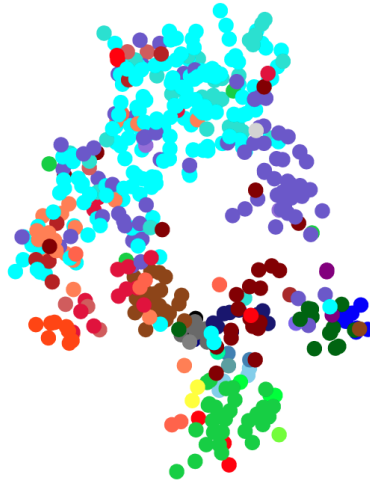


A work in progress:

# NeuralGREWT: Neural Grammar Rule Extraction and Word Taxonomy



[Click to Interact](#)

Unsupervised learning and comprehensible representation of grammar rules and fuzzy symbol categories by high-dimensional clustering of string probabilities derived from natural language models and Comprehensible Convolutional Neural Networks (CCNN).

## Theory

A *grammar* is a set of rules for how symbols<sup>[1]</sup> may be arranged to form valid strings in a language. Strings of symbols that follow all grammar rules are grammatically *valid*. Grammatical validity is necessary for, but does not imply, sensibility. For example: the string of symbols (in this case a sentence of words) "The hat runs and cooks into a hairdresser." is a grammatically valid sentence that is also nonsensical in isolation<sup>[2]</sup>.

Given this definition of grammatical validity, valid sentences will be more common than invalid ones in any string-set comprised mostly of sensible strings; making it possible to infer a string's validity from its probability.

Therefore, it may be possible to learn grammar rules of any language, including computer, fictional, and extra-terrestrial languages, without needing to make sense of anything written in the given language<sup>[3]</sup>.

Let's test this hypothesis.

One good first step to learning grammar rules may be to identify the different categories of symbols present in a given language.

Symbols will be said to share a syntactic *category* in proportion to their mutual interchangeability. In other words: 2 symbols will share a category in proportion to the probability that one may be replaced by the other in a randomly chosen valid string without rendering that string invalid.

Knowing this, a natural language generator<sup>[4]</sup> can be used to determine the degree of mutual interchangeability of symbols in a language using total string likelihood before and after replacement as an indicator of relative validity, and therefore of mutual interchangeability — implying taxonomy. Once we have these validity scores we can use T-SNE to infer discrete symbol categories from the degree of mutual interchangeability (or equivalently the clustering of validity-under-replacement scores).

## Procedure

- Compile a large *corpus* of valid and/or sensical writings in the chosen language.
- Train a natural language model, or *predictor*, on the corpus until it is very good at predicting symbols in the language given context. Anything like BERT or GPT-2 will do fine, and in fact are probably overkill.
  - NOTE: It's definitely overkill. Future work will employ a maximally-minimal version of BERT to accelerate calculation of validity tensors and eliminate "understanding intrusion". Large BERT and GPT-2 models possess a good deal of understanding of written text that leads them to rank the probability of valid but otherwise non-sensical strings significantly lower than they would if they understood only base grammar relationships. Clearly this is not good if we are looking for a measure of base validity regardless of sensuality<sup>[5]</sup>.
- Compile a large "ground set" of valid and/or meaningful strings in the chosen language. (in the case of unknown languages, this ground set can just be a random subset of the corpus)
- Compile a "symbol set" of all symbols in the chosen language, or at least a large set of the most common ones.
- Create a 4 dimensional *perturbation tensor* from the ground and symbol sets by:
  - Add to the perturbation tensor *acube* for each *symbol* in the symbol set; construct each cube by stacking a *plane* for each *ground string* in the ground set; construct each plane by stacking a *vector* for all strings that can be created by replacing any one item in the ground string with the symbol. Of course, making sure to do this in the same order for all vectors<sup>[6]</sup>.
- Create a 4 dimensional *validity tensor* from the perturbation tensor by:
  - For each vector in the perturbation tensor, judge the probability of that vector by summing the relative likelihoods of each symbol appearing at its location given all previous symbols in the vector (using the predictor), taking the difference between this and the sum-validity of it's corresponding ground string to obtain a *validity delta*, and dividing by the length of that vector. That division might be unnecessary, but we will find out.
    - Update: It occurs to me that the division by sentence length would have a largely unhelpful effect on the validity score of any one dimension relative to all the others; exaggerating similarities between dimensions as vectors grow longer. The ground-truth validity delta may or may not have anything to do with string length depending on the language so it is wrong to force such a correlation in general. A proper normalization across all dimensions regarded equally is what we want here.
- Perform T-SNE followed by PCA on the validity tensor to infer number and relative importance of symbol categories.
  - NOTE: In this case PCA will not, and cannot provide a useful classifier for datapoints that were not already present in the T-SNE plot. Here we are simply looking to quantify the number of clusters, and get some small idea of the distances between clusters<sup>[7]</sup>.
- Name each symbol category. (can be totally arbitrary)
- Create a *sym-cat* mapping of each symbol to a list of its categories sorted in descending order of the symbol's *belongingness* to each category.
- Create a *cat-sym* mapping of each category to a list of its symbols sorted in descending order of each symbol's belonging-ness to the category.
- Create a *token set* from the corpus by replacing each symbol in the corpus with a name, *ortoken*, for its corresponding category<sup>[8]</sup>.
- Create a *garbage set* from the token set by randomizing a large enough proportion of the tokens in the token set to render each string likely invalid.
- Train a comprehensible, spiking, convolutional neural network, *ogrammar net*, to distinguish between items of the token and garbage sets.
  - The filters of the trained grammar net are first order grammar rules, the fully connected layers behind those are second order grammar rules, and so on. (or something to that effect depending on the structure of the grammar net.)

## Status

- ☑ Gather Corpus
- ☑ Train Natural Language Model (GPT-2)
- ☑ Compile Ground Set
- ☑ Compile Symbol Set
- ☑ Generate Perturbation Tensor
- ☑ Calculate Validity Tensor
  - Signs from the first few checkpoints are looking very promising. Replacements known to largely preserve validity (such as pronouns with any other pronoun) are displaying markedly higher probability to the predictor. The trend is so striking that it is visible to the naked eye even before normalization. Feels like cheating a little bit, since I am trying to stick to a pre-registered method, and peeking at the data while it's being processed can't be good for my objectivity. I did have to look at least once to make sure that the program was calculating what I meant it to, so I suppose it can't be helped that I went looking for patterns like these. I'll just have to be extra careful not to let this good news inform any of my future decisions. Best way to do that is to make as few of them as possible and stick closely to the plan as written.
  - [ Trial set generated in 10 days of computation time on a GTX 1070 M ]

#### ☒ T-SNE

- To improve our dynamic range, each dimension of the validity tensor is scaled independently of the others so that its values now range between -1 and 1 for all symbols. This is to ensure signals of dimensions with naturally smaller values are not so severely washed-out by signals of naturally larger dimensions in the next step.
- After scaling, the validity tensor is passed through a heaviside function to produce a tensor of sparse boolean vectors with ones for all values above a specified threshold. Implicitly, this means that all values above some high-but-ultimately-arbitrary validity threshold<sup>[9]</sup> are regarded as totally valid replacements, and all others are assumed to be equally invalid.
- T-SNE was performed using the jaccardian dissimilarity index of each vector against every other vector as a distance metric together with Barnes-Hut approximation to find a good representation of the clustering in a visualizable space.
- The step above was performed with many perplexities ranging from 5 to 50. Qualitative optimum was found to lie between 12.5 and 17.5<sup>[10]</sup>.

#### ☒ PCA

- It is clear from the visualization that PCA is not the best choice here afterall. The trouble now is not in finding those dimensions that best define the data, but in finding a way to identify the boundaries of distinct (and not-so-distinct) t-sne clusters in the visualizable space with minimal if any human involvement. DBSCAN seems well suited to this task. Barring that, many of these clusters seem isolated enough to eyeball without injecting too much subjectivity (hence the reduction to visualizable space), but an automatic approach is always preferable.

#### ☒ Automatic Cluster Identification with DBSCAN

##### ☐ Category Naming and Mapping

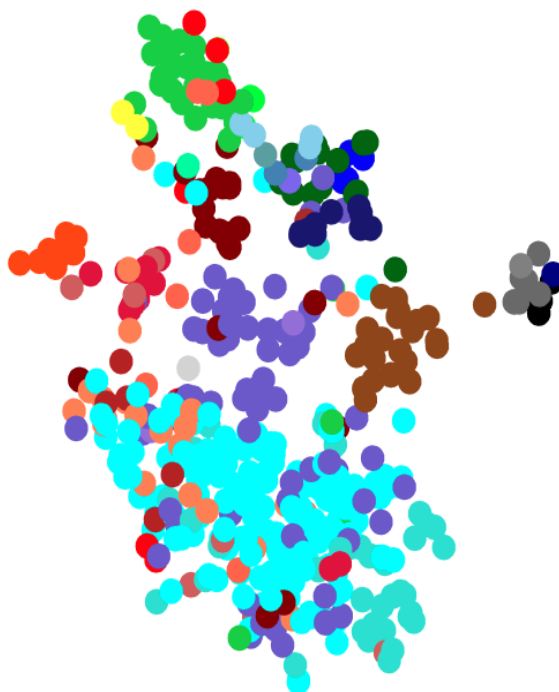
##### ☐ Generate Token and Garbage Sets

##### ☐ Train Grammar Net




















- One good way to make the grammar net more comprehensible would be to use a very large number of filters in the convolutional layers and **strong** regularization across the board, so as to ensure that the vast majority of filters obviously do nothing<sup>[11]</sup>, while emphasizing the important few that communicate a lot.

## Results

- Word Taxonomy Visualization:



[Click to Interact in 3D](#)

Color	Part of Speech	Example
	sentence terminators	. ! ?
	middle punctuation	- , ; " ... ( )
	quotes	"
	dollar sign, list marker, or foreign word	\$, 1), Кошка
	interjection	Erm, Um, Drat!
	digits	0, 1, 2, ...
	single letters a, b, c, ...	
	determiner	a, an, the, every
	predeterminer	'all the kids'
	coordinating conjunction	and, but, or
	preposition/subordinating conjunction	
	to	go 'to' the store.
	existential there	"there is" ... "there exists"
	particle	give up
	modal	could, will, should, may
	adjective	big
	adjective comparative	bigger
	superlative adjective	biggest
	personal pronoun	I, he, she
	possessive pronoun and possessive endings\	my, his, hers, 's
	singular noun	desk
	noun plural	desks
	singular proper noun	Harrison
	plural proper noun	Americans
	wh-determiner	which
	wh-pronoun	who, what
	possessive wh-pronoun	whose
	wh-adverb	where, when
	adverb	very, silently
	comparative adverb	better
	superlative adverb	best
	base verb	take
	past tense verb	took
	present participle verb	taking

## Footnotes

1 More precisely: "morphemes" composed of symbols. ↩

2 "in isolation" meaning in the absence of explanatory context. ↩

3 So long as we are in possession of an ample body of text that would be sensical to a speaker of that language. ↩

4 Any predictor of morphemes from context really. ↩

5 This will also ensure that we get a good measurement of validity in some imaginable language where it is impossible to have grammatically valid strings that don't make sense. This would mean only that validity implies sensicality. In such languages, learning base validity is identical to learning sensicality, which may-or-may-not require a larger network to accomplish (interesting philosophical question there) but is nevertheless possible as demonstrated by large language models of recent years. ↩

6 In an actual tensor all such vectors need to be padded to uniform dimension with null strings. In practice we can use a 3 dimensional tensor of pointers to vectors of variable dimension. ↩

7 Distances between T-SNE clusters are sometimes meaningless, so the distances according to PCA are not to be taken seriously in this case. However, the number of clusters and degree of overlap communicated by PCA on T-SNE will be reliably meaningful if good clusters are found.

- NOTE: this subtlety is obviated by PCA's replacement with DBSCAN in the cluster identification step. ↩

8 In the proof of concept we will simply pick the highest ranked category from each symbol's entry in the sym-cat mapping. Future work

will be needed to make an informed choice of category in the case of context dependent taxonomy. ↩

9 In this case above -5 or thereabouts after scaling.↩

10 Specifically: optimum for a set of 500 points in ~5500 dimensions. This optimum is likely to change depending on your point-count to dimensionality ratio. ↩

11 or are lower-weighted, redundant duplicates that all do the same thing↩

## How to Install

[TBD]

## How to Use

[TBD]

## Contributing

For contributors to the project; do this before making your first commit:

- Install pre-commit

```
cd /path/to/this/repository/  
sudo apt install pre-commit  
pre-commit install
```

(we do all of our development on linux)

- To test updates to the readme and other GitHub flavored markdown, simply install Grip and feed it your desired file.

```
pip3 install grip  
python3 -m grip README.md
```

- Then follow the link provided by the Grip sever for a live preview of your work.
- When satisfied with your changes you can compile to an html file with:

```
python3 -m grip README.md --export README.html
```

## Authors

- **Gabe M. LaFond** - *Initial work* - [ExamDay](#)

See also the list of [contributors](#) who participated in this project.

## License

This project is licensed under the MIT License - see the [LICENSE.md](#) file for details