Grammar Induction - Experimental Results

Linas Vepštas

February 2022

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.The language-learning effort involves research and software development to implement the ideas concerning unsupervised learning of grammar, syntax and semantics from corpora. This document provides a summary of results observed to date.

Introduction

The goal of the OpenCog Learning project is to induce structure from raw observational data. In the most naive sense, this would be visual or auditory data. In a more sophisticated sense, "observational data" should be taken to be data that might already have some structure in it, 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. The words "grammatical structure" are meant to encompass morphological structure, syntactic structure, and lower reaches of semantic understanding, as far as this can be pushed.

The general theory of how this is to be done is sketched out in a companion article.[1] 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 appears to capture the syntactic structure of the text reasonably well.[2] This parse tree can be disassembled into individual "jigsaw puzzle pieces" which encode the syntactic structure of the parse.[3, 4] Individual jigsaw pieces encode the local syntactic environment of a given word. Collections of such jigsaw pieces 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.[5, 6]

This text provides reports on the algorithms developed to perform the above, and on the statistical properties of the resulting graphical networks. This report is a summary, encompassing 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 so far haven't been. In the "downwards" direction, work on morphology, needed for Indo-European languages, and segmentation, needed for Chinese. Further down, the implementation of the same ideas to extract phonetic structure and perform general audio processing. In the upwards direction, one could could search for high-MI pairs across sentences, both to resolve references and to detect entities. The "syntactic" relationships between entities presumably encodes their semantic properties. Such syntactic relations can also be extracted between different sense streams, say, between words and blueprints/diagrams, or words and photographs, or even perhaps big-data settings such as computer vulnerability databases, or financial news and stock prices. 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 actual experiments; the current limitation to carrying these out is the development of a suitable software infrastructure.

The remainder of this paper focuses on the narrow tasks defined above, in the same order as 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? The similarity between words can be judged with assorted different measures; how do these behave, and how does clustering and word-sense disambiguation proceed, in practice?

[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 to examine.

Software Infrastructure

The software infrastructure used to perform the above can be 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, it is then straight-forward to compute marginal probabilities, conditional probabilities, and other typical statistical quantities. This is, of course, quite ordinary: what is new here 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 mathematics and statistics packages do not provide such tools for sparse data. The other difference between the AtomSpace and conventional statistical tools is that the rows and columns are themselves distinct collections of subgraphs of some larger graph. Given some large graph of data, one can slice-n-dice it in any number of ways, and ask how portions of it correlate, by imposing a matrix structure on the parts of interest. To put those parts under a microscope.

During the pair-counting stage, the rows and columns of the matrix are words, the left and right words, respectively, of an ordered word pair; the matrix is approximately square. During the word-sense disambiguation and syntactic clustering stage, the rows are individual words, and the columns are the connector sequences witnessed for those words. The connector sequence is thus an example of a non-trivial subgraph. The matrix is rectangular; there is typically an order of magnitude more connector sequences than there are words: this is simply because each connector sequence is a context for a word: it resembles a skip-gram in many ways.[7]

Pair Counting

How do I count thee? Let me show the ways I love most.

Obtaining a suitable corpus for exploring the English language is not as simple as it seems: it is not enough to train on a dump of Wikipedia articles. This is because Wikipedia is descriptive, and has a paucity of verbs, and a surfeit of proper nouns and technical terms. Verbs such as run, jump, catch: the verbs of everyday outdoor adventure are missing. This can be remedied by working with a corpus of young adult adventure literature. Such a corpus is not readily available, but can be pieced together by working with Project Gutenberg publications and publicly licensed fan fiction.

Some minimal amount of cleanup can be useful, but it doesn't seem to be critical: if some specialized markup (say, HTML markup) or the occasional table, figure caption or index squeezes through, it should not matter much: after all these also have structure, and the goal of the project is to discern structure.

The size of the vocabulary increases as the size of the corpus increases. This is because of the Zipfian distribution of words: it will always be the case that about half of the vocabulary is observed only once: unusual given names, obscure geographical place names, foreign language loan words.

Word-pair counting can be done in several different ways: one is to count each word-pair occurring within a window of a fixed width, say, of width six. Another possibility is to count the word-pairs occurring in a completely random planar parse of a sentence. There does not appear to be much difference between these two styles of counting, although the second style seems more appealing on theoretical grounds.

A good way to characterize the number of observed word-pairs is by means of the sparsity: the $\log_2$ of the fraction of observed word-pairs vs. all possible word-pairs. Hewing to an information-theoretic foundation, other characteristics include the left and right entropies, and the total mutual information. This is somewhat more interesting than the more conventional statements about the average degree of a vertex in a graph, or the average "hubbiness" (root-mean-square of the degree). This is because of the Zipfian distribution of words: given this, most of the graph statistics follow. The mutual information is more interesting, as it is a key ingredient to an information-theoretic foundation. A summary and some graphs follow.

[2] See GitHub opencog/learn
[3] See GitHub opencog/atomspace
[4] See AtomSpace Matrix API

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: for example, the period never occurs at the start of a sentence; 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 so is expressed in bits. It indicates what fraction of all possible word-pairs are actually observed. Note that 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

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

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. Language is very highly structured. The goal of the learning project is to amplify the information content; to extract information, to data-mine it.

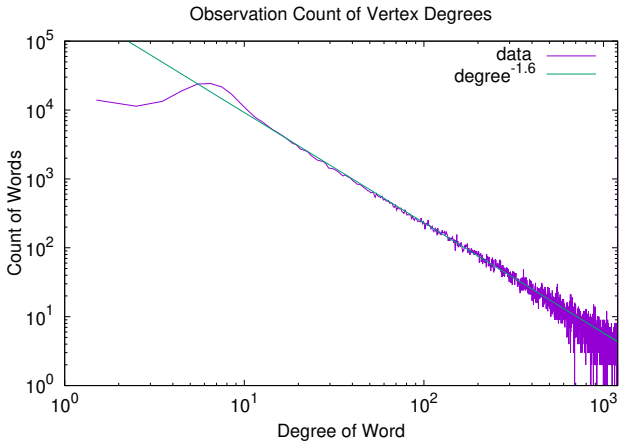
Remarkably, the overall numbers appear to be language-independent; almost identical values are seen for Mandarin Chinese text.[6] To perform this counting, each Hanzi is taken to be a "word". This is not technically correct, as Chinese words may consist of one, two, three or more Hanzi. However, segmentation 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.

[5] See Diary Part Two, page 75 and Diary Part Three, page 7.

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

Sample-size Effects

The distribution of word-pairs is conventionally Zipfian, with an exponent of $\gamma \approx 1.0$, and is not shown. The focus here is on sample-size effects. The graph below shows the observation count for vertex degrees, for dataset 3 above. Consider each word to be a vertex, and each observed word-pair an edge. This forms a graph. The degree of a vertex is the number of edges attached to it. What is the distribution of degrees in the graph?

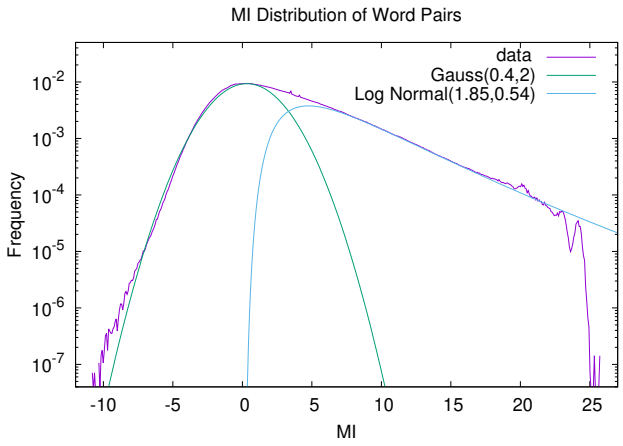


Clearly, the distribution is Zipfian, with exponent $\gamma \approx 1.6$. There are two notable sample-size effects. At the left-hand side of this graph are words that participate in only a handful of word-pairs. This dataset has 142K words in it; the graph shows that more than 100K of them participate in ten or fewer word-pairs. The flattening on the left hand side indicates that these are wildly under-sampled. What are these words? Sample inspection shows that many of these are junk: bad punctuation, badly-escaped quotes, typos, stray markup mixed in together with foreign-language words, obscure place-names and other rarities. It is reasonable, before proceeding to later stages of grammatical induction, to label these words as "junk" and trim them away. The result is a dramatically smaller dataset. This is a good thing: it occupies less RAM and storage; later algorithmic processing runs faster. But does trimming damage information? Apparently not; trimming to remove more than half the words changes the sparsity not at all, lowers the entropy by five percent, and lowers the MI by about a percent.

At the right-hand side of the graph, a risk to histogramming is exposed. These are words that appear in thousands of distinct pairs. Yet, for any given, fixed degree, there are only a small handful of words having exactly that degree, and sometimes even zero words. The higher the degree, the more likely that there are zero words having that degree. This also is a kind-of under-sampling: when generating histograms of distributions, the lack of data in this range will also affect slopes and with it the conclusions about the nature of the dataset. Words with the very highest degree are punctuation: the period, comma and dash, and the determiner "the": these appear in pairs with more than 2/3rds of the other words in the dataset.

MI Distribution

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



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. Perhaps a formal expression for the common-mode noise is easily derived, given a fixed vocabulary size and number of samples. Theories of why the remainder would be a log-normal distribution are unknown to the author.

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.[2] One considers all possible 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 number of different algorithms for finding this tree exist; they all give reasonable results, and do not appear to affect overall results. 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. In LG, the necessity for grammatical agreement between nearby words introduces a stringency and rigidness to the resulting parses, resulting in a greater rejection of bad parses. This result should carry over to the present case, and should 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. Following the LG convention, these connector sequences are called disjuncts (for short), as they are disjoined when considering an LG parse. Denoting the word as w and the disjunct as d, this results in word-disjunct pairs (w,d). Parsing a large corpus this way, the counts are accumulated in the database, to give a count $N(w,d)$. In the current codebase, the counting is done without weighting; one could consider also counting with weighting, giving higher weight to those disjuncts that seem confident. Anyway, given the counts $N(w,d)$, one can again play the matrix game, and examine marginal and conditional probabilities, and the word-disjunct MI.

The first thing one notices is that there are vastly more disjuncts than there are words: two orders of magnitude more! Next, one notices that the vast majority of disjuncts are observed only once. This raises the question: what is signal and

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

what is noise? This can be explored through trimming, by removing words, disjuncts and word-disjunct pairs whenever their observation counts fail to pass a minimal threshold.

Results are summarized in the table below.[8] The column labels indicate the trimming thresholds; in the rightmost column, all words with an observation count of less than 160 were removed; all disjuncts with an observation count of less than 32, and all word-disjunct pairs with a count of less than 20. 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	20-4-2	40-8-5	80-16-10	160-32-20
$\log_2 N_{\text{words}}$	18.526	13.395	12.405	11.548	10.720
$\log_2 N_{\text{disjuncts}}$	24.615	18.313	17.065	15.905	14.737
$\log_2 D_{\text{Tot}}$		19.059	17.705	16.517	15.301
Sparsity		11.903	11.764	10.937	10.156
Rarity		3.205	2.970	2.790	2.572
$\log_2 N_{\text{Tot}}/D_{\text{Tot}}$					
Total Entropy		13.326	9.6489	6.6372	3.5924
Left Entropy					
Right Entropy					
MI					

Notable in this data is that even modest trimming removes the vast majority of words and disjuncts in the dataset. The total entropy is strongly dependent on the trim; the sparsity and rarity a good bit less so. 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.

TODO:

- Graph of (w,d) MI distribution

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, and the application is limited to predicting the next words in a sentence.

How should similarity be compared? The workhorse of conventional neural net approaches is the cosine distance. Experimentally, this appears to be the worst-possible way of determining similarity.[9] Other possibilities include the Jaccard distance, and several weighted variants of the Jaccard distance. One of these has been shown to be optimal, in that it maximizes all possible similarities.[8] All of these do quite well. In keeping with the information-theoretic theme, using the mutual information between two vectors seems quite 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.

[8] The raw data is in Diary Part Two page 58.

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

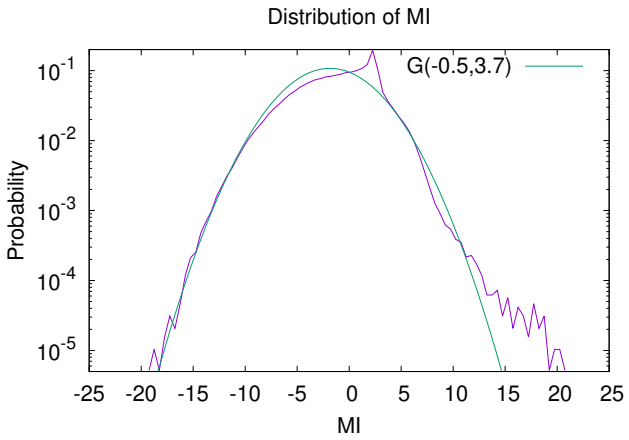
How well does this work? Fantastically well; creating a list of the top-100 most similar words, as ranked by MI, provides excellent results, excellent in the sense that just looking at the list really are either synonyms, or are obviously grammatically similar. But then a problem becomes evident: all of the high-MI word pairs involve rare, infrequent words. If one is to build a syntactical lexis, one really wants to 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.[10]

The graph below shows the distribution of MI similarities for the top-ranked (most frequent) 1200 words.[11] 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 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.[12]

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

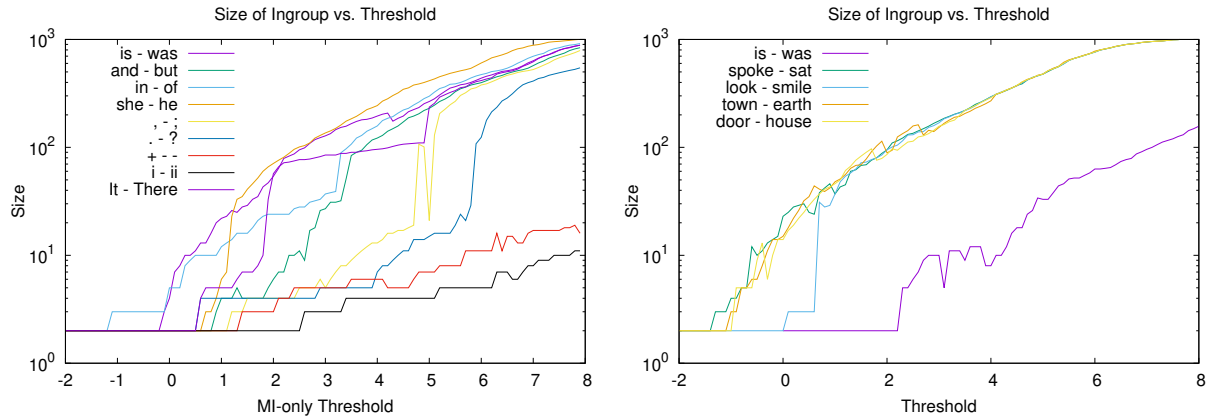
The clusters above are sets of words, all of which have a high pairwise-VI amongst themselves. Most of these are obvious. The cluster of capitalized words is perhaps surprising: but these are all sentence-starters: they are similar because they all start occur first, and start similar sentences.

One can form such clusters quite naively, by searching for cliques of similar words. However, to get adequate word-sense disambiguation, one must be more sophisticated. This is, because when one forms a cluster, one should include only this disjuncts which all of the words share in common, and exclude those that do not. A canonical example of this is the word "saw", which can be the past tense of the verb "to see", and a cutting implement. In forming a cluster with other

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

viewing (or listening) verbs, one wishes to accept only the disjuncts pertinent to such verbs. Excluding the others leaves behind the disjuncts for the other verb senses, in this case, those for cutting tools.

An algorithm that appears to accomplish this can be understood as implementing "exclusive club – common-interests" membership.[13] Starting with the pair of words having currently the highest possible VI, one searches for additional words with a high VI. These form the candidate members to the club. It is useful to have a reasonably large club, and so one can set a threshold, admitting as candidates all those above the threshold. As the threshold 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 thigh 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 kept out of the club. If it decreases, then that candidate should be kept.

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 word-senses. Thus, this algorithm appears to solve the word-sense disambiguation problem.

If one includes disjuncts shared by a majority, then this algorithm also solves the generalization problem; that is, of moving 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.

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

References

[1] Linas Vepstas, "Purely Symbolic Induction of Structure", , 2022, URL https://github.com/opencog/learn/raw/master/learn-lang-diary/agi-2022/grammar-induction.pdf.

[2] Deniz Yuret, *Discovery of Linguistic Relations Using Lexical Attraction*, PhD thesis, MIT, 1998, URL http://www2.denizyuret.com/pub/yuretphd.html.

[3] Daniel Sleator and Davy Temperley., *Parsing English with a Link Grammar*, Tech. rep., Carnegie Mellon University Computer Science technical report CMU-CS-91-196, 1991, URL http://arxiv.org/pdf/cmp-lg/9508004.

[4] Daniel D. Sleator and Davy Temperley, "Parsing English with a Link Grammar", in *Proc. Third International Workshop on Parsing Technologies*, 1993, pp. 277–292, URL http://www.cs.cmu.edu/afs/cs.cmu.edu/project/link/pub/www/papers/ps/LG-IWPT93.ps.

[5] Rada Mihalcea, et al., "PageRank on semantic networks, with application to word sense disambiguation", in *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, 2004, pp. –, URL http://web.eecs.umich.edu/~mihalcea/papers/mihalcea.coling04.pdf.

[6] Rada Mihalcea, "Unsupervised Large-Vocabulary Word Sense Disambiguation with Graph-based Algorithms for Sequence Data Labeling", in *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Morristown, NJ, USA, 2005, pp. 411–418.

[7] Linas Vepstas, "Gradient Decent vs. Graphical Models", , 2018, URL https://github.com/opencog/learn/learn-lang-diary/skippy.pdf.

[8] R. Moulton and Y. Jiang, "Maximally Consistent Sampling and the Jaccard Index of Probability Distributions", *International Conference on Data Mining, Workshop on High Dimensional Data Mining*, 2018, pp. 347–356, URL https://arxiv.org/abs/1809.04052.