Grammar Induction - Experimental Results

Linas Vepštas[0000−0002−2557−740X]

OpenCog Foundation <linasvepstas@gmail.com>

Abstract. The OpenCog Learning project explores novel techniques for the induction of symbolic structure from environmental sources. As a first application of these ideas, the automated learning of the grammatical structure of English is explored. This document provides a summary of results observed to date. The symbolic structure manifests itself as large, complex graphs tying together words and grammatical structure. The induction process provides probability distributions on the graphs. The primary task is to describe and understand the nature of those graphs.

Introduction

The goal of the OpenCog Learning project is to induce structure from raw observational data. In the most direct sense, this would be visual or auditory data. In a more sophisticated sense, "observational data" will already have been processed by other information extraction layers, as the goal of learning is to do so hierarchically, recursively: to deduce structure in structure inside of structure... If this can be made to work, then, "starting in the middle" is not a bad place to start. In the present case, "the middle" is a corpus of English text, and the goal is to extract grammatical structure, including morphology, syntax and the lower reaches of semantics.

The general theory of how this can be done is sketched out in a companion article.[6] The overall process is as follows: first, one computes the mutual information (MI) between word-pairs occurring in the corpus. The MI assigns a numerical score to how often a pair of words occur near each-other. This can be used to generate a maximum spanning tree (MST) parse of the text, which captures the syntactic structure of the text.[7] This parse tree can be disassembled into individual "jigsaw pieces" which encode the syntactic structure of the parse.[4] Individual jigsaws encode the local syntactic environment of a given word, not unlike skip-grams. Collections of such jigsaws can be understood to be vectors: each word is endowed with a lexis of all of the syntactic contexts in which it was observed. But this is too fine-grained; one wishes to generalize from the particular to the general. The vectors can be used both to discover words that have similar syntactic contexts (based on the similarity of the local environments) and also to factor out different word-senses, since different syntactic contexts associate strongly with distinct word-senses.[1,2]

This text reports on the algorithms developed to perform the above, and on the statistical properties of the resulting graphical networks. The report is summary, encom-

passing more than a decade of research and development.[1] There are obvious directions in which both the theory and the experiment could be taken, but haven't been so far. In the "downwards" direction lies morphology, needed for Indo-European languages, and segmentation, needed for Chinese. Further down, the extraction of phonetic structure and general audio processing. Levering the same tool-set upwards, one can search for high-MI pairs across multiple paragraphs, both to resolve references and to detect entities. Syntactic relations can also be extracted between different sense streams, say, between words and blueprints/diagrams, or words and photographs, or other hi-tech sensory data. The author believes the basic algorithm is generic, and can be ratcheted upwards, to arbitrarily high reaches. Whether this proves to be true cannot be known without conducting actual experiments.

The remainder of this paper focuses on the narrow tasks defined above, in the order given. The computation of word-pairs results in a graph, whose vertexes are words, and whose edges are word-pairs. What are the statistical properties of this graph? MST parsing and the creating of jigsaw pieces results in a matrix of (word, connector-sequence) pairs. What are the statistical properties of this matrix? Words are vectors in this matrix; the similarity between words can be judged with assorted different vector measures; how do these behave, and how does clustering and word-sense disambiguation proceed, in practice?

Software Infrastructure

All results were obtained by using the software found in the GitHub repo for the OpenCog Learn project.[2] It is built on top of the OpenCog AtomSpace[3] as a foundational layer. The AtomSpace is an in-RAM graph database with powerful graph query capabilities, tuned for performance. It has shims that allow data to be stored in conventional disk databases and to be distributed across the network. Most notable for the present task is the "matrix API":[4] it allows arbitrary collections of subgraphs to be used as the basis elements of a vector space. If one has two such vector spaces, then pairs of elements form a matrix of rows and columns. Given a matrix, one may compute marginal and conditional probabilities. This is, of course, quite ordinary. What is new is that the AtomSpace allows extremely sparse matrices to be represented: say, 100K $\times$ 100K entries, of which all but a few million are zero. Conventional software packages do not provide tools for sparse data. Furthermore, the "base data" in the AtomSpace are graphs; the matrices themselves are but slices through the graphs. For example, during word-sense disambiguation, columns correspond to connector sequences; these are selected subgraphs taken from the total parse graph.

[1] A detailed diary of results, spanning a many hundreds of pages, can be found in the Dairy, Parts One-Six, and the adjunct reports. Rather than peppering this text with self-citations, footnotes will be used to indicate which of these to examine.

[2] See https://github.com/opencog/learn

[3] See https://github.com/opencog/atomspace

[4] See https://github.com/opencog/atomspace/tree/master/opencog/matrix

Pair Counting

Obtaining a suitable corpus for the English language is not as simple as it seems: it is not enough to train on a dump of Wikipedia articles. Wikipedia is descriptive, having a paucity of verbs and a surfeit of proper nouns and technical terms. The verbs of everyday outdoor adventure: run, jump, catch, are missing. This was remedied by working with a corpus of young adult adventure literature taken from Project Gutenberg and publicly licensed fan fiction.

Word-pair counting is done by generating random planar parse trees, and counting the connected pairs. This provides a uniform sampling over the space of all possible parse trees, eliminating edge effects from conventional finite-sized window sampling. Overall, though, the difference between this and window sampling are not obviously discernible.

Data sets Five snapshots of an increasingly larger English-language corpus were taken. These are summarized below.[5]

Corpus	1	2	3	4	5
$\log_2 N_L$	16.678	17.097	18.214	18.600	19.019
$\log_2 N_R$	16.690	17.117	18.228	18.620	19.039
$\log_2 D_{\text{Tot}}$	23.224	23.797	24.748	25.180	25.627
Sparsity	10.144	10.416	11.693	12.040	12.431
Rarity	6.540	6.690	6.527	6.570	6.598
$\log_2 N_{\text{Tot}}/D_{\text{Tot}}$	4.779	5.079	5.128	5.235	5.335
Total Entropy	17.827	17.889	18.378	18.503	18.631
Left Entropy	9.7963	9.8102	10.069	10.109	10.148
Right Entropy	9.5884	9.5463	9.8321	9.8801	9.9265
MI	1.5572	1.4677	1.5227	1.4863	1.4431

In the table above, N_L and N_R are the height and width of the matrix (the number of unique words occurring on the left and right of observed word-pairs). They are almost the same, but not quite: periods and question marks never occur at the start of sentences; capitalized words rarely appear at the end. The sizes are given as logarithms, as this makes comparison to entropies more immediate. The total number of unique pairs is D_{Tot} while the total number of observations of these pairs is N_{Tot}.

The sparsity is $-\log_2 D_{\text{Tot}}/N_L \times N_R$, and is expressed in bits. It indicates what fraction of all possible word-pairs are actually observed. Observe how the sparsity increased by two bits, as the dimensions of the matrix increased by two bits (a factor of four; non-trivial in terms of compute-time, corpus size and RAM usage.) Thus, rarity is defined as $\log_2 D_{\text{Tot}}/\sqrt{N_L \times N_R}$, and is seen to be approximately constant, independent of dataset size.

The total entropy is

[5] See Diary Part Two, page 75 and Diary Part Three, page 7. The corpus was divided into five "tranches"; each dataset includes one more tranche of the corpus.

$$H_{\text{Tot}} = \sum_{w,v} p(w,v) \log_2 p(w,v)$$

with the sum ranging over all word-pairs (w,v) and the frequency $p(w,v) = N(w,v)/N(*,*)$ where $N(w,v)$ is the observation count of a pair, and $N(*,*)$ is the total count over all word-pairs. The left entropy is

$$H_{\text{Left}} = \sum_{w} p(w,*) \log_2 p(w,*)$$

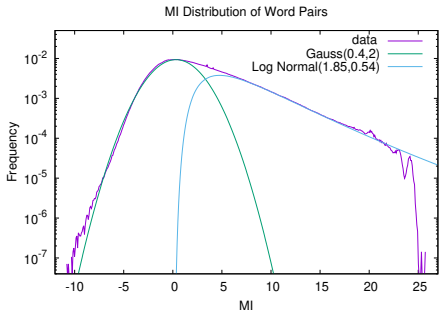
where $p(w,*) = \sum_v p(w,v)$ is the left-marginal sum; likewise for the right entropy. The mutual information is

$$MI = H_{\text{Tot}} - H_{\text{Left}} - H_{\text{Right}} = \sum_{w,v} p(w,v) \log_2 \frac{p(w,v)}{p(w,*)\, p(*,v)}$$

Notable trends are that the number of distinct pairs increases as the square-root of the possible number of pairs. The total number of observations per pair is very nearly constant, increasing only slowly with the size of the dataset. Notable is that the MI really does appear to be constant, independent of the size of the dataset! The low value of the MI, one and a half bits, indicates the paucity of word-pairs. They convey some information, but not very much. On the other hand, the sparsity is quite high: of all possible word pairs that could have been observed, only minuscule fractions of a percent are actually observed.

Remarkably, the overall numbers appear to be language-independent; almost identical values are seen for Mandarin Chinese text.[6] To perform this counting, each Hanzi is taken to be a "word". This is not technically correct, as Chinese words may consist of one, two, three or more Hanzi. However, segmentation is a challenge, and so at this level, for pair-counting, issues of segmentation are ignored. Presumably, the discovery of Chinese words as set-phrases will occur later in the learning pipeline.

MI Distribution The distribution of word-pair MI is shown below. This is for dataset 3, which contains 28 million pairs. The MI is sorted into 500 histogram bins.



[6] See the "Word Pair Distributions" document.

The distribution is clearly not symmetric. The two sides appear to be bounded by straight lines. The author is unaware of any theoretical explanation for this shape. However, the following guess is offered. If word-pairs are chosen completely at random, and the number of sampled pairs is much smaller than the total possible pairs, then one obtains a Gaussian distribution. Such a distribution is centered on a small but positive MI, due to sample-size effects. For larger samples, the mean tends to zero. Thus, perhaps the left-hand-side of this figure is just a Gaussian. An eyeballed, imprecise fit is shown.

Taking this to be "common-mode noise", and subtracting it leaves an excess of word pairs with positive MI, having a peak near $MI \sim 4$. The straight-line slope on the right suggests that the excess can be described by a log-normal distribution. Again, an eye-balled, imprecise fit is shown. These two, summed together, model the observed distribution almost perfectly.

Pairs with the highest MI are observed very infrequently. The highest observable MI value is directly related to the sample size: it is a bit below the log of the number of observations. Thus, the sharp drop on the right side is purely a sample-size effect. Trimming does not appreciably change the shape of this distribution, other than to eliminate the very highest MI values. This distribution is not language-specific; a nearly identical distribution is seen for Chinese Mandarin Hanzi pairs.[7]

MST Parsing

Armed with word-pair data, one can proceed to perform Maximum Spanning Tree (MST) parsing, as described by Yuret.[7] One considers all possible planar trees connecting all of the words in a sentence, and, out of all of these trees, the one with the greatest sum-total MI of the edges is selected. A variation is the Maximum Planar Graph (MPG), which takes the above tree, and adds edges for high-MI pairs, as long as edges don't cross and the MI is above a threshold. Experience with Link Grammar (LG) suggests that MPG parses offer a significant advantage over trees. There, cycles force grammatical agreement between nearby words, which introduces a rigidness, resulting in a greater rejection of bad parses. This result should carry over to the present case, and help constrain the learned grammar.

Given an MST or MPG parse, each edge is cut in half, and the half-edge is labeled with the word it used to connect to. This results in a connector sequence for each word, some connectors connecting to the left, some to the right. Every connector sequence d, or disjunct for short, is paired with a word w to form a word-disjunct pair (w,d).[8] Parsing a large corpus this way, the counts are accumulated in the database, to give a count $N(w,d)$.

The first thing one notices is that there are vastly more disjuncts than there are words: two orders of magnitude more! This is easily explained: each disjunct is a context for a word: it resembles a skip-gram in many ways.[5] Next, one notices that the vast majority of disjuncts are observed only once. This raises the question: what is signal and what is noise? This can be explored through trimming, by removing words,

[7] See "Word-Pair Distributions", page 18.

[8] For example, parsing "*John hit the ball*" results in a (w,d) pair `(hit, John- & ball+)` for the verb "hit": a subject on the left, and an object on the right. Details can be found in [5].

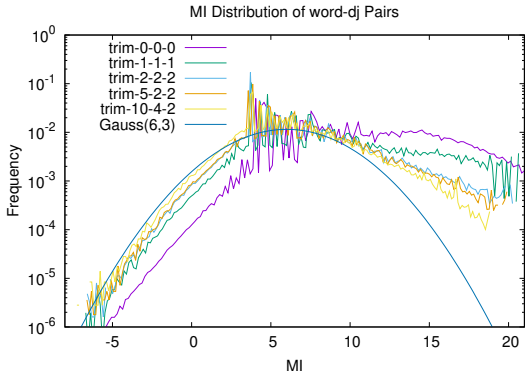
disjuncts and word-disjunct pairs whenever their observation counts fail to pass a minimal threshold.

Results are summarized in the table below.[9] The column labels indicate the trimming thresholds; in the rightmost column, all words with an observation count of less than 10 were removed; all disjuncts with an observation count of less than 4, and all word-disjunct pairs with a count of less than 2. After this initial trimming, an additional consistency trim is performed, to guarantee that all remaining words can connect to some connector in some disjunct, and *vice versa*.

Trim cuts	full set	1-1-1	2-2-2	5-2-2	10-4-2
$\log_2 N_{\text{words}}$	18.526	15.542	13.644	12.889	12.249
$\log_2 N_{\text{disjuncts}}$	24.615	20.599	18.662	18.447	17.369
$\log_2 D_{\text{Tot}}$	24.761	20.967	19.247	19.086	18.443
Sparsity	18.380	15.174	13.058	12.251	11.175
Rarity	3.191	2.896	3.095	3.418	3.634
$\log_2 N_{\text{Tot}}/D_{\text{Tot}}$	0.356	2.248	3.384	3.461	3.889
Total Entropy	24.100	19.486	17.711	17.508	16.875
Left Entropy	23.494	18.346	16.417	16.163	15.379
Right Entropy	10.157	7.937	7.280	7.268	7.258
MI	9.550	6.796	5.987	5.923	5.763

Notable in this data is that even modest trimming removes the vast majority of words and disjuncts in the dataset. The total entropy and sparsity is strongly dependent on the trim; the rarity is almost a constant. It will later become apparent that heavy trimming is perhaps not a good idea: there is a considerable amount of information held in the infrequently-observed disjuncts. Characterizing this precisely, understanding what it means remains an ongoing task.

The MI between words ad disjuncts can be computed; this forms a distribution, shown below, both for the full dataset, and for the trimmed variants.[10]



[9] The raw data is in Diary Part Six.

[10] An earlier version of this graph, for a different dataset, can be found in "Connector Set Distributions" (2018) page 22.

Drawn for comparison is a Gaussian, centered at an MI of 6, with a standard deviation of 3. There is no compelling reason to believe that the distribution should be a normal distribution; just that it seems not inappropriate. Compared to the Gaussian in the earlier figure for word-pairs, this one is clearly centered far away from MI=0. A certain "amplification of information" appears to have occurred.

Grammatically Similar Words

The machinery above brings us to the first interesting grammatical application: determining what words are grammatically similar. Each row in the above matrix is a vector; it corresponds to a word with a collection of disjuncts associated with it. Each disjunct is a context for that word: it is very much like a skip-gram, familiar from corpus linguistics. Unlike conventional skip-grams, it is determined by MST/MPG parsing, rather than simple observation frequency. This should result in much higher quality data, although this hasn't been directly demonstrated, yet. The interpretation is also different than a conventional skip-gram: the context of the word is re-interpreted as jigsaw connectors; and connectors must connect to form a valid parse of a sentence. This is very unlike conventional neural net approaches, where there is no explicit appearance of any grammar.

How should similarity be compared? Conventional approaches employ the cosine distance. Experimentally, this appears to be the worst-possible way of determining similarity.[11] Other possibilities include the Jaccard distance, and weighted variants thereof. One of these is optimal, in that it maximizes all possible similarities.[3] Any of these do quite well. Keeping with the information-theoretic theme, using the mutual information between two vectors seems appropriate. This needs to be defined afresh, as it is not simply a restatement of that given above. Let the joint probability $p(w,d)$ of a word-disjunct pair (w,d) be as before: $p(w,d) = N(w,d)/N(*,*)$. Define a vector inner product of two words w,v as

$$i(w,v) = \sum_d p(w,d)\, p(v,d)$$

The corresponding MI is then

$$MI(w,v) = \log_2 \frac{i(w,v)\, i(*,*)}{i(w,*)\, i(v,*)}$$

Note that this MI is symmetric: $MI(w,v) = MI(v,w)$, which follows from the symmetry of the inner product.

How well does this work? Fantastically well; creating a list of the top-100 most similar words, as ranked by MI, provides excellent results, excellent in the sense that just looking at the list it is clear that they really are synonyms, or are obviously grammatically similar. But then a problem becomes evident: all of the high-MI word pairs involve rare, infrequent words. If one is to build a syntactical lexis, one really wants to

[11] See Diary Part Two, pages 55-74.

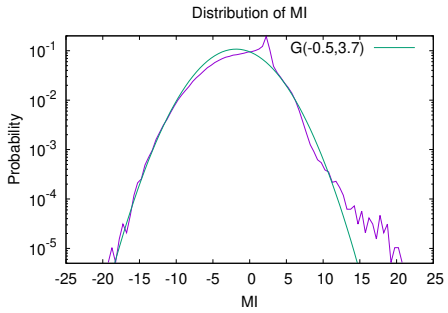
begin by looking at common, frequent words. A weighting that brings frequent words to the forefront is desirable.

Such a weighting is given by the "variation of information". The variation of information is given by

$$VI(w,v) = \log_2 \frac{i(w,v)}{\sqrt{i(w,*)\,i(v,*)}}$$

There are also other possibilities, which look even more promising; the above, however, is the cheapest and easiest to compute.[12]

The graph below shows the distribution of MI similarities for the top-ranked (most frequent) 1200 words.[13] In principle, there are $N(N+1)/2 = 720600$ such pairs. In practice, 386380 pairs are observed; the remaining pairs have no overlap! (and thus a $-\infty$ for the MI.)



The curve marked G is a Gaussian, with the indicated mean and standard deviation. The negative mean MI is just saying that most words are *not* similar to one-another (duh.) The graph for the VI is nearly identical, having the same deviation, but with the mean shifted to 1.5.

Clustering

How well does this work? Some of the top-most similar groups of words are shown in the table below.[14]

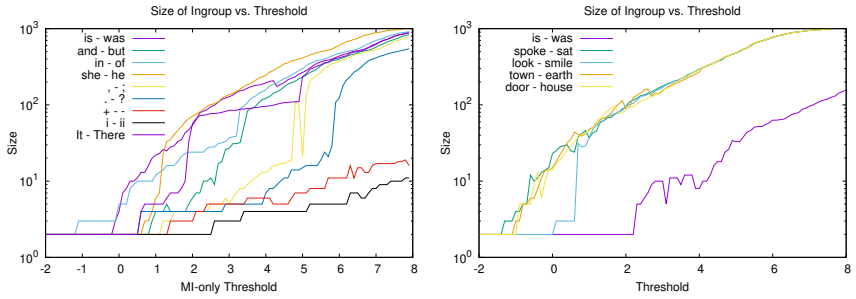
Top-ranked Clusters		
+ — " " _	? . !	must would
, ;	He It I There	he she
was is	of in to from	are were
but and that as	has was is had could	might should will may

[12] See Diary Part Five.
[13] See Diary Part Three, page 14; see also Diary Part Five.
[14] See Diary Part Five.

The clusters above are sets of words, all of which have a high pairwise-VI amongst one-another. Most of these are obvious. The cluster of capitalized words is perhaps surprising: but these are all sentence-starters: they are similar because they all start occur first, and start similar sentences. (Recall, the corpus is adventure literature, with a lot of dialog in it.)

To get adequate word-sense disambiguation (WSD), one must be more sophisticated in clustering. A cluster should include only those disjuncts which are shared by the majority, excluding those that are not. An example of this is the word "saw", which can be the past tense of "to see", or a cutting implement. In forming a cluster with other viewing (or listening) verbs, one wishes to accept only the disjuncts pertinent to viewing/listening. Excluding the others leaves behind the disjuncts for cutting tools. Thus WSD is accomplished.

This can be accomplished with "exclusive club – common-interests" membership.[15] Starting with the pair of words having the highest possible VI, one searches for additional words having a high VI to these two. These form the candidate members to the club. Candidates are accepted if they pass a threshold. It is useful to have a reasonably large club, which can be expanded by gradually lowering the threshold. As it is lowered, the candidate list increases, until suddenly it explodes. The exclusive club is no longer exclusive, but is inviting everyone to join. The graph below shows the size of the candidate list, as the barrier to entry is lowered. The listed word pairs are the initial seed-words for forming a cluster.



On the left, all of the seed-pairs are made of very frequent words. Does this pattern continue to hold for infrequent words? It appears to, as the graph on the right shows.

The above only provides a list of candidate members. Setting a high threshold keeps the candidate list small. But which words, exactly, should be admitted to the club? This can be determined by seeing what disjuncts the words share in common. For a given candidate list, one simply counts the fraction of all disjuncts that the words have in common (think of these as common interests in a social club). Next, consider ejecting one of the candidates; does the fraction of shared disjuncts increase, or not? If it increases, then the ejected candidate should be thrown out of the club. If it decreases, then that candidate should be kept in.

In forming the group, only those disjuncts that the group members have in common are kept as part of the group; the other, non-shared disjuncts presumably belong to other

[15] See Diary Part Four, pages 13–25.

word-senses. This algorithm performs word-sense disambiguation. If one includes disjuncts shared by a majority, then this algorithm also performs generalization: it moves from particulars to generalities. Some words may have been seen in some contexts; others in different, but generally similar contexts. In admitting these disjuncts into the common group, all of the group members gain the ability to engage in these contexts, thus generalizing. The majority need not be a strict 50% majority; it can be a plurality, set at any threshold. This can be understood as the Jaccard index for the group.

To summarize, high MI (or high VI) just indicates general similarity between words. The actual list of shared disjuncts is computed with a Jaccard index. It is this set of shared disjuncts that determine the grammatical behavior of the group.

Conclusion

A collection of research results were described. Research is ongoing. An immediate next step is to evaluate the quality of the resulting grammars. The most important next step is to climb up the hierarchy: to repeat the process, but now looking at multi-sentence, paragraph, and corpus-wide correlations, with the intent of identifying entities and their properties. Equally important, yet equally daunting, is to apply these techniques to vision, sound or other kinds of information streams. The primary limitation to further research is the development of the tools, the software and infrastructure needed to carry out these experiments. Based on the current results, the future looks extremely promising.

References

1. Kahane, S.: The meaning-text theory. Dependency and Valency. An International Handbook of Contemporary Research **1**, 546–570 (2003), http://www.coli.uni-saarland.de/courses/syntactic-theory-09/literature/MTT-Handbook2003.pdf
2. Mihalcea, R., Tarau, P., Figa, E.: Pagerank on semantic networks, with application to word sense disambiguation. In: COLING '04. p. 1126. ACL (2004). https://doi.org/10.3115/1220355.1220517, http://web.eecs.umich.edu/~mihalcea/papers/mihalcea.coling04.pdf
3. Moulton, R., Jiang, Y.: Maximally consistent sampling and the jaccard index of probability distributions. International Conference on Data Mining, Workshop on High Dimensional Data Mining pp. 347–356 (2018). https://doi.org/10.1109/ICDM.2018.00050, https://arxiv.org/abs/1809.04052
4. Sleator, D., Temperley, D.: Parsing english with a link grammar. Tech. rep., Carnegie Mellon University Computer Science technical report CMU-CS-91-196 (1991), http://arxiv.org/pdf/cmp-lg/9508004
5. Vepstas, L.: Gradient decent vs. graphical models (2018), https://github.com/opencog/learn/learn-lang-diary/skippy.pdf, unpublished
6. Vepstas, L.: Purely symbolic induction of structure (2022), https://github.com/opencog/learn/raw/master/learn-lang-diary/agi-2022/grammar-induction.pdf, submitted for publication, AGI-2022
7. Yuret, D.: Discovery of Linguistic Relations Using Lexical Attraction. PhD thesis, MIT (1998). https://doi.org/10.48550/arXiv.cmp-lg/9805009, http://www2.denizyuret.com/pub/yuretphd.html