Aggregate Satisfaction

Linas Vepštas

Version of 29 April 2025 (Started on 17 April 2025)

Abstract

Diary entry, attempting to work out some ideas. Written in pseudo–chronological order. Stream–of–consciousness, kind of. Moving forward, I hope. Maybe going in circles.

The goal was to try give a precise conception of how to take a collection of statements or assertions, and to satisfy as many of them as possible, all at the same time, while keeping contradictions under control.

Introduction

What follows will be free–form unstructured writing about aggregate satisfiability. It is an attempt to reconcile a number of quite common and inter–related ideas, and to develop a mathematical framework for them. The key ingredients are:

- The Ising model as a motivational example. This pulls in assorted concepts from physics that can be elaborated on.
- Link Grammar parsing, in which syntactic jigsaw pieces are assembled to create parses, called "linkages", that are assigned a score. The lower the linkage cost, the more likely correct linkage is correct.
- Aggregate satisfiability, in that satisfiability refers to discrete logical expressions, while aggregate suggests that there is some mean that indicates the best possible solution.

There are also some other inter–related concepts, which may of may not the ideas to be developed here. These include ideas like:

- Markov Logic Networks: perhaps what I intend to describe will end up being a Markov Logic Network. Or maybe not.
- Hopfield networks: these have commonality with regard to assorted physics concepts. Maybe they connect with what I intend to describe, and maybe they don't.
- Mean field theory. Who knows. A knee–jerk reaction is that mean–field theory will describe the rsults. I suppose. Seems plausible.

- Ensembles. Of course, everything to be considered is meant to be an ensemble. But I won't list this as a central point.

This text is incomplete and imprecise. So here we go.

Ising Model

The part of the Ising Model that is interesting here is the Hamiltonian. It is a sum of pair–wise interactions of nearest neighbors:

$$\mathscr{H} = -\sum_{ij} J_{ij}\sigma_i\sigma_j - \mu\sum_j h_j\sigma_j$$

with σ_i the spins. I won't explain this. I copied it from Wikipedia. Read that article if you don't know what this is.

Motivation

Why start this essay with the Ising Model? For several reasons:

- Imagine one has a collection of "facts", "assertions", or "constraints", each of which can be True or False. Map "True" to "spin up" and "false" to "spin down". The if there is a relationship between two assertions σ_i and σ_j then set $J_{ij} \neq 0$ and set it positive, if both have to be true at the same time, else set it negative.

- For the conventional Ising model on a 2D grid, $J_{ij} \neq 0$ only when σ_i and σ_j are nearest neighbors, in which case $J_{ij} = J$. For zero temperature and zero magnetic field, the Hamiltonian is obviously minimized by having all spins aligned. The model is interesting because of it's thermodynamic behavior, at finite temperature, in which thermal energy is allows some of the spins to be non–aligned.

- The attempt being made here is to draw an analogy to systems of logic, such as the Boolean SAT problem. By convention, Boolean SAT is solved with DPLL. The algo there recasts J_{ij} as a graph adjacency matrix. The corresponding graph is then some graph with cycles, and a bunch of trees attached to it. The trees are trimmed away, because they can be uniquely determined by the root. This leaves an interconnected kernel. Then, conventionally, one performs a search, possibly an exhaustive search over that kernel, and determine if it is satisfiable. Various techniques can be attempted to speed that search.

- For example, there may be a circular chain of connections, $\sigma_a \to \sigma_b \to \cdots \to \sigma_p \to \neg\sigma_a$ which cannot be satisfied because it forces a self–contradiction. So strict zero–temperature Boolean SAT indicates this problem is unsatisfiable. However, by using simulated annealing, one might be able to find find truth assignments that satisfy most of the constraints.

- The problem of satisfiability reared its head in Dough Lenat's CYC; the solution there was to propose "microtheories" which were known to be self–consistent,

and then hop between these. The suggestion being made here is that such inconsistencies can be thought of as "lattice frustrations", and mostly resolved through annealing.

- In the human world of common–sense judgments, it is clear that most humans insist that they are "rational" and "logical", even though it is clear that this cannot possibly be true, if one tries to drill down and interrogate the foundations of beliefs. Yet, except for extreme cases involving various forms of mental illness, most humans mostly do have a collection of beliefs that they perceive to be consistent. Those beliefs appear to be malleable via persuasion, propaganda, cult indoctrination, etc. The proposal here is that perhaps a Hamiltonian approach to a description of belief networks can give some sense of the grounded–ness of those beliefs.

- The relational weights J_{ij} can be made dynamic. Thus, for example, inconsistent beliefs about what time of day one left the house might be utterly inconsequential, unless this plays a role in a criminal trial, in which case the truth/falsehood coupling J_{ab} between a statement σ_a of when one left the house, and the statement of σ_b of the defendant becomes very important. A large enough value of J_{ab} can make the prosecutions case fall apart or to firmly establish guilt. A much softer variant of this can occur during attempts to solve quotidian engineering problems, where one wishes to make trade–offs between different design parameters or criteria: the assigned weights J_{ab} can be raised or lowered to taste.

The above is a motivational sketch of why the Ising Model can be interesting and useful in the pursuit of "common sense" inference or during evidence gathering. It provides a formalism in which "aggregate satisfiability" can be discussed.

There are several problems with the direct use of the Ising Model:

- The matrix entries J_{ij} are (by convention) symmetric: $J_{ij} = J_{ji}$ whereas conventional logic is causitive or inferential; one writes $P \to Q$ to say "P implies Q" or "P causes Q to happen" or "P believes that Q is true".

- Logical statements typically have multiple antecedents: $A, B \to C$.

- One may wish to strongly couple inferences: that is, to write $A \to B, C$. Now, in conventional logic, this can be decomposed into two statements: $(A \to B, C) \implies (A \to B) \wedge (A \to C)$ which is fine for crisp logic, but a problem if one is looking for approximate satisfiability. That is, one may want to strongly avoid the situation of $(A \to B) \wedge \neg (A \to C)$, choosing instead to negate both B, C or neither.

A lesser issue, but one that we should get out of the way, is that of hierarchical satisfiability. That is, one may have strongly interacting clusters, with only weak connections between the clusters. That is, the J_{ij} are large, when i, j both belong to the same cluster, but are small when i, j are in different clusters. The Ising Model doesn't prevent this, but perhaps there's some better way of describing natural clustering.

Jigsaws

In light of the criticisms above, the suggestion is to move to a jigsaw or tiling approach to relationships. The jigsaw paradigm provides a natural setting for writing expressions such as $A, B \to C, D$ as well as many other more complex relationships. For example, boolean logic is two–valued: logical variables can assume the values of true or false. In intuitionistic logic, three states are available: true, false or unknown. In typed term algebras, or typed lambda calculii, the variables are typed, and terms, atomic sentences and so on can only be composed if the types match.

The prototypical setting for jigsaws is Link Grammar. See next section..

Link Grammar

I won't explain what Link Grammar (LG) is, or how it works. Read the docs. (Note: I do end up explaining much of it, further below. It's unavoidable.)

I want to write down a Hamiltonian for it, which is something that the Link Grammar docs do not do. It is analogous to the Ising Model Hamiltonian:

$$\mathscr{H} = -\sum_{ijk\cdots} C_{ijk\cdots} w_i w_j w_k \cdots$$

The w_i are the words in the sentence. Written out here, in order to resemble the Ising Model. The $C_{ijk\cdots}$ is meant to be the LG cost, a single real–valued floating point number, of the the LG disjunct $ijk\cdots$.

The analogy here is strained, and the notation is inadequate. This is a brainstorm idea, so we'll fix the problems here later. If the idea pans out.

What are the w_i really? One interpretation is to say $w_i = +1$ always, no matter what, and then the sum is simply a sum over the costs of the disjuncts appearing in the sentence. This is the conventional LG model. Another interpretation is to write $w_i = +1$ if that word appears in position i in the sentence, else it is zero. To arrive at this, we have to be more carefully with the notation: Define

$$\delta(w, v) = \begin{cases} 1 & \text{if } w = v \\ 0 & \text{if } w \neq v \end{cases}$$

where w, v are specific words in the lexis. Then the Hamiltonian is

$$\mathscr{H} = -\sum_{uv\cdots} C_{uv\cdots} \delta(u, w_i)\, \delta(v, w_j)\, \delta(\cdots) \cdots$$

so that the subscript i encodes the position of a word in a sentence, and w_i is the word in that position.

This interpretation is unsatisfying, and wrong, the analogy doesn't work. First of all, the spins in the Ising model can flip up and down, whereas the observed word–order in a given sentence is fixed, inviolable. The above picture could work for sentence generation, where we are considering how to generate some sentence, but that is not yet currently viable given what is written here, so far,

The other issue is that, in classical LG, the disjunct costs $C_{uvw\cdots}$ are hand–crafted by the academic scholar defining the lexis. The learn project attempts to replace these costs by those learned from a training corpus, starting from a pair–wise frequentist counting scheme. The $C_{uvw\cdots}$ are interpreted as mutual entropies, not energies. There is a reasonable argument for why it should be entropy. For the present case, we note that energies are also additive, so we can keep our minds open about what is being additive, here: energy, information, something else abelian, maybe.

The symbolic learning project is to determine the lexis automatically. To learn the correct numerical values of $C_{uvw\cdots}$. And I want to do this by writing down some kind of Hamiltonian–like summation, and attempting to minimize that. But to do that, I need a dramatic change of notation. Lets try it. See what happens. I have no idea if this is a good idea or a bad idea. Or an old idea or a new idea.

Corpus Model

So, first, define a corpus: a collection of sentences. For the moment, let this be "bag of sentences", in jumbled order. A "sentence" double be a dozen words. But it could also be an ordered sequence of 100K words, i.e. a book, article, newspaper story, twitter/bluesky post, whatever. The length is immaterial; instead, we just take a collection of them, assumed to be independent. For now. Of course, ten books written on the same topic are not really independent of one–another, but lets not get ahead of ourselves.

The corpus is $\mathscr{C} = \{S_a\}$ which is a set of sentences S_a labelled with some index a. There is a total of $|\mathscr{C}|$ sentences, where $|\cdot|$ denotes the size (cardinal) of a set. The Hamiltonian is then

$$\mathscr{H} = \sum_{S_a \in \mathscr{C}} H(S_a)$$

where the sum ranges over all sentences in the corpus, and $H(S_a)$ is the total cost of the parse (linkage) of that sentence.

And so here we meet our first conceptual problem: the linkage is not unique: any given sentence may have many linkages, and, in fact, we have to consider, in principle, all possible linkages.

So here we flop over the a graph/network representation. First, lets drop the subscript a on S_a and just say $S \in \mathscr{C}$ is some sentence in the training corpus, and the Hamiltonian will be a sum over these. That doesn't change.

Next, we say that each sentence $S = [w_1, w_2, \cdots, w_n]$ is a temporal sequence of specific words w_i aka "tokens" in conventional machine learning. Here, the index i has a specific meaning: it indicates word–order. Sequential time.

For each sentence S, consider the set $\mathscr{G}(S) = \{G(S)\}$ of all possible connected graphs $G(S)$ for that sentence. A graph here is a set of edges and vertexes: $G = \{V, E\}$ (dropping the indicator S because we fix the sentence for now.) For a given sentence S, the set of vertexes is fixed: $V = \{w_i | w_i \in S\}$ is the set of the words in the sentence S. As a set, the order no longer really matters. But it is a multi–set, so its OK to have a word show repeatedly. An edge is just a conventional graph edge: it is an ordered pair of vertexes. Notation: $E_{ij} = (w_i, w_j)$ is the edge connecting words at locations i, j. It is an ordered pair: edges have direction. For a fixed graph G the set of edges can

be thought of that the adjacency matrix. This is all very super–basic stuff, but I am trying to be as clear as possible here. Sorry. The adjacency matrix is usually take to be a matrix with just ones and zeros in it, with one's taken to mean there is an edge, and zero meaning there is not. This can be generalized, but we're not doing that yet.

A connected graph is one where every vertex is reachable by a path from any other vertex. We want to consider only connected graphs; the ability to disconnect will be handled via a different mechanism.

For a fixed sentence S, write the Hamiltonian

$$H(S) = \sum_{G \in \mathcal{G}} h(G)$$

The sum ranges over all possible connected graphs over the words in the sentence.

Here, we arrive at the next stumbling block. Is it correct to call $h(G)$ the energy, or is it something else? The above has the classic form of a superposition. If we interpret a single graph to be a "state", then the above is a superposition of states, and $H(S)$ should be interpreted as the partition function, and not the energy. For partition functions, the appropriate notation would be a Boltzmann sum:

$$Z_\beta = \sum_{G \in \mathcal{G}} e^{-\beta h(G)}$$

So, which is it? I'm confused, here.

Lets go outside and touch grass. In classical physics, in the classical Ising model, we say that the spins at each lattice site have a distinct, explicit value. Any given physical instance consists of only one configuration of those spins. The thermodynamic ensemble is the collection of all possible spin arrangements. The partition function is a sum over the entire ensemble, giving a Boltzmann weight to each particular spin arrangement.

What, exactly am I doing here, if I am confronted by a sentence, a sequence of words, a sequence of tokens? Should I say that there is only one acceptable parse of that sentence? This is the classic Chomskian answer. Do I need to acknowledge the possibility of multiple ambiguous parses? Of course: "I saw the man with the telescope." said the linguistics textbooks, back when I was in college. Is my reality split, a superposition of two different possibilities? Well, yes it is. The quantum people say so. The Bayesian people say so, and so do the neuroscientists. The linguists merely state that both parses are possible, until I learn some additional information: how has the telescope: me, or the man? Once I learn this additional bit of information, the wave function collapses, the sentence is no longer ambiguous, I know which parse to use, and how to determine the meaning of that sentence.

Several problems here. First, the mistake of treating sentences as assertions of truth. Peotry is ambiguous, and one of the points of poetry is that it never collapses. You get to reinterpret poetry in various ways: its evocative, not assertive. Next, parsing is a syntactic activity; we've not touched base with semantics.

Training vs. Parsing

So, where does that leave us? For now, lets ditch semantics: th In particular, we wante original problem was to discover a collection of disjuncts that adequately describe possible syntactic parses of sentences. The goal was to do this with some sort of expression of aggregate satisfiability: some kind of function assigning weights to disjuncts, and maximizing it.

That is, the corpus is fixed, immutable. The words in the sentences are fixed, immutable. There's a confusion between two different tasks. In one task, the costs of the disjuncts are known, fixed, and the goal of parsing is to find the collection of disjuncts that best fit the sentence. The sum–total of these costs is the aggregate satisfiability. In this mode, the disjuncts are analogous to the Ising spins, and we want to find an arrangement of these spins/disjuncts that minimize the parsing error: minimize the parse cost.

Just like a ferromagnetic lattice, we acknowledge that there might be multiple ambiguous parses, and that's OK, there is nothing wrong with that ambiguity. For this task, it is entirely appropriate to use the partition function, because the partition function captures the likelihood of any given parse as being correct. That is, $h(G)$ is the "energy" of a parse G and the probability of that parse being reasonable is

$$P_\beta(G) = \frac{e^{-\beta h(G)}}{Z_\beta}$$

This is the "configuration probability" of the parse G, and there sentences like "I saw the man with the telescope" just says there are two parses G and G' with $h(G) = h(G')$.

So that is one task: find the statistical distribution of parses that are the most likely.

The other task is to learn what the disjuncts are, to learn the cost of each disjunct. The costs are NOT fixed, we are instead exploring the space of possible disjunct assignments. For this we need to take the other form:

$$H(S) = \sum_{G \in \mathscr{G}} h(G)$$

This is the appropriate form, because we want to consider all possible disjunct cost assignments, and minimize those. Unlike the earlier case where multiple distinct (ambiguous) parses can be tolerated and are even desirable, here, not so much. Here, we do want to discover single unique, unambiguous collection of disjuncts. It is hard to understand why more than one would be desired. Perhaps some hand–waving is possible, but hard to see what that is.

So lets try that. Lets say that there is "one true set" of disjuncts that guide the parsing and understanding of the external world. If they are correct and accurate, then all is well and we continue onwards. Sometimes, model updates would be needed. Most of these should be small, but it is easy to imagine avalanches in "understanding": if some lexis no longer parses reality, that lexis may need an update. That update may be dramatic.

So why would one want to keep around multiple different lexis? Well, to try them out, maybe see how they differ. In this model, each lexis is an "individual", and that

individual has it's own view of what the best kinds of parses might be. Most individuals agree, but clearly, if the training corpus for one individual is a bit different from that of another individual, there will be differences.

So I'm having fun with this: any particular collection of disjuncts is an individual. This can of course be taken much more broadly: different LLM weight–sets are indeed "individuals", but this diary is not about LLM's, so we won't go there.

Disjuncts

It's time to firm up what the disjunct discovery process will be. For that, an expression for $h(G)$ is needed. So, G is a graph, having edges between some words in a sentence, but not others. Talking about words keeps things concrete, but most of the below should generalize just fine for any kind of network graph.

For each word w_i in the sentence, define the disjunct D_i to be the set of edges attached to that word: so

$$D_i = \left\{E_{ij} : E_{ij} \neq 0\right\} = \left\{E_{ij} : E_{ij} \in G\right\}$$

which are two different notations for the same thing: the edge E_{ij} is in the graph G. The colon notation : means "such that": the edges such that they belong to the graph. Properly, the disjunct is an ordered list. The sequence of the edges matters (because word–order matters).

Thus, instead of using set–builder notation $\{\cdot\}$ we could use other kids of notation. Don't freak, but here are other notations commonly found in the literature:

$$D_i = E_{ij_1} \otimes E_{ij_2} \otimes \cdots \otimes E_{ij_k}$$

where $\otimes$ is the tensor product. The reason that the tensor product notation is correct can be found in a variety of other places, so I won't try to explain the motivation for it. Link Grammar uses the & instead of $\otimes$ to write down disjuncts. The above is a tensor of arity k – it has k different connectors.

Another notation is the bra–ket notation:

$$D_i = \left|E_{ij_1}\right\rangle \otimes \left|E_{ij_2}\right\rangle \otimes \cdots \left|E_{ij_m}\right\rangle \otimes \left\langle E_{ij_{m+1}}\right| \otimes \cdots \otimes \left\langle E_{ij_k}\right|$$

where kets are used whenever $j_n < i$ and bras are used when $j_n > i$. That is, j_m is the index such that $j_m < i < j_{m+1}$. Note the inequalities are strict: a word ca never link to itself. The point of bra–ket notation is to say that words with $j_n < i$ link to the left, otherwise they link to the right. This is fine for English but just makes things hard for other languages and for general graphs; the bra–ket notation is problematic for a number of reasons. One is that natural language is not a symmetric monoidal category: the is no dagger operator. It is not true that $|\cdot\rangle^\dagger = \langle\cdot|$. Another problem is that bra–ket notation is inherently "heterosexual", in that inner products can only be formed with one bra and one ket, whereas in "real life", connectors may come with other kinds of sexes and mating rules. For example, monosexual connectors, where there is only one sex, are appropriate for graphs with undirected edges, for which left–right

ordering does not matter. In other languages, such as Turkish, Finnish and Lithuanian, word order mostly does not matter, but head–dependent order matters very strongly. For those languages, there are four sexes: a product of the dominant head–dependent dependency, and a much weaker left–right mating distinction for a small minority of words. Enough linguistics for now; this gives a flavor of some of the issues that come up in practice.

Although the bra–ket notation is problematic, the tensor notation is not: the non–symmetric monoidal category really is the appropriate category for language. It really is a fragment of linear logic. See Wikipedia articles on pre–group grammars. See all my (Vepstas) paper on "Categorial Grammars are Equivalent to Link Grammar"; I think search engines can find this.

To conclude: a single disjunct is the same thing as a single–element tensor (a basis tensor, i.e. a product of basis vectors) and this is the same thing as a jigsaw piece. All three of these are the same thing.

Costs

To each disjunct, we wish to assign a cost: a floating–point, real–valued number α_i for each D_i. The total cost is then

$$h(G) = \sum_{1 \le i \le n} \alpha_i$$

Recall that G is fixed, and that therefore the D_i are fixed: they are uniquely determined by the graph. The task is to find an assignment of α_i that minimize the total cost over the entire training corpus.

$$\mathscr{H} = \sum_{S \in \mathscr{C}} H(S) = \sum_{S \in \mathscr{C}} \sum_{G \in \mathscr{G}} h(G)$$

Of course, this can be met by setting $\alpha_i = -\infty$ which is pointless, and so a bound is needed: let the costs $0 \le \alpha_i$ be always positive (or above some lower bound. It doesn't matter just right now.)

The above is a general expression. Its correct, but awkward, in a way. For natural language, we want D_i not to depend on the word–position i but on the word w_i at that position. That is, write not D_i but $D(w_i)$ and since the disjunct should not depend on the word position, but only the word, then $D(w)$ should be written for the word w.

Likewise, for the edge E_{ij} we don't want a dependency on the word–position j but instead on the word w_j at location j. Thus, write $E_{ij} = E(w_i, w_j)$ or, better yet, since w_i is the same for all edges in the disjunct, simply write $D(w) = u \otimes v \otimes \cdots$ where $u, v, \cdots$ are other words. That is, the disjunct no longer depends on the sentence that it is on, nor does it depend on which location it has in the sentence: it only depends on the word, and it's connectors.

So as not to get lost in abstraction, a practical example is in order. Consider the canonical parse of "the dog chased the cat". This has "chased" as a transitive verb, "dog" and "cat" as nouns, and "the" as determiner. The disjuncts are:

$D(\text{CHASED}) = \texttt{DOG}- \,\&\, \texttt{CAT}+$
$D(\text{DOG}) = \texttt{THE}- \,\&\, \texttt{CHASED}+$
$D(\text{CAT}) = \texttt{THE}- \,\&\, \texttt{CHASED}-$
$D(\text{THE}) = \texttt{DOG}+$ or $\texttt{CAT}+$

Here, the plus and minus indicate a linkage to the left or to the right. The transitive verb links to a subject on the left and an object on the right. English is an SVO language, so we can tell apart the subject noun and the object noun from the directional indicators + and -. So "cat" is the object, because it has "chased-" on it, and not "chased+". The ampersand & was used in place of the tensor $\otimes$ otimes symbol. This is in keeping with Link Grammar.

Again, this is super–basic stuff; I'm repeating it here so that we can keep our feet on the ground.

That's the good news. The bad news is that I am laboriously re–inventing the skip–gram. To some large degree, if we walk much farther down this path, we risk re–inventing BERT. And that is not really the goal. Soo....

Training

So, armed with a definition of a disjunct that closely resembles a BERT–style skip–gram, can we invent a training technique that avoids the standard transformer foot–path? That is, the goal here is to ask "is there another way, and what might it be?"

The End

That's all for now.