Purely Symbolic Induction of Structure

Linas Vepštas

February 2022

Abstract

Techniques honed for the induction of grammar from text corpora can be extended to visual, auditory and other sensory domains, providing a structure for such senses that can be understood in terms of symbols and grammars. This simultaneously solves the classical "symbol grounding problem" while also providing a pragmatic approach to developing practical software systems that can articulate the world around us in a symbolic, communicable fashion.

Introduction

The symbolic approach to cognition is founded on the idea that observed nature can be categorized into distinct entities which are involved in relationships with one another. In this approach, the primary challenges are to recognize entities, and to discover what relationships there are between them.

The recognition problem is to be applied to sensory input. That is, we cannot know nature directly, as it is, but only by means of observation and sensing. Conventionally, this can be taken to be the classical five senses: hearing, touch, smell, vision, taste; or, more generally, scientific instruments and engineered detectors. Such sensors generate collections of data; this may be time-ordered, or simply a jumbled bag of data-points.

Out of this jumble of data, the goal of entity detection is to recognize groupings of data that *always* occur together. The adverb "*always*" here is key: entities are those things that are not events: they have existence over extended periods of time (Heidegger's "Dasein"). The goal of relationship detection is to determine both the structure of entities (part-whole relationships) as well as events (statistical co-occurrences and causation).

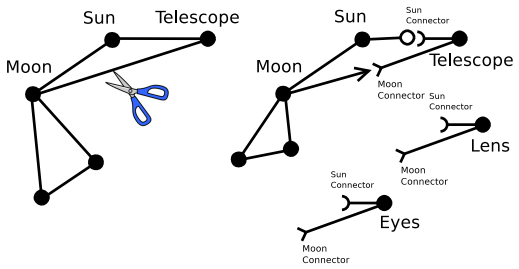
If one is somehow able to detect and discern entities, and observe frequent relationships between them, then the path to symbolic processing becomes accessible. Each entity can be assigned a symbol (thus resolving the famous "symbol grounding problem"), and conventional ideas about information theory can be applied to perform reasoning, inference and deduction. Here, the words "information theory" are taken in the broadest sense: not just signal processing and finite state transducers, but also Bayesian nets, theorem provers and Turing machines; the whole of what is computationally accessible in the current era.

The goal of this paper is to develop a general theory for the conversion of sensory data into symbolic relationships. It is founded both on a collection of mathematical formalisms and also on a collection of experimental results. The experimental results are presented in a companion text; this text focuses on presenting the mathematical foundations in as simple and direct a fashion as possible.

The organization is as follows. First, the general relationship between graphs and grammars is sketched out, attempting to illustrate just how broad, general and all-encompassing this is. Next, it is shown how this symbolic structure can be extended to visual and auditory perception. After this comes a mathematical deep-dive, reviewing how statistical principles can be used to discern relationships between entities. Working backwards, a practical algorithm is presented for extracting entities themselves. To conclude, a collection of hypothesis and wild speculations are presented.

From Graphs to Grammar

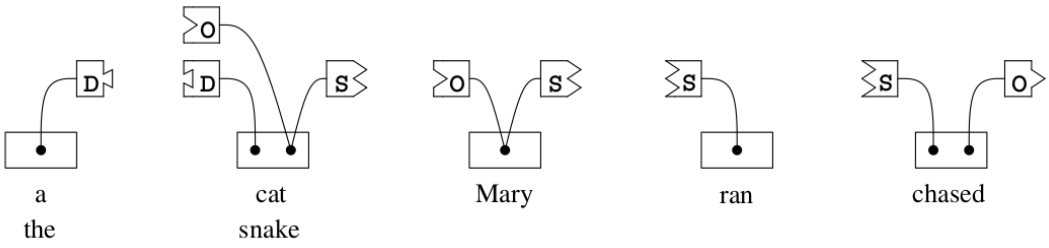
Assuming that sensory data can be categorized into entities and relationships, the natural information-theoretic setting would seem to be graphs: each entity is represented by a vertex, each relationship is represented by an edge. In the most simplistic terms, vertexes are labeled with symbols, and edges with symbol pairs. An example of this kind of naive symbolic graphical relationship is illustrated below.



The figure on the left shows a graph-theoretic sparse graph of relationships between entities. On the right is the same graph, with some of the edges cut into half-edges, with the half-edge connectors labeled with what they can connect to. The connectors are drawn with distinct shapes, intended to convey what they are allowed to connect to. Such vertices, together with a collection of connectors, can be imagined to be jigsaw puzzle pieces, waiting to be connected.

The simplicity of the above diagram is deceptive. In fact, there is a deep and broad mathematical foundation for them. Such jigsaw pieces are the elements of a category-theoretic "monoidal category". The connectors themselves are type-theoretic types. The puzzle pieces themselves are the syntactical elements of a grammar, formally defined. These last three statements arise from a relatively well-known generalization of Curry–Howard correspondence: for every category, there is a type theory, a grammar and a logic; from each, the others can be determined.[1] The mathematics of these relationships is foundational and quite challenging to the uninitiated.

The paradigm of jigsaw pieces has long been known in linguistics, and has been repeatedly rediscovered. The diagram below is taken from the first paper describing Link Grammar, a kind of dependency grammar.[2] Grammatically-valid (syntactically valid) sentences are formed whenever jigsaw connectors can be mated, leaving non unconnected. This fashion of specifying a syntax and a grammar may feel a bit foreign and unusual; be aware that such grammars can be automatically (i.e. algorithmically) transformed into equivalent HPSG, DG, CG, LFG, *etc.* style grammars.



A recent rediscovery of the jigsaw-puzzle paradigm can be found in the work of Bob Coecke.[3] Notable results include a linear-time quantum algorithm for parsing natural language.[4] This work is notable in that it provides a good bridge between concepts in linguistics and the hard science of mathematics. It is not the first such; perhaps one of the earliest formal algebraic treatments of linguistics can be found in Marcus.[5]

Compositionality and Sheaves

The naive replacement of entities by vertexes and relationships by edges seems to itself have a problem with well-foundedness. If an entity is made of parts, does this mean that a vertex is made of parts? What are those parts made of? Is there an infinite regress? How might one indicate the fact that some entity has a composite structure? The answers to these questions are resolved by observing that a partially-assembled jigsaw puzzle resembles a singular jigsaw price: it externalizes some number of unconnected connectors, while also describing the connectivity of the fully-assembled territory. This is how the part-whole relationship is established: the "whole" entity is a partially assembled jigsaw; the parts are the individual pieces, and the way that the entity can interact with other entities is through the as-yet unconnected connectors.

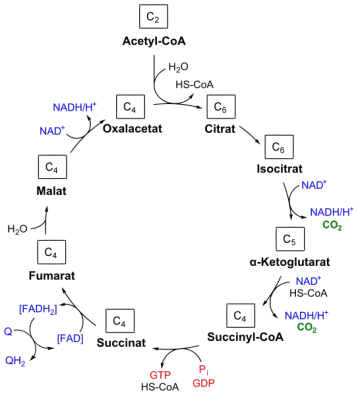
Sheaf theory is the branch of axioms that explores how such part-whole structures behave. The "sheaf axioms" provide a handful of simple rules that describe how a valid sheaf can be created. In simplistic terms, the sheaf axioms describe how jigsaw pieces connect.[6, 7] When sheaf theory is taken in it's full glory, it has been offered up as an alternative to set theory as a foundation for all of mathematics.[8] This is because the sheaf axioms suffice not only to be a foundation for topology (as a generalization of frames and locales), but also for logic (via the extended Curry–Howard

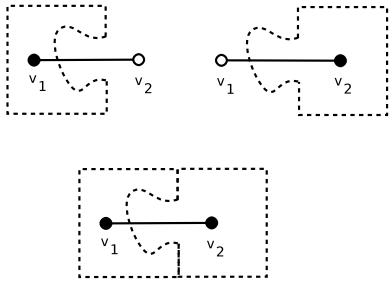
correspondence mentioned above). These are formidable claims with an impeccable mathematical pedigree; the general mathematical theory seems entirely adequate as a foundation for AGI. Armed with this, one can move forward.

In philosophy, the question of part-whole relationships, identity, existence and thing-ness is studied under the name of mereology.[9] Some of the key concepts of mereology include ideas such as "contact", "fastentation", "cohesion", "fusion" and "brutal composition". A shallow understanding of these concepts is enough to convince one that jigsaw pieces fit the bill.[10] Why mention philosophy in the same breath as mathematics? The function of philosophy is to wrestle with vague, phantasmic questions, to thrash about in the fog, hoping to find a purchase on something solid. Whenever such contact can be made, the scientists take over: this is how "natural philosophy" morphed into "physics". Insofar as AGI wishes to be a foundational theory of cognitive-everything, the role of philosophy in suggesting where to look remains invaluable. In this particular case, the ideas of mereology provide the requisite bridge from the abstract to the concrete.

Pervasiveness

After becoming familiar with the jigsaw paradigm, it becomes evident that it is absolutely pervasive. This is illustrated in the two figures below.



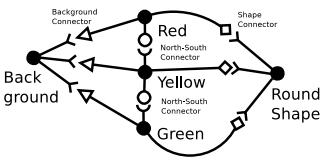


The figure on the left is the Krebs cycle (the citric acid cycle in biochemistry). It has an obvious graphical structure, and it is not difficult to see how it is to be decomposed into jigsaw pieces. What is notable here is that some of the jigsaw pieces find a literal embodiment in the three-dimensional shape of molecules. Diagrams of DNA are often shown with jigsaw connectors standing in for the amino acids ATGC.

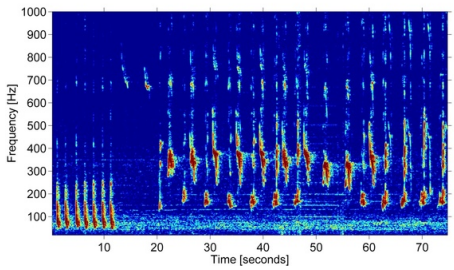
The diagram on the left is more abstract, and is meant to illustrate composition or beta reduction in term algebra. A term is $f(x)$ or an n-ary function symbol $f(x_1, x_2, \cdots, x_n)$. Variables are denoted as $x, y, z, \cdots$ while constants are type instances, such as 42 or the string "foobar". Beta reduction is the act of "plugging in": $f(x) : 42 \mapsto f(42)$. Re-interpreted as jigsaw connectors, the term $f(x)$ is the jigsaw on the left, and 42 is the jigsaw on the right. In order to connect, the two types must match. The type-theoretical type of the variable x must match the type of of what is plugged in. This kind of plugging-in or composition is rampant throughout mathematics. Prime examples can be found in proof theory,[11] lambda calculus,[12] term algebras[13] and model theory.[14]

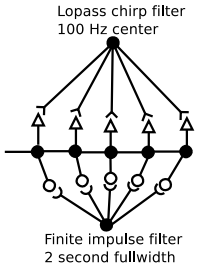
Vision and Sound

Shapes have a structural grammar, too. The connectors can specify location, color, shape, texture. The key point of this structural decomposition is that it is *not about pixels*! The structural decomposition is scale-invariant (more or less, unless some connector fixes the scale) and rotationally invariant (unless some connector fixes direction). The structural grammar captures the morphology of the shape, it's general properties, omitting details when they are impertinent, and capturing them when they are important.



Not only do two-dimensional photographs have a structure grammar, but so do sounds. On the left is a spectrogram of a whale song. It is ostensibly "one-dimensional", with time as the primary dimension, but is more accurately multi-dimensional: the vertical axis shows frequency, the colors encode a third dimension, the intensity.





On the right is a graphical representation of the midsection of the song. It shows the number of repetitions (six), as well as the shape of the frequency distribution (its a chirp, which can be discovered with a chirp filter, a certain kind of digital signal processing filter.) Individual repetitions can be spotted with a finite impulse response filter. The point here is that basic sensory information can also be described in grammatical terms.

Symbolic Learning

In order for a graphical, sheaf-theoretic, grammatical theory of structure to serve as a foundation stone for AGI, there most be a practical algorithm for extracting such structure from sensory data. Such an algorithm is sketched below. It consists of three interacting parts. The first step is chunking, the division of sensory data into candidate entities and candidate interactions that might possibly be identified as entities or interactions by the second step. The second step takes a collection of candidate graphs, splits them into jigsaw pieces, and then attempts to classify jigsaw pieces into common categories, based on their commonalities. The third step is a recursive step, to repeat the process again, but this time taking the discovered structure as the sensory input. It is meant to be a hierarchical crawl up the semantic ladder.

Experimentally, the second step has been the most thoroughly explored. An implementation exists within the OpenCog system,[1] with extensive research diaries logging results.[2] A summary of these results is presented as a companion paper to this one. Explorations of the first and third steps have hardly begun. It is easiest to describe the second step first.

Grammatical Induction

In linguistics, one is presented with a pre-chunked sequence of words; the conversion of raw sound into phonemes and then words is presumed to have already occurred. The task is to extract a more-or-less conventional grammar given a corpus of text. The first step is to perform a Maximum Spanning Trees (MST) parse; the second step is to split the MST parse into jigsaw pieces, and classify those pieces into lexical vectors. The generation of the MST parse requires collecting statistics on a corpus, based on maximum entropy principles; this has a long and deep tradition in Corpus Linguistics, going as far back as Biblical Concordances in earlier centuries, before linguistics was a distinct field of study. Lexical semantics also a deep and broad research literature, much more modern. Of note here is that this vector-based description has more than a few similarities to the theory of neural-nets, and yet is fundamentally different, because it is ultimately lexical (*i.e.* is symbolic.) It is not obvious if deep-learning techniques can be applied to obtain these lexical vectors; what is described below is a simpler, plainer approach.

Maximum Planar Graph Parsing

The MST parsing algorithm is described by Yuret, as follows.[15] Starting with a corpus, maintain a count $N(u,w)$ of nearby word-pairs (u,w). Here, "nearby" usually means "in a window of width six". A frequentist probability $p(u,w) = N(u,w)/N(*,*)$ is just the count of a given word-pair divided by the total count of all word-pairs. The star

[1] See the "learn project" in github.

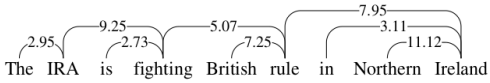
[2] See the diaries in the aforementioned project.

indicates a marginal sum or marginal probability, so that $p(u,*) = \sum_w p(u,w) = \sum_w N(u,w)/N(*,*) = N(u,*)/N(*,*)$. Given these frequencies, defines the Lexical Attraction between word-pairs as

$$MI(u,w) = \log_2 \frac{p(u,w)}{p(u,*)\,p(*,w)}$$

This lexical attraction is just the mutual information; it has a somewhat unusual form, as word-pairs are necessarily not symmetric: $(u,w) \neq (w,u)$ and conventional texts on probability theory usually do not address this non-symmetric situation. Note that the MI may be negative! The range of values depends on the size of the corpus; for a "typical" corpus, it ranges from -10 to +30.

Given a table of $MI(u,w)$ obtained by counting, one obtains an MST parse of a sentence by considering all possible trees that connect all of the words, and selecting the one tree that has the largest possible total MI. An example of such a tree is shown below, taken from Yuret's thesis. The numbers in the links are the MI between the indicated words.

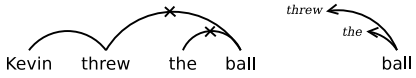


There exist a number of fast algorithms for obtaining maximal spanning trees. Variants include those that generate only planar trees, or even planar graphs (graphs with loops, but no intersecting links). Maximal planar graphs (MPG) appear to offer experimentally-observable advantages over trees, as they appear to constrain the grammar more tightly. In this sense, they offer some of the advantages seen in catena-based linguistic theory.[16]

At any rate, the point here is that the MST parses are linguistically plausible: it is fairly straightforward to verify that they correspond, more or less, to what trained linguists would write down for a parse. The accuracy is reasonably high. For the present algorithms, perfect accuracy is not needed, as later stages make up for this. Yuret indicates that the best results are obtained when one accumulates at least a million sentences. This is not outrageous: work in child psychology indicates that human babies hear several million sentences by the age of two years.

Lexical Entries

Given an MST or MPG parse, the lexis is constructed by chopping up the parse into jigsaw pieces, and then accumulating the counts on the jigsaw pieces. This is shown below.



There are multiple different notations possible for the above. There is a tensorial notation, popular in quantum approaches; the lexical entry would be written as ball : $\left|\overleftarrow{\text{the}}\right\rangle \otimes \left|\overleftarrow{\text{throw}}\right\rangle$ while in Link Grammar, it is denoted as `ball` : `the` $-$ & `throw` $-$ where the minus signs indicate connections to the left. The ampersand is the conjunction operator from a fragment of linear logic; it demands that both connector be present. Linear logic is the logic of tensor algebras (by the aforementioned Curry–Howard correspondence.) Unlike tensor algebras, natural language has a distinct left-right asymmetry, and so the corresponding logic (of the monoidal category of natural language) is just a fragment of linear logic. Note that all of quantum mechanics lies inside of the tensor algebra; this explains why assorted quantum concepts seem to recur in natural language discussions.

The connector sequences `the` $-$ & `throw` $-$ or $\left|\overleftarrow{\text{the}}\right\rangle \otimes \left|\overleftarrow{\text{throw}}\right\rangle$ often called "disjuncts": different connector sequences d are disjoined from one-another. Given a word w, a lexical entry consists of all word-disjunct pairs (w,d) together with their observed count $N(w,d)$. Given this count, one may define a frequency $p(w,d) = N(w,d)/N(*,*)$ where this time, $N(*,*)$ is the sum over all word-disjunct pairs. A lexical entry is thus a sparse skip-gram-like vector:

$$\overrightarrow{w} = p(w,d_1)\,\widehat{e_1} + \cdots + p(w,d_n)\,\widehat{e_n}$$

One can use the logical disjunction "or" in place of the plus sign; this would be the "choice" operator in linear logic (as in "menu choice": pick one or another). The basis vectors $\widehat{e_k}$ are one and the same thing as the skip-gram disjuncts $\left|\overleftarrow{\text{the}}\right\rangle \otimes \left|\overleftarrow{\text{throw}}\right\rangle$, just offering a short-hand notation.

Similarity

The lexis generated above contains individual words with connectors to other, specific words. Although each vector is sparse, and the lexis, as a whole, taken as a matrix is sparse, it is still quite large. By contrast, conventional linguistic grammars talk about nouns and verbs and adjectives. Part of the learning process is then to automatically find similar categories: to organize words into groups based on similarities. A very conventional similarity metric is the cosine distance, given as

$$\cos\theta = \vec{w}\cdot\vec{v} = \sum_d p(w,d)\,p(v,d)$$

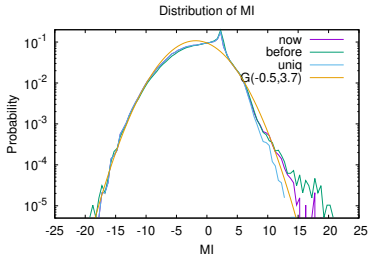
This similarity metric fails. The space spanned by these vectors is *not Euclidean space*! It does not have rotational symmetry! Properly, it is a probability space, a simplex, as one must preserve unit-length probability vectors: $1 = \sum_{w,d} p(w,d)$. The correct information-theoretic similarity is the mutual information:

$$MI(w,v) = \log_2 \frac{\vec{w}\cdot\vec{v}}{\left(\vec{w}\cdot\vec{*}\right)\left(\vec{*}\cdot\vec{v}\right)}$$

where

$$\vec{w}\cdot\vec{*} = \sum_d p(w,d)\,p(*,d)$$

Unlike the MI for word-pairs, this MI is symmetric: $MI(w,v) = MI(v,w)$. Experimentally, the distribution of the MI for word pairs appears to be Gaussian, as shown below.



The above is obtained from MPG parsing and statistics gathering as described above.[3] Note the graph is on a semi-log scale: normal distributions become parabolas when graphed this way.

Classification

Clustering words into grammatical categories based on similarity may seem straight-forward, but can founder on several different details. First, the MI, as defined above, has the property that words with the very highest MI tend to be very infrequent, rare. As a practical detail, one wishes to first cluster the most frequent words. To do this, one must add some kind of offset, so as to recommend common words first. Such an offset is provided by the average of the logarithm of the frequency of the two words. This leads to the definition of

$$\begin{aligned} MI_{\text{ranked}}(w,v) =& MI(w,v) + \frac{1}{2}\left[\log_2 p(w,*) + \log_2 p(v,*)\right] \\ =& \log_2 \frac{\vec{w}\cdot\vec{v}}{\sqrt{\left(\vec{w}\cdot\vec{*}\right)\left(\vec{*}\cdot\vec{v}\right)}} \end{aligned}$$

The shape of the Gaussian above is unaffected, although it is shifted to the right; clearly, the ranking is completely different.

[3] See the Language Learning Diary Part Three, *op. cit.*

Word-sense disambiguation

A second practical difficulty arising during clustering is that of word-sense disambiguation. Once two words have been judged similar enough to be merged into a common class, this does not imply that all disjuncts are to be dumped into that common class, as well. Instead, a given vector is to be decomposed into two: a part to be merged, and a part that is left over. The part to be merged will share disjuncts in common with the other word-vector being merged. The parts that are not shared presumably belong to distinct word-senses. For example, parts of the word-vector for "saw" can be clustered with other cutting tools, while the remainder can be clustered with viewing verbs.

Splitting the word-vectors in this way prevents the use of off-the-shelf clustering algorithms that can be found on the Internet (as these in general perform an all-or-nothing merge.)

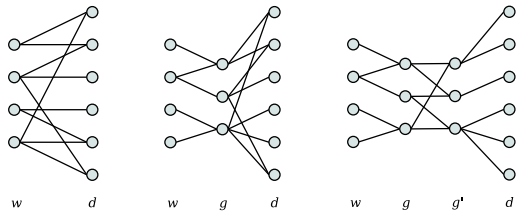
A further difficulty is that connectors must also be merged. The rewriting of connector sequences is subtle, as it affects word-vectors outside of those being merged (the merged connectors might appear "anywhere"). To maintain coherency, "detailed balance" must be preserved: the grand total counts must remain the same both before and after. This is much the same as detailed balance in chemistry, where the grand-total number of atoms (and their specific types) in a chemical reaction is preserved, even as the number of molecules, and the connections change. Again, this is a place where the sheaf axioms make an appearance.

Factorization

The clustering described above can be understood to be a form of matrix factorization. The word-disjunct matrix $p(wd)$ is factorized into three matrices *LCR* as

$$p(w,d) = \sum_{g,g'} p_L(w,g)\, p_C\left(g,g'\right)\, p_R\left(g',d\right)$$

where g is a "word class" and g' is a "grammatical relation". These two are commonly conflated in linguistic theory; here, they will be distinguished. The reason for this is that this is the *de facto* organization of the Link Grammar dictionaries for English, Russian and other languages. Examples of grammatical relations are "subject", "object", "modifier"; examples of word classes are "common noun" or "preposition". The matrices *L* and *R* are very sparse, which *C* is compact, dense and highly connected. This sparse-dense factorization is visualized below.

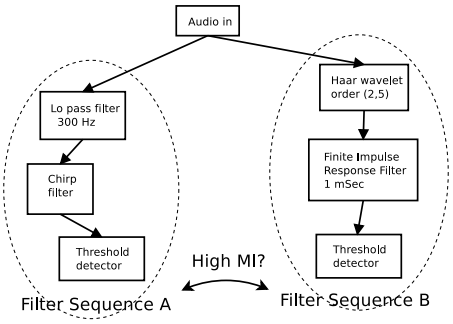


For the current Link Grammar English dictionary, there are about 100K words, 2K word classes, several hundred grammatical relations (LG "macros") and 30 million disjuncts. This dictionary is hand-curated; it is an example of what can be accomplished "by hand".

Neural networks accurately capture the dense, interconnected central region; this is *why* they work. They necessarily perform dimensional reduction on the sparse left and right factors. By erasing/collapsing the sparse factors, neural net become uninterpretable; an opaque mass of wight vectors. Interpretability is all about regaining (factoring back out) the sparse factors! This is exactly what the symbolic learning algorithm does.

Chunking

By working with text, we got lucky. Sounds have been pre-chunked into words. The vocabulary of words is relatively small. Working with raw audio or raw video does not provide such chunked data. How can it be arrived at? One possibility is to carefully engineer hand-crafted transducers, from sounds to phonemes and syllables and words. Another is to automate the creation of such transducers.



The above illustrates a pair of transducers in block-diagram form. Each block corresponds to a digital signal processing (DSP) function. The automation of their discovery can proceed as follows. Randomly generate a collection of a few hundred filter sequences. Each takes audio-in, and generates a single-bit output. Each such filter sequence can be thought of as a "feature recognizer": it detects certain features in the input, responding with a one if the feature is present, else responding with a zero. To join up with the symbolic learning algorithm described above, each such filter sequence can be thought of as a "candidate word". As it was a randomly generated filter, it probably does not correspond to anything meaningful. However, by applying all of these filters on a corpus of sounds, correlations in their output can be observed. Large correlations have a high MI, otherwise not.

To obtain better filter sequences, more meaningful, random perturbations and extensions of the most highly correlated filters can be explored. This is a form of genetic program learning. Each block in the block diagram corresponds to a DSP function, plus it's parameters. Random genetic mutations are those that alter parameters. Genetic cross-overs are those that re-arrange blocks and reconnect parameters. The OpenCog system has already explored this kind of genetic program learning system; it is called MOSES.[17, 18] That particular system is designed for supervised training on a table of inputs; the blocks in that system are logical and arithmetic operators, and a handful of simple functions. The proposal here is to apply a similar training schedule, but one for which the blocks are audio (or video) DSP units, and the table of inputs is replaced by a corpus of audio snippets (or of photographs), and the reinforcement learning aspect is replaced by an entropy-maximization function.

The above is a research proposal; it has not been implemented. There is considerable experience with the genetic program learning subsystem, and it works quite well. The software would have to be completely redesigned to be able to manipulate program trees representing DSP functions.

An unresolved theoretical issue is that of common mode rejection. That is, it is quite possible to learn two very different filter sequences, yet these both detect exactly the same features. For audio, this is less of a problem, as one can compare time-shifted outputs. For still photographs, such as of the stop-light, the goal is to learn three filters, for red, yellow and green disks, and not three different filters all detecting a red disk. Resolving this issue, and doubtlessly many more, is effectively impossible without hands-on experimentation.

The Symbol Grounding Problem and the Frame Problem

An old problem in philosophy (dating back to Socrates) is the symbol grounding problem.[4] When one says the word "chair", what does that mean? One can attempt to make extensional lists of things one can sit on, but that list can never be complete. One can make intensional lists of the properties of a chair; such a list invariably fails to encompass all the possibilities. There is in fact a third, commonly overlooked possibility: that of discovering affordances. What must an object be like, to be sit-on-able? The filter sequence is precisely an affordance-detector.

Lets take a simpler example. If someone says "I hear whistling in the distance", what does the word "whistling" actually mean? How to describe it? What is the grounding for the symbol "whistling"? Well, in the above, the grounding is explicitly manifest: it is a certain kind of hi-pass filter attached to a chirp filter with a certain finite impulse response time. That is what "whistling" is. What else could it possibly be? Is the filter sequence an infallible detector of whistling? Well, no. Obviously, birds are taken in by fake bird calls; doing better requires higher intelligence and better hearing.

Might there be an affordance-detector for chairs? That is, a certain filter that examines photos, and responds with a yes/no answer to "can I sit on that"? Sure. How can it be learned? Direct, embodied experience: one looks at things, and then tries to sit on them. One learns. But what about the Frame Problem? That is, of all the stimuli, all the things going on around, which ones are relevant to actually sitting? Well, but this is precisely what the entropy-maximizing training

[4] See the Stanford Encyclopedia of Philosophy, "Frame Problem" and "Embodied Cognition".

of filter sequences is doing: it is using constructs like mutual information to focus on what is important, and discard what is irrelevant. At this stage of research, this is but hand-waving. What is new here is that the above describes an algorithm that can actually be implemented, and is well-founded on known mathematics and existing, practical software design principles.

Abstraction and Recursion

The development above has focused on a very low level: either parsing in natural language, or sensory input. What about common-sense reasoning, one of the Holy Grails of AGI? It is easily argued that additional methods and techniques are needed to reach such heights. But is that actually the case? The author wishes to argue that the techniques described so far are essentially all that is needed to reach up into the highest levels of abstraction and general intelligence.

To get a feeling for the next few rungs of the ladder, it is easiest to return to the domain of linguistics. In lexical semantics, there is an idea of "lexical implication rules"[19]. These are rules that control how words used in one context can be used in a different context. Can the discovery of these rules be automated? It would seem so: each rule can be described as a jigsaw-piece; if a certain set of connectors match the conditions on the input of the rule, then the rule fires, producing or controlling output.

This works because rules (in the sense of a rule-engine) can be interpreted as jigsaw pieces. Of course, a single rule is just olde-fashioned stimulus-response AI (SRAI). The difference here is that we have a practical mechanism for learning rules (out of "nothing at all") as well as a practical mechanism for assembling them. Jigsaw assembly is parsing: given a set of constraints (a sequence of words) parsing is the act of finding jigsaw pieces that fit the word-sequence. Parsing technologies are well-understood. If a sequence of jigsaw pieces need to be stacked deeper, there are relatively well-understood technologies for that as well: automated theorem-proving. A collection of sequents, such as

$$L\vee\frac{A,\Gamma\Rightarrow C\quad B,\Gamma\Rightarrow C}{A\vee B,\Gamma\Rightarrow C}\qquad R\forall\frac{\Gamma\Rightarrow A\,[x/y]}{\Gamma\Rightarrow\forall xA}$$

can be understood as certain specific jigsaw pieces, with the premises above the line being "input" connectors, conclusions below the line being connectors as well, and theorem-proving (and all of its subtleties, such as cut elimination and efficiency in general) is the assembly of these jigsaw pieces in a syntactically valid fashion.

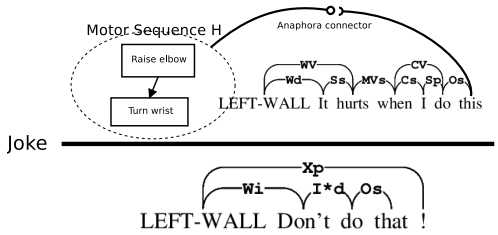
The concept of lexical implication rules generalizes to that of "lexical functions" (LF) of Meaning-Text Theory (MTT).[20] The MTT is a very well-developed theory of the "semantic" layer of linguistics, sitting atop the syntactic layer described above. What makes it "semantic", and how might it be learned? MTT describes the "semantic" layer as a collection of concepts which can be expressed through various alternative word-choices. How might these be learned? Such an algorithm is described by Poon & Domingos[21]. Their algorithm learns synonymous phrases. it is not unlike what is described above; the only issue there is that they make the disastrous choice of using lambda-calculus notation for describing jigsaw pieces! This appears to lead to pointless complexity and confusion; a tensorial, type-theoretic function would have been more appropriate.

Another example for deriving meaning based on connectivity can be found in the Mihalcea algorithm for associating WordNet word senses to individual words in a given sentence.[22] Her algorithm is phrased in terms of the Page-Rank algorithm, to flow weights along edges along a fully connected graph. Looked at more carefully, this is a probabilistic parsing algorithm: discovering what is connected to what by altering weights on the matrix, until certain edges are ruled out, are deemed "unconnectable".

Texts are not single sentences: texts are paragraphs and pages. To navigate through a text, one must perform anaphora resolution, or, more generally, reference resolution, determining when two words in a text refer to the same entity. But, in a certain sense, we already have that: the pair-MI algorithm given earlier can provide a rough cut for how related two things are. The relation can be sharped by discovering context via MST parsing (although at this level, it more closely resembles the Michalcea algorithm). Specificity can is then further heightened in the clustering step. Rinse, repeat.

Common Sense

Can this be used to learn common sense? I believe so. How might this work? Let me illustrate by explaining an old joke: "Doctor Doctor, it hurts when I do this! Well, don't do that!". The explanation is in the form of a rule, such as those drawn in proof-theory. The thick horizontal bar separates the premises from the conclusions. It is labeled as "Joke" to indicate what kind of rule it is.



The "sequent" here is the anaphora connector, which connects the word "this" the a specific motor sequence. Which motor sequence? Well, presumably one that was learned, by automatic process, to move a limb. All of the components of this diagram are jigsaw pieces. All of the pieces can be discovered probabilistically. All of the connectors can be connected probabilistically. The learning algorithm shows how to discern structure from what is superficially seems like a chaotic stream of sensory input. Common sense can be learned.

References

[1] John C. Baez and Mike Stay, "Physics, Topology, Logic and Computation: A Rosetta Stone", *Arxiv/abs/09030340*, 2009, URL http://math.ucr.edu/home/baez/rosetta.pdf.

[2] 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.

[3] Bob Coecke, "Quantum Links Let Computers Read", *New Scientist*, 2010, URL http://www.cs.ox.ac.uk/people/bob.coecke/NewScientist.pdf.

[4] William Zeng and Bob Coecke, "Quantum Algorithms for Compositional Natural Language Processing", *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, 221, 2016, URL https://arxiv.org/abs/1608.01406.

[5] Solomon Marcus, *Algebraic Linguistics; Analytical Models*, Elsevier, 1967, URL https://monoskop.org/images/2/26/Marcus_Solomon_editor_Algebraic_Linguistics_Analytical_Models_1967.pdf.

[6] Linas Vepstas, "Sheaves: A Topoloogical Approach to Big Data", , 2017, URL https://arxiv.org/abs/1901.01341, arXiv abs/1901.01341.

[7] Alberto Speranzon, et al., "Abstraction, Composition and Contracts: A Sheaf Theoretic Approach", *ArXiv abs/180203080*, 2018, URL https://arxiv.org/abs/1802.03080.

[8] Saunders Mac Lane and Ieke Moerdijk, *Sheaves in Geometry and Logic*, Springer, 1992.

[9] Achille Varzi, "Mereology", *The Stanford Encyclopedia of Philosophy (Spring 2019 Edition), Edward N Zalta (ed)*, 2019, URL https://plato.stanford.edu/archives/spr2019/entries/mereology/.

[10] Linas Vepstas, "Mereology and Postmodernism", , 2020, URL https://github.com/opencog/atomspace/raw/master/opencog/sheaf/docs/mereology.pdf, unpublished.

[11] A. S. Troelstra and H. Schwichtenberg, *Basic Proof Theory, Second Edition*, Cambridge University Press, 1996.

[12] H. P. Barendregt, *The Lambda Calculus, Its Syntax and Semantics*, North-Holland, 1981.

[13] Franz Baader and Tobias Nipkow, *Term Rewriting and All That*, Cambridge University Press, 1998.

[14] Wilfrid Hodges, *A Shorter Model Theory*, Cambridge University Press, 1997.

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

[16] Timothy Osborne, et al., "Catenae: Introducing a Novel Unit of Syntactic Analysis", *Syntax*, 15, 2012, pp. 354–396.

[17] Moshe Looks, *Competent Program Evolution*, PhD thesis, Washington University St. Louis, 2006.

[18] Moshe Looks, "Meta-optimizing semantic evolutionary search", in *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings, London, England, UK, July 7-11, 2007*, edited by Hod Lipson, ACM, 2007, p. 626.

[19] N. Ostler and B.T.S.Atkins, "Predictable Meaning Shift: Some Linguistic Properties of Lexical Implication Rules", *Proceedings of the First SIGLEX Workshop on Lexical Semantics and Knowledge Representation*, 1991.

[20] Sylvain Kahane, "The Meaning-Text Theory", *Dependency and Valency An International Handbook of Contemporary Research*, 1, 2003, pp. 546–570, URL http://www.coli.uni-saarland.de/courses/syntactic-theory-09/literature/MTT-Handbook2003.pdf.

[21] Hoifung Poon and Pedro Domingos, "Unsupervised Semantic Parsing", in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Singapore, 2009, pp. 1–10, URL http://www.aclweb.org/anthology/D/D09/D09-1001.

[22] 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.