting point for our method for predicting $d$ will be the fact that in expectation, random projections approximately preserve norms of input vectors. This means that after projecting the vector differences between $h_i$ and $h_j$ with a randomly initialized $B$, their norm will (in expectation) be preserved, so by applying our methods from the first experiment to decode a distance matrix into a tree, we can generate labels from $d_B$, and then use these labels to update the probe parameters. Our algorithm takes place in three steps: First we generate $d_B$ from data. Second, we find the set of trees, $\mathcal{T}$, that best explain $d_B$. Finally, we update $B$ using $\mathcal{T}$ as the gold labels.

In the true spirit of EM, ideally we would have all possible trees contained within $\mathcal{T}$, and we could weight them based on how well they explain the distances between representations as given by $d_B$. However, the number of trees on $n$ nodes is $n^{n-2}$, so it quickly becomes unwieldy to enumerate all of them.

Instead, we sample trees that are likely to explain $d_B$ and put them in $\mathcal{T}$ instead. We describe two such methods: finding the minimum spanning tree and sampling random walks from the metric.

As experiment 1 showed, the minimum spanning tree is an effective way to find a tree that explains a distance matrix. Therefore, we set $\mathcal{T} = \{MST(d_B)\}$, using the minimum spanning tree as a point estimate for the distribution over trees. In this scenario our objective is to iteratively minimize:

$$\sum_s \frac{1}{\ell^2} \sum_{i,j}^{\ell} |d_B(h_i, h_j) - d_{MST(d_B)}(w_i, w_j)|$$

Another option is to actually sample trees that are likely to fit $d_B$ from the distribution of possible trees. To do this, we compute random walks according to $d_b$, where the probability of including an edge $(u, v)$ in the graph is inversely proportional to $d_B(h_u, h_v)$. We then use the inverse of the sum of the distances along the random walk as the weight of that walk, and update B accordingly. We formalize our objective as:

$$\sum_s \frac{1}{\ell^2} \sum_{i,j}^{\ell} |d_B(h_i, h_j) - \sum_{T \sim \mathcal{T}} \lambda(T) d_T(w_i, w_j)|$$

where the weight $\lambda(T) \propto \frac{1}{\sum_{(u,v)\in T} d_B(h_u, h_v)}$.

Computing these walks is computationally expensive, so instead of simulating random walks, we make an approximation using random projections. To do this, note that a random walk can be approximated by listing the vertices the walk visits in some order. This means that to generate random walks it is sufficient to have a way of generating orderings of vertices such that vertices that are closer are more likely to appear adjacent in the lists. Because these vertices are represented by vectors, we can create such an ordering by projecting these these vectors onto some other randomly sampled vector. In practice, to do this, we mean center our model representations, sample 500 random unit vectors, compute the dot product between the unit vectors and each hidden representation, and sort the vertices by the resulting dot products to produce the set of candidate trees for that sentence.

To evaluate our trees, we use the undirected unlabeled attachment score (UUAS) proposed by Hewitt & Manning (2019). In effect this is the same as the weighted edge accuracy from experiment 1, but in this case there are no weights. We are explicitly comparing our probing results to dependency trees, in effect evaluating only on ability to pick out syntactic structure.

### 5.2. Data and Model Parameters

For these experiments, we follow Hewitt & Manning (2019) and use the Penn Treebank (Marcus et al., 1994). The Treebank contains news-style texts and contains standardized training, development and test sets containing approximately 40,000, 1,700, and 2,400 sentences respectively.

For our deep, contextualized word representation model, we use ELMo Peters et al. (2018). ELMo representations lie in $\mathbb{R}^{1024}$.

We follow (Hewitt & Manning, 2019) for a number of key hyperparameters. They find evidence that the 1st layer of ELMo contains the more syntactic information than any other layer, so we do our analysis on this layer as well. They additionally claim that having the probe project into a space of more than 64 dimensions does not help much, so we also limit the rank of the probe to 64. Optimization is performed with Adam with a learning rate of 0.001.

### 5.3. Results

Our results are far from positive (See Table 3). Our unsupervised minimum spanning tree probe preforms a bit better than the random baseline, but nowhere near the supervised probe's performance. Our unsupervised random walk probe preforms even worse, even outperformed by the random baseline! This seems to imply that either the optimization

| Probe | UUAS |
|---|---|
| Untrained Baseline | 0.31 |
| Unsupervised MST | 0.39 |
| Unsupervised Random Walk | 0.24 |
| Supervised | 0.77 |

*Table 3.* UUAS (edge prediction accuracies) for our candidate unsupervised probes as well as a randomly initialized probe as a baseline, and trained supervised probe as an upper bound.

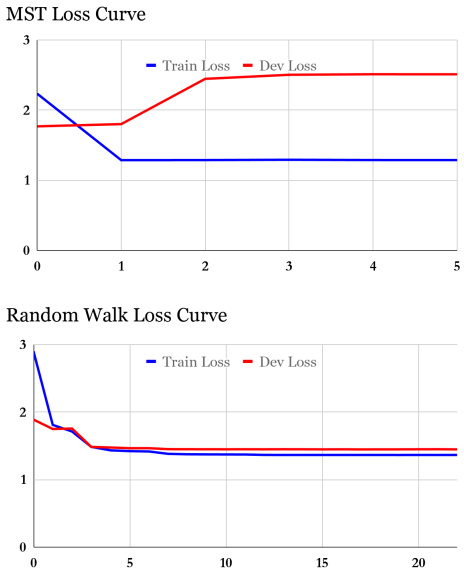


*Figure 1.* Loss curves for both the unsupervised MST probe and unsupervised random walk probe

did not work, or there was some other non-syntactic structure that the sampled random walks gravitated toward had nothing to do with syntax. Figure 1 shows that during training, the training loss decreased for both the MST probe and the Random Walk probe, while the dev loss only decreased for the random walk probe. This likely indicates that the probe is picking up on something. This might be indicative of finding some kind of structure, but the UUAS values indicate that it has little to do with dependency syntax.

## 6. Discussion and Future Work

There are a few interpretations of these negative results. The first is that this probing strategy does not work. Even if using a weighted average of random walks as labels in an EM-style way is not effective, it's still possible that a similar method could make progress. Perhaps a more heoretical motivated and principled derivation of a weighting function (analogous to q-value in the E step, or some kind of probability of a tree) could yield better results.

A second interpretation is that syntactic information just isn't stored in the distances between certain dimensions of ELMo. Perhaps having labeled data to learn a mapping from however syntactic information is stored (e.g. clusters of syntactically similar words) to a format needed to do well at this probing task. This is somewhat unlikely—the probe does not have many parameters (in fact it's linear), and the original paper introducing the probe compared it to strong baselines indicating that it's likely that some information is encoded in this way. Nonetheless, a very good way to confirm this would be to train a probe to predict some tree structure that is *unrelated* to syntax and see if the probe can still perform well. If it can, likely it is the data that is contributing a lot of these performance (Hewitt & Liang, 2019).

Yet a third interpretation of these results is that the unsupervised probes worked and found some structure, but the structure doesn't correspond to dependency structures. The structure might be related to semantic relationships such as coreference or semantic roles. One way to check this would be additionally find datasets that have this sort of data and see how well this learned probe performs.

There are some additional directions to move in after sorting out how to perform the sampling and getting a clearer picture of the results. First, as mentioned previously, there has been work on sampling trees from distance matrices, such that the sampled trees are expected to distort the distances by only $O(\log(n))$ with high probability (Fakcharoenphol et al., 2004). It would be interesting to try to use their sampling procedure to generate likely trees. Unfortunately the trees sampled by their method are closer to constituency trees rather than dependency trees (i.e. all of the vertices are leaf nodes), so some consideration has to go into the conversion.

## 7. Conclusions

With large-pretrained contextual word embedding models leading to huge performance gains on a number of English-language tasks, figuring out what these opaque models learn is very important. The structures extracted from these models could be used to better understand the relationship between language and linguistic structure and could inform the creation of better word representations for languages that do not have as much data available to train these large models on. Most of the previous work constructs supervised probes, and therefore adds information into the system, or treats the models as black blocks, ditching supervision for the sake of interpretability. In this work we attempt to bridge this gap by introducing an unsupervised structural probe based on the EM algorithm. While our results are poor, there is still hope that unsupervised probes will allow prove useful for obtaining the insights we desire.

## References


Andreas, J. Measuring compositionality in representation learning. *arXiv preprint arXiv:1902.07181*, 2019.

Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., and Glass, J. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*, 2017.

Blevins, T., Levy, O., and Zettlemoyer, L. Deep rnns encode soft hierarchical syntax. *arXiv preprint arXiv:1805.04218*, 2018.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., and Baroni, M. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018.

Dalvi, F., Durrani, N., Sajjad, H., Belinkov, Y., Bau, D. A., and Glass, J. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. 2019.

Fakcharoenphol, J., Rao, S., and Talwar, K. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

Finkel, J. R. and Manning, C. D. Enforcing transitivity in coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pp. 45–48. Association for Computational Linguistics, 2008.

Hewitt, J. and Liang, P. Designing and interpreting probes with control tasks. *arXiv preprint arXiv:1909.03368*, 2019.

Hewitt, J. and Manning, C. D. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4129–4138, 2019.

Hupkes, D., Veldhoen, S., and Zuidema, W. Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926, 2018.

Lakretz, Y., Kruszewski, G., Desbordes, T., Hupkes, D., Dehaene, S., and Baroni, M. The emergence of number and syntax units in lstm language models. *arXiv preprint arXiv:1903.07435*, 2019.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pp. 114–119, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. ISBN 1-55860-357-3. doi: 10.3115/1075812.1075835. URL https://doi.org/10.3115/1075812.1075835.

Marvin, R. and Linzen, T. Targeted syntactic evaluation of language models. *arXiv preprint arXiv:1808.09031*, 2018.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Van Durme, B., Bowman, S. R., Das, D., et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019.

Warnow, T. Some combinatorial optimization problems in phylogenetics. *Unpublished manuscript*, 1998.


## A. Contributions

While I am the only one who performed this project, I am currently working on a related project with John Hewitt in the NLP lab. Much of the unsupervised probe code is based on his supervised probe work. Code can be found here: https://github.com/bnewm0609/tree-decomp.