Language Learning Diary

Linas Vepstas

April 14, 2017

Abstract

Collection of thoughts and notes, pursuant to language learning. Organized in mostly chronological order.

31 December 2013

Easier said than done. Should be 'obvious' but its not. Goal: construct a set of linkage rules to model only the observed language, using Link Grammar. That is, using the graphical structure implied in Link Grammar. This, of course, implicitly assumes that Link Grammar provides an adequate theory for describing and constraining the syntactical parsing of a natural language.

Lexical Attraction, Mutual Information, Interaction Information

The goal of this section is to clarify some of the formulas used by Deniz Yuret in his PhD thesis "*Discovery of Linguistic Relations Using Lexical Attraction*", MIT 1998 (http://www2.denizyuret.com/pub/yuretphd.pdf). These formulas are vitally important, because they provide a strong tool when working with text; this has been shown by Yuret in his thesis, as well as by many others, as well as by my own practical exprience with using them.

Possibly the most useful formula is the one in the middle off page 40. By the time that we get to it, the terms "mutual information" and "lexical attraction" are being used interchangably. This formula states the $MI(x,y)$ for two words x and y; yet it is manifestly not symmetric in x and y, since x is the word on the left, and y is the word on the right. By constrast, textbook (wikipedia) definitions of MI are symmetric in thier variables. Below I try to dis-entangle the resulting confusion a bit, and give a more correct derivation of the formula. The key is to observe that the formula contains an implicit pair-wise relationship between two words, and that there are actually three variables: two words, and thier relationship. If this implicit relationship is made explicit, then the confusion evaporates. It also opens the door to talking about the MI (or the interaction information InI) of more complex relationships, not just pair-wise ones.

Being able to correctly write down the MI and the InI for complex relationships is important for NLP: relationtionships can be labelled by types (subject, object) and by word classes (noun, verb), and have various dependency constraints between them.

Thus, we need to be able to talk both about a labelled directed graph, and the entropy or mutual information contained in it's various sub-graphs.

In defense of Yuret, he does say, on page 22, that "lexical attraction is the likelihood of a syntactic relation." However, the relation starts becoming implicit by eqn 12 on page 29. An unexplained leap is then made from eqn 12 to the formula on page 40. The below gets fairly pedantic; this seems unavoidable to avoid confusion.

Definitions

Let $P(R(w_l,w_r))$ represent the probability (frequency) of observing two words, w_l and w_r in some relationship or pattern R. Typically, R can be a (link-grammar) linkage of type t connecting word w_l on the left to word w_r on the right; implicitly, both w_l and w_r occur in the same sentence. The goal of this discussion is to enable relations R that are more general than this; for now, though, R is a word-pair occuring in a single sentence.

The simplest dependency grammar language model has only one type t, the ANY type. This is the type that Yuret uses: it makes no distinction at all between subject, object relations (that is, all depdencies are unlabelled), and it does not make a head-dependent distinction (all dependencies are bi-directional). Thus, in what follows, we do the same: initially, the relation $R(w_l,w_r)$ is simply the statement that the words w_l and w_r are connected by an unlabelled, un-directed edge. For this simplest case, what $R(w_l,w_r)$ does do is to capture that w_l is to the left of w_r.

In what follows, the relation $R=R(w_l,w_r)$ refers to a generic two-word relation, and not necessarily this simplest one. To regain Yuret's formula, use the simplest relation, described above.

We are then interested in the conditional probability $P(R(w_l,w_r)|w_l,w_r)$ of observing the two words w_l and w_r in a relation $R=R(w_l,w_r)$, given that the two individual words were observed in a sentance (in no particular order). Let $P(w)$ be the probability of observing word w in a sentance (with multiplicity). We know from experiece this is a Zipfian distribution; however, this fact will not be used in what follows. From the definition of conditional probabilities, we get

$$P(R,w_l,w_r)=P(R|w_l,w_r)P(w_l,w_r)$$

or that

$$P(R|w_l,w_r)=\frac{P(R,w_l,w_r)}{P(w_l,w_r)}$$

Here, the relation R encompasses several facts: that one word is to the left of the other, and that they are connected by a certain link-type, as well as possibly capturing other 'ambient' information, perhaps such as other nearby words. The expression $P(R,w_l,w_r)$ is the probability of observing both the words w_l and w_r in a sentance, as well as observing that they are related as $R(w_l,w_r)$ in the sentence. Clearly, R cannot occur if w_l and w_r don't, but the converse is not true: the words can appear in a sentence, but fail to be related by R. For this reason, it is "safe" to use the notational short-hand $P(R)=P(R(w_l,w_r),w_l,w_r)$; nothing is thrown out, because R already depends on w_l and w_r.

The probability $P(w_l,w_r)$ is interesting, because it's definition is not enirely unambigous: we can define it as the probability of observing both words in the same sentence, or we can define it as the probability of observing both words, as if they were independent of one-another. Clearly, the occurances of pairs of words are correlated; suggesting the firt defintion is correct. The second one seems to be a dumb, brute-force assumption of independence, i.e. that $P(w_l, w_r) = P(w_l)P(w_r)$.

What should be done? This is a bit confusing, and is worth thinking about. To observe two words in the same sentence, we could say that they obey the relationship $S(w_l, w_r)$ that captures thier correlation to one-another. This leads one to ponder the probability $P(S(w_l, w_r), w_l, w_r)$. But this is not what we want, in the end; we are interested in R and not S. The relationship R includes S implicitly: two words can only be related by R only if they are already related by S. Thus, the only appropriate definition for $P(w_l, w_r)$ that we can use is that the two words really are un-related, un-correlated, independent. That is, we must define $P(w_l, w_r) = P(w_l)P(w_r)$. This is important: we don't "assume" independence, we define it. The goal is to capture the dependence as a part of the relation R. Thus, we conclude with the conditional probability:

$$P(R|w_l, w_r) = \frac{P(R, w_l, w_r)}{P(w_l)P(w_r)} \tag{1}$$

This is the probability of observing the relationship R given that the individual parts of the relationship have been observed.

Frequentism

In order for the above to be manipulable in practice, we need to provide a definition for the probabilities, and, for this, the definition can only be frequentist. That is, the defintion is to be obtained from empircal data; from counting frequencies as they occur in nature. The frequency $P(w)$ of observing a word w is obvious:

$$P(w) = \frac{N(w)}{N(*)}$$

where $N(w)$ is the count of observing word w and $N(*)$ is the total number of words observed. That is, by definition, it is the wild-card summation

$$N(*) = \sum_w N(w)$$

The conditional probability for two words in a relation is similarly defined:

$$P(R|w_l, w_r) = \frac{N(R, w_l, w_r)}{N(w_l)N(w_r)}$$

This follows from the form of eqn 1. It is worth unpacking the unconditional probability. It is given by

$$\begin{aligned} P(R, w_l, w_r) =& P(R|w_l, w_r)P(w_l)P(w_r) \\ =& \frac{N(R, w_l, w_r)}{N(w_l)N(w_r)} \frac{N(w_l)}{N(*)} \frac{N(w_r)}{N(*)} \\ =& \frac{N(R, w_l, w_r)}{N(*)N(*)} \end{aligned} \tag{2}$$

That is, the probability is just the number of times the relationship was observed, divided by the total number of times two words are observed.

Another interesting quantity is the probability of seeing a pair-wise relationship, in general, independent of the actual words in the pair. This may seem somewhat pointless at first, but becomes interesting when a variety of different relations considered. This probability is given by

$$P(R(*,*)) = \sum_{w_l,w_r} \frac{N(R,w_l,w_r)}{N(*)N(*)} = \frac{N(R(*,*))}{N(*)N(*)}$$

XXX Report on values of the eabove for English.

Yuret Notation

We need to harmonize here a little bit with the Deniz Yuret concepts and notation. He defines a probability $\mathscr{P}(w_l,w_r)$ of seeing the ordered pair; that is, the relation R is implicit. To make it explicit, we should write: $\mathscr{P}(w_l,w_r) = P(R(w_l,w_r)|R(*,*))$ to indicate the relation explicitly, and to note that the order of the positions in the relation matter. The right-hand side is the conditional probability of seeing a given word pair, conditioned on seeing any word pair. To avoid confusion, the cursive $\mathscr{P}$ is used instead of the roman P. Yuret also uses the notation $\mathscr{P}(w_l,*)$ and $\mathscr{P}(*,w_r)$ for wild-card summations, defined as

$$\mathscr{P}(w_l,*) = \sum_{w_r} \mathscr{P}(w_l,w_r) \qquad \text{and} \qquad \mathscr{P}(*,w_r) = \sum_{w_l} \mathscr{P}(w_l,w_r)$$

It is tempting to conflate $\mathscr{P}(w_l,*)$ with $P(w_l)$ but that would be wrong; not every possible word can occur on the w_r position. This suggests a different, but tempting, error, that $\mathscr{P}(w_l,*) \leq P(w_l)$. This is also not the case! This comes from the fact that the frequentist definitions of the two are incompatible: the denominators are not the same. Viz,

$$\mathscr{P}(w_l,w_r) = \frac{N(w_l,w_r)}{N(*,*)}$$

while

$$P(w) = \frac{N(w)}{N(*)}$$

so that the former is normalized by the number of times $N(*,*)$ the pairs have been observed, while the later is normalized by the number of times that singletons have been observed. The incompatible normalizations can be understood as conditional probabilities:

$$\mathscr{P}(w_l,w_r) = P(R(w_l,w_r)|R(*,*)) = \frac{P(R(w_l,w_r))}{P(R(*,*))}$$

That is, the Yuret probability of seeing a given word pair is actually the conditional probability of seeing that pair, conditioned on seeing *any* word pair. The numerator of this expression is given by eqn 2, that is, $P(R(w_l,w_r)) = P(R(w_l,w_r),w_l,w_r)$, while the denominator is the wild-card sum:

$$P(R(*,*)) = \frac{N(R(*,*))}{N(*)N(*)}$$

which can be viewed as a conditional probability:

$$\mathscr{P}(w_l,*) = \frac{P(R(w_l,*))}{P(R(*,*))} = \frac{P(R(w_l,*)|w_l)}{P(R(*,*))}P(w_l) \le P(w_l)$$

In practice, then, for word-pairs, one has that $P(R|w_l)/P(R)$ is almost equal to 1, but not quite. That is, $P(R|w_l) \approx P(R)$. This can be confirmed by actual measurements for English, and should hold true for most languages. XXX TODO Report on this (double-check) XXX.

Thus, Yuret's notation can be expressed either in terms of conditional probabilites, as

$$\frac{\mathscr{P}(w_l,w_r)}{\mathscr{P}(w_l,*)\mathscr{P}(*,w_r)} = \frac{P(R(w_l,w_r)|w_l,w_r)\;P(R(*,*))}{P(R(w_l,*)|w_l)\;P(R(*,w_r)|w_r)}$$

or in terms of the unconditional probabilities, as

$$\frac{\mathscr{P}(w_l,w_r)}{\mathscr{P}(w_l,*)\mathscr{P}(*,w_r)} = \frac{P(R(w_l,w_r),w_l,w_r)\;P(R(*,*))}{P(R(w_l,*),w_l)\;P(R(*,w_r),w_r)} \tag{3}$$

The left hand side above is used by Yuret to define the “lexical attraction”, as

$$\mathrm{MI}(w_l,w_r) = \log_2 \frac{\mathscr{P}(w_l,w_r)}{\mathscr{P}(w_l,*)\mathscr{P}(*,w_r)} \tag{4}$$

so that large positive MI is associated with words that rarely seen one without the other (e.g. '*Northern Ireland*' from his examples.) Note the absence of a minus sign in the above! See below for an explanation. Large-MI word pairs occur when $\mathscr{P}(w_l,w_r)$ is roughly comparable to $\mathscr{P}(w,*) \approx \mathscr{P}(*,w) \approx P(w)$.

It is worth reviewing Yuret's example, at this point. He looks at the word pair 'Northern Ireland' and states (based on a particular corpus that was analyzed) that $-\log_2 P(\text{'Northern'}) = 12.60$ and that $-\log_2 P(\text{'Ireland'}) = 14.65$ and finally that $-\log_2 \mathscr{P}(\text{'Northern'},\text{'Ireland'}) = 16.13$. What these numbers mean is that although either word alone occurs at a rate of roughly once in ten-thousand words, the word-pair together occurs at the rate of one in thirty-thousand or so: the word pair occurs almost as often as either word alone. Thus, the resulting MI is very large: $\mathrm{MI} = -16.13 + 12.60 + 14.65 = 11.12$. The choice of sign in eqn 4 is such that word that co-occur have a large positive value. In practice, the distribution of the MI for word-pairs runs from about -15 to about +35, and, when ranked accoring to MI, the probabilites form a rounded mountain-peak, two-sided, each side being linear (Zipfian) with the peak at about MI=4 or 6. (See my other notes for a graph.)

Observationally, we have that $P(R(w_l,*)|w_l) \approx P(R(*,w_r)|w_r) \approx 1$, and thus must conclude that $P(R(w_l,w_r)|w_l,w_r)$ is 'large'; much larger than (unconditional) word-pair frequencies. XXX double check this; the definitions have changed.XXX.

The RHS of eqn (3) is “nicer” than Yuret’s definition, in that it is manifestly symmetric in w_l and w_r. This symmetry is essential for making contact with the ’standard’ definitions of MI, which are always provided in a symmetric fashion. It also helps make clear the role of the left-right relationship; eqn 4 is in fact the three-point interaction information

$$I(R;w_l;w_r) = MI(w_l,w_r)$$

that is, the interaction information of three random variables: w_l, w_r and $R = R(w_l,w_r)$. In particular, Yuret’s lexical attraction is specifically *not* the mutual information of two random variables; it includes a specific, defined relationship between them: that one word is explicitly to the left of another, as well as the “ambient” relation that they occur within the same sentence.

Update 14 Jan 2014

OK, The LHS of equation (3) then demonstrates how to obtain conditional entropies in general. Thus, given an n-point relation $R(x_1,x_{2,}\cdots,x_n)$ one computes first the unconditional probability $P(R(x_1,x_{2,}\cdots,x_n))$. The conditional probability is then obtained as usual:

$$P(R(x_1,x_{2,}\cdots,x_n)\,|\,x_1,x_{2,}\cdots,x_n) = \frac{P(R(x_1,x_{2,}\cdots,x_n))}{P(x_1)P(x_2)\cdots P(x_n)}$$

The entropy is then built recursively by normalizing by the probability of wild-card relations:

$$MI(R(x_1,x_{2,}\cdots,x_n)\,|\,x_1,x_{2,}\cdots,x_n)) = \log_2\frac{P(R(x_1,x_{2,}\cdots,x_n)\,|\,x_1,x_{2,}\cdots,x_n))}{P(R(*,x_{2,}\cdots,x_n)\,|\,x_{2,}\cdots,x_n))\,P(R(x_1,*,\cdots,x_n)\,|\,x_1,x_{3,}\cdots,x_n))\cdots} \tag{5}$$

(Right? We’re not dividing by multiple *’s here, yes?)

1 January 2014

OK, after that side distraction, which helped clear up notation, back to the main show ...

The main show is this: We want to model language, and specifically, find a ’minimal’ set of relations R that are accurately generative. The meaning of ’minimal’ seems obvious, intuitively, but a lot harder to pin down mathematically. We need to pin it down to get an algorithm that works in a trust-worthy, understandable fashion.

So: what is the total space of relations, and how do we find it? The simplest model is then a Zipfian distribution of words, but placed in random order. This model has a total entropy of

$$H = -\sum_w P(w)\log_2 P(w)$$

For a recent swipe at parsing a few hundred articles from the French wikipedia, I get H=7.2. This is on 17K words, observed a total of 35M times (actually, observerd each sentence 100 times, so really just 350K ’true’ observations of words).

How does one count the entropy of the rule-set? Elucidating this is the goal-set.

But first, step back: describe the rules.

OK ... so, once again ... sentence structure is to be described via link-grammar, using disjoined conjunctions of connectors. This is theoretically sound, as it seems to be isomorphic to categorical grammars (via type-theory of the connectors; need a formal proof of this someday, but for now it seems 'obvious'). Also link-grammar is fully compatible with dependency grammar. So lets move forward. But this is an old debate, off to the side, immaterial for now.

How to count relations

Consider a sentence with n words in it, numbered $w_1, w_2, \cdots, w_n$ left to right. We want to constrain grammar by discovering a set of relations $R(w_1, w_2, \cdots, w_n)$ such that $P(R(w_1, w_2, \cdots, w_n)) > 0$ when the sentence is gramatically valid (*i.e.* such an R exists), and P is zero when no such R exists (*i.e.* the sentence is not gramatically valid.) The first and most obvious simplification rule is to observe that R can be replaced by $R(W_1, W_2, \cdots, W_n)$ where each W_k is a set of words. That is, instead of listing each sentences individually, we list certain classes of sentences. In other words, the relations $R(w_1, w_2, \cdots, w_n)$ are in one-to-one correspondance with a list of grammatical sentences $(w_1, w_2, \cdots, w_n)$, so simply listing all possible sentences is a very verbose way of specifying a grammar. It is linguistically 'obvious' that sentences fall into classes, and so the two relations $R('this', 'is', 'a', 'dog')$ and $R('this', 'is', 'a', 'cat')$ can be replaced by $R('this', 'is', 'a', W_n)$ where $W_n = \{'dog', 'cat'\}$. In fact, W_n can be a rather large set of nouns.

So ... the question is: what is the reduction of complexity, by performing this classification? What is the correct way of counting? I assume that 'complexity' is a synonym for 'entropy', so we are looking to do two things: enumerate the states of the system, and proivde a measure for complexity. So, lets consider a language with N nouns, so that the cardinality of W_n is $|W_n| = N$ and the only valid sentences are $('this', 'is', 'a', w)$ with $w \in W_n$. Before simplification, we had N relations R, one per sentence. We also had $N+3$ sets, each set containing a single word; the N nouns, and the three words $'this', 'is', 'a'$. After simplification, we have one relation R, and four sets; three of the sets have cardinality 1, the fourth set has cardinality N.

Revision: July 2014

There seem to be several ways of counting. Some of these seem to give wrong answers. Some just seem wrong. This is all very confusing, so I've altered the entries to explicitly show the different ways of counting.

Method 1 (naive counting): One counting rule is to count set-membership relations on equal footing with structural relations. Thus, before simplification, we had $N+3$ sets, each a singleton, and thus $N+3$ set membership relations. After simplification, we have four sets, but still have $N+3$ set membership relations. Thus, this particular simplification step does not reduce the number of membership relations at all. This seems disconcerting... Let's provisionally go with this and see what happens. Thus, before simplification, we had $2N+3$ relations grand-total, and afterwords, we have $N+4$ relations grand-total.

What is the correct 'thermodynamic' picture of what's going on? In this toy problem, we have a grand-total state space of size $(N+3)^4$ since any of the $N+3$ words can appear in any of the four slots in a four-word sentence (micro-canonnical ensemble). The entropy, at 'infinite temperature' where all possible four-word sequences occur with equal probability is then $4\log_2(N+3)$. The entropy of the set of grammatical sentences is $\log_2 N$ since there are only N possible grammatical sentences. In this toy grammar, there are also invalid setences of length 1,2,3,5,6,7,... and so the total size of the space of word-sequences is clearly infinite.

OK, so the space of word-sequences is very concrete, and easy to describe and measure, at least for toy grammars. What about the space of relations? Well, the claim is that the entropies of the before-and-after models are $\log_2(2N+3)$ and $\log_2(N+4)$, respectively. Neither of these matches the entropy of the set of allowed sentences (which is $\log_2 N$), so this seems paradoxical, and begs the questions 'did we count correctly?' and 'did we actually simplify anything by making the above change of description?' Hmm. The correct answer seems to be 'no' and 'no'.

Method 2 (subtract one): To 'fix' the oddball results above, an alternative counting methodology is to subtract 1 from the cardinality of every set. This would then give both $\log_2 N$ as the entropy for both the before and after relation-sets. Thus, before, we had N relations and $N+3$ sets, each of weight zero, for a total weighted-relation count of N. After, we have one relation and four sets; three of the sets have weight zero, one set has a weight of $N-1$ so the total weighted relations is again N. This seems to resolve the paradox. But why subtract one? That's a bizarre rule, almost unheard-of in information theory.

Method 3 (naive log addition): Total complexity is given by:

$$K = \log_2 |Rel| + \sum_{W \in Wrds} \log_2 |W|$$

where Rel is the set of relations, and $Wrds$ is the set of word-lists, and $|W|$ is the cardinality of each word-list. We then get, before simplification, $|Rel| = N$ and $|W| = 1$ for each of the word-sets. The total complexity is thus $K = \log_2 N$ as expected (i.e. equal to the log of the total number of possible sentences). After simplification, there is $|Rel| = 1$ and 3 sets with $|W| = 1$ and one set with $|W| = N$, thus yeilding a total of $K = \log_2 N$ again. This seems to give a plausible answer, and provides a plausible argument.

Method 4 (relational complexity): Treating each relation as being equally complex seems odd. It would seem to make more sense to have each relation contribute according to its complexity, so that the contribution of the relations to the total complexity is:

$$\sum_{R \in Rel} C_R$$

with C_R the complexity of each relation, itself the log of some measure. But how do we measure complexity? Is it Kolmogorov complexity? There's no obvious *a priori* defintion for this. The defintion of this complexity would seem to depend on the particular algorithm machinery of the grammar; that is, on the 'programming language' used to represent the relation. This is the traditional ambiguity attached to the Kolmogorov complexity.

Method 5 (corpus distribution): Instead of measuring the complexity of a grammatical expression (in an as-yet unkown grammar), instead, use the corpus frequency ass a proxy. For the above example, if the N sentences are equi-distributed (i.e. occur equally likely in the corpus), then, before simplification, each of the relations has a complexity

$$C_R = -\frac{1}{N} \log_2 \frac{1}{N}$$

so that, before simplification,

$$K = \sum_{R \in Rel} C_R = \log_2 N$$

which again seems to be the desired answer. After simplification, there is one relation that applies to the entire corpus, so that $C_R = 0$ after simplification.

Method 6 (corpus word-counts): If we are taking word-relation frequencies from the corpus, then we should be taking word-set frequencies from the corpus as well. That is, the word-set contribution $\log_2 |W|$ is assuming an equi-distribution. This should be replaced by the corpus contribution

$$-\sum_{w \in W} p(w) \log_2 p(w)$$

Summary. Provisionally, the last two methods seem to be the best way to move forward. To summarize, the complexity is given by

$$K = -\sum_{R \in Rel} P_R \log_2 P_R - \sum_{W \in Wrds} \sum_{w \in W} P_w \log_2 P_w \tag{6}$$

where $P_R = P(R) = P(R(W_1, W_2, \cdots, W_n))$ is the probability of observing the relation R in a sample corpus, and $P_w = P(w|W)$ is the probability of observing word w in the corpus, conditioned on its appearence in the corpus having to do with it belonging to the word-class W.

Counting Link-Grammar Relations

Per link-grammar, each relation is decomposable into pair-wise relations; this is the so-called 'parse' of a sentence. If the relation is a single word-per-slot sentence relation, then the 'parse' is literal. We write

$$R(w_1, w_2, \cdots, w_n) = \prod_{j,k,m} R_\alpha(w_j, w_k, t_m) \; Q(R_\alpha, R_\beta, \cdots, R_\omega) \tag{7}$$

where $R_\alpha(w_j, w_k, t_m)$ is a single connected pair of words, connected by the connector t_m. The product symbol $\prod$ implies that all such binary relations must hold. The awkward $Q(R_\alpha, R_\beta, \cdots, R_\omega)$ at the end is the additional no-links-cross constraint in the current link-grammar parser. Its a non-local constraint involving all of the binary relations. It also subsumes any 'post-processing' rules, although, for the language learnign exercise, there won't be any post-processing rules. At any rate, Q is a place where higher ordrer constraints can be applied. In particular, the most genneral form for Q should be $Q(R_\alpha, R_\beta, \cdots, R_\omega, w_1, w_2, \cdots, w_n)$ since, in principle, it could depend on the word-choice, although the no-links-cross constraint does not.

Yuret proposes a way of discovering the pair-wise relations[Yur98]. He makes the implicit, unvoiced assumption that there is a single, unique connector type t_m for every ordered pair of words w_j, w_k; that is, that $t_m = t_m(w_j, w_k)$. Viz, specifically, that such connectors are in 1-1 correspondance with word-pairs. (I don't think he's aware of this assumption; I don't think anyone has ever before realized that he's making such an assumption; certainly, I haven't). Yuret then makes two claims: first, that the only possible grammatically correct parses are those of the above form (eqn (7)) for which the relations $R_\alpha(w_j, w_k, t_m(w_j, w_k))$ have been previously observed; secondly, that there is a natural ranking of such allowed parses by summing the total mutual information associated with each word-pair.

These two concepts give rise to the idea of minimum-spanning-tree parsers. Such parsers work in a two-step process: a training phase, and a parse phase. In the training phase, one gathers a lot of statistics about mutual information. The important point here is that this is unsupervised training. To parse, one first creates a graph clique, with every word connected to every other. One uses the gathered MI to define graph edge lengths. Finally, the corrrect parse is then the maximum spanning tree of the graph (maximizing the MI, summed over the tree edges in the graph).

Here, we use the same idea, but then take the next step. The spanning tree can be decomposed into a set of link-grammar disjuncts, one disjunct per word. The disjunct is merely a list of the connections that one word makes. It consists of the type, and the direction. The direction is left or right. The type is the $t_m = t_m(w_j, w_k)$ defined above. By parsing a large number of sentences, we can now automatically discover a large number of disjuncts, in an unsupervised manner.

The goal, the next step, is then to reduce the total number of disjuncts, and the total number of types, by clustering and discovering similarities.

3 January 2014

No-crossing Minimum Spanning Trees

It turns out that writing an algorithm for a no-crossing minimum spanning tree is surprisingly painful; enforcing the no-crossing constraint requies treatment of a number of special cases. But perhaps this is not actually required! R. Ferrer i Cancho in "Why do syntactic links not cross?"[iC06] shows that, when attempting to arrange a random set of points on a line, in such a way as to minimize euclidean distances between connected points, one ends up with trees that almost never cross!

Other related references:

- Crossings are rare: Havelka, J. (2007). Beyond projectivity: multilingual evaluation of constraints and measures on non-projective structures. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07): 608-615. Prague, Czech Republic: Association for Computational Linguistics.

- Hubbiness is a better model of sentence complexity than mean dependency distance: Ramon Ferrer-i-Cancho (2013) "Hubiness, length, crossings and their relationships in dependency trees", ArXiv 1304.4086 — also states: maximum number of crossings is bounded above by mean dependency length. Also, mean dependency length is bounded below by variance of degrees of vertexes (i.e. variance in number of connectors a word can have).

- Language tends to be close to the theoretical minimum possible dependency distance, if it was legal to re-arrange words arbitrarily. See Temperley, D. (2008). Dependency length minimization in natural and artificial languages. Journal of Quantitative Linguistics, 15(3):256-282.

- Park, Y. A. and Levy, R. (2009). Minimal-length linearizations for mildly context-sensitive dependency trees. In Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT) conference.

- Sentences with long dependencies are hard to understand: The original claim is from Yngve, 1960, having to do with phrase-structure depth. See – Gibson, E. (2000). The dependency locality theory: A distance-based theory of linguistic complexity. In Marantz, A., Miyashita, Y., and O'Neil, W., editors, Image, Language, Brain. Papers from the first Mind Articulation Project Symposium. MIT Press, Cambridge, MA.

- (Cite this, its good) Mean dependency distance is a good measure of sentence complexity – for 20 languages – Haitao Liu gives overview starting from Yngve. [Liu08]. Haitao Liu "Dependency distance as a metric of language comprehension difficulty", 2008, Journal of Cognitive Science, v9.2 pp 159-191 http://www.lingviko.net/JCS.pdf

- Sentences with long dependencies are rarely spoken: Hawkins, J. A. (1994). A Performance Theory of Order and Constituency. Cambridge University Press, Cambridge, UK. —-Hawkins, J. A. (2004). Efficiency and Complexity in Grammars. Oxford University Press, Oxford, UK. —-Wasow, T. (2002). Postverbal Behavior. CSLI Publications, Stanford, CA. Distributed by University of Chicago Press.

- Dependency-length minimzation is universal: Richard Futrell, Kyle Mahowald, and Edward Gibson, "Large-scale evidence of dependency length minimization in 37 languages" (2015), doi: 10.1073/pnas.1502134112

So, rather than imposing no-crossing as a constraint on the parser, instead, let it find its own way into the grammar. Just implement a plain-old MST parser, punt on crossing.

11 January 2014

Clustering Redux

OK, so what is the very next algorithmic step? Up to here, we've generated a large number of unique disjuncts. Now what?

Back to counting. Lets do the French dictionary. The database fr_pairs contains table atoms_mi_snapshot. So:

- `select count(*) from atoms_mi_snapshot;` returns 415532

15 January 2014

Embodied Learning

OK, so maybe learning syntax before emantics puts the cart before the horse. Can we learn a world-model first, and then gradually annotate and correct it as our linguistic comprehension improves? So, for example, can we start with a world-model obtained via document summarization? How do we annotate this model with newly discovered data?

Related question: how to automatically discover ontologies? Automated, unsupervised concept, entity extraction? Semantic context change over time?

Steps:

1. How do I extract entities out of a text? The extraction doesn't have to be perfect; having candidate entities is enough. How do I put a confidence rating on the entiy, and how do I discard the low-cnfidence ones?

2. Once entities are extracted, I want to start decorating them with attributes (adjectives, modifiers), to build a network.

3. Once a network is built, it needs to be factually reconciled, using logical reasoning and an ontology (is-a and has-a relations). Need to do this so that upon reading "colorless green ideas", we can deduce that ideas are either colorless or green, but not both.

4. How to automatically extract an ontology from free text?

The above seem to be the central steps/core issues for creating a world-model, unsupervised, from text.

Entropy

Some refresher notes:

- "The Boltzmann distribution is the so-called canonical distribution, meaning it maximizes entropy subject to a constraint on the expected value of energy." (viz, this is the MaxEnt principle. except for MaxEnt, the constraint is not on energy, but a set of features.)
- Define "Shanon Entropy" as $S_s = -k_B \sum p \log p$
- The "Boltzmann Entropy" S_B is the shanon entropy of the microcanonical ensemble: it maximizes the entropy (MaxEnt) for a fixed value of the energy. (MaxEnt: not the energy, but for a fixed set of features). (viz, $S_B = k_B \log \left(\varepsilon \frac{d\Omega}{dE} \right)$ with Ω being number of states, E the energy, ε a constant of dimension energy to make arg of the log dimensionless.)
- The "Gibbs Entropy" is the Shanon entropy, maximized for a system held to the constraint that energy is less-than-or-equal to E. (!) This gives $S_G = k_B \log \Omega$ (duhh, take $p = 1/\Omega$ for Ω states. For a non-sharp cutoff, the Shannon entropy is primal.).
- Gibbs and Boltzmann entropies give different results for N-particle systems when N is very small. Viz, an off-by-one error for N. In some ways, S_G is more correct (at low temp, quantum systems). See Jörn Dunkel and Stefan Hilbert (2014) "Consistent thermostatistics forbids negative absolute temperatures" Nature Physics DOI: 10.1038/NPHYS2815

Why does Yuret's MST work?

There is an interesting simplification that happens with minimum-spanning tree parsers driven by entropy. If we use Yuret's definition of the MI of word-pairs, then, Yuret says (I should re-read his stuff) that we should maximize the entropy

$$\sum_{w_l, w_r} MI(w_l, w_r) \tag{8}$$

Why? Why this, instead of the maybe "more obviously correct" sum:

$$\sum_{w_l, w_r} P(w_l, w_r) MI(w_l, w_r) \tag{9}$$

I think I can hand-wave the answer. The answer is that we don't really know the probability of $P(w_l, w_r)$ *for the given sentence!* We know $P(w_l, w_r)$ for a large corpus, but its somewhat of a mistake to assume that this identical to what it would be for expressing a particular idea in a certain specific way. Its possible that, to express the idea, the only sentences that one could ever possibly use would have $P(w_l, w_r)$ that strongly deviate from a large-corpus average. Unfortunately, there is no easy way of knowing what this sentence-specific $P(w_l, w_r)$ is. So, instead we make the uniform distribution assumption, that they're all the same, and thus get eqn (8) instead of (9). Does Yuret ever make this argument himself? Dunno.

A supporting argument is that we also ignored 3,4,5-point interactions as well. Which brings us to the next point: why should we expect a link-parse to work better than an MST parse? Because Yuret-MST ignores the valence of words, whereas the disjuncts don't! The disjuncts provide a better, more accurate way of capturing valency!

Entity Extraction

See Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates (2005) "Unsupervised Named-Entity Extraction from the Web: An Experimental Study". So: KNOWITALL utilizes a set of eight domain-independent extraction patterns to generate candidate facts. For example, the generic pattern "NP1 such as NPList2" ... "cities such as Paris,..."" Of course, this is not really unsupervised, since it uses human-generated search patterns ("such as") and also applies constraints (the targets must be proper nouns, which is not a-priori known).

3 March 2014

Start again, after long distraction.

Finding patterns

To problem. Consider an alphabet of $N = 5$ letters, $\alpha = \{A, B, C, D, E\}$ and a corpus built from those letters. The five letters occur with probability $p(w)$ with $w \in \alpha$. Assume the corpus consists entirely of pairs AB, CB and DE, each occurring equally often: so: $p(A,B) = p(C,B) = p(D,E) = 1/3$. From this, we can reconstruct that $p(A) = p(C) = p(D) = p(E) = 1/6$ and $p(B) = 1/3$. This follows because the corpus can be reduced to $\{AB, CB, DE\}$, so A occurs 1 out of 6 times, B two out of 6 times, etc. The total single-letter entropy is thus

$$\begin{aligned} h_{SING} &= -\sum_{w \in \alpha} p(w) \log_2 p(w) \\ &= -\frac{4}{6} \log_2 \frac{1}{6} - \frac{1}{3} \log_2 \frac{1}{3} \\ &= \frac{2}{3} - \log_2 \frac{1}{3} = 2.25163 \end{aligned}$$

By contrast, in a random 2-letter corpus, we expect all possible letter pairs to occur equally often, i.e. $p(w) = 1/5$, which would result in $h_{RAND} = -\log_2 1/5 = 2.321928$ and so we see that the total entropy for this corpus is less than the random corpus.

The total double-word entropy is

$$\begin{aligned} h_{PAIR} &= -\sum_{w_1,w_2\in\alpha} p(w_1,w_2)\log_2 p(w_1,w_2) \\ &= -\log_2 \frac{1}{3} = 1.5849625 \end{aligned}$$

Compare this to $h_{PR-RAND} = -\log_2 1/25 = 4.643856$ for the random 2-letter corpus. The pair-entropy is sharply lower.

What do we know about mutual information? We can also deduce that $p(*,B) = 2/3$, $p(A,*) = 1/3$ and so

$$\begin{aligned} MI(A,B) &= \log_2 p(A,B) - \log_2 p(A,*) - \log_2 p(*,B) \\ &= \log_2 3/2 = 0.585 \end{aligned}$$

and likewise $MI(C,B) = MI(A,B)$ while $MI(D,E) = \log_2 3 = 1.585$.

By contrast, in a random 2-letter corpus, we expect all possible letter pairs to occur equally often, i.e. $p(w_1,w_2) = 1/25$, which would result in an $MI(w_1,w_2) = \log_2 1 = 0$ for all word pairs.

Given this corpus, we wish to deduce the following answer: there is a cluster $\gamma = \{A,C\}$ and two link relations $R(\gamma,B)$ and $R(D,E)$ occuring with probabilities $p(\gamma,B) = p(A,B) + p(C,B) = 2/3$ and $p(D,E) = 1/3$. Note that $p(\gamma,*) = p(A,*) + p(C,*) = 2/3$ so that

$$\begin{aligned} MI(\gamma,B) &= \log_2 p(\gamma,B) - \log_2 p(\gamma,*) - \log_2 p(*,B) \\ &= \log_2 3/2 = 0.585 \end{aligned}$$

So how do we deduce this?

Well, consider the reduced space, with $N = 4$ letters: $\beta = \{\gamma,B,D,E\}$. In this space, only two pairs are observed in the corpus, γB and DE with probabilities as above. The single-letter probabilities are $p(D) = p(E) = 1/6$ and $p(\gamma) = p(B) = 1/3$. The single-letter entropy is

$$\begin{aligned} h^{red}_{SING} &= -\sum_{w\in\beta} p(w)\log_2 p(w) \\ &= -\frac{2}{6}\log_2\frac{1}{6} - \frac{2}{3}\log_2\frac{1}{3} \\ &= \frac{1}{3} - \log_2\frac{1}{3} = 1.9182958 \end{aligned}$$

This can be compared to the entropy of the random 4-word corpus: $h^{red}_{RAND} = -\log_2 1/4 = 2$. Note that

$$h^{red}_{RAND} - h^{red}_{SING} = 0.081704 > 0.070298 = h_{RAND} - h_{SING}$$

In other words, the reduced corpus shows more order than the comparable unreduced corpus! Interesting! The above can be written as:

$$h_{SING} - h^{red}_{SING} = 0.333334 > 0.321928 = h_{RAND} - h^{red}_{RAND}$$

What about the reduced pair entropy? For this case, we have

$$\begin{aligned}
h^{red}_{PAIR} &= -\sum_{w_1,w_2\in\beta} p(w_1,w_2)\log_2 p(w_1,w_2) \\
&= -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} \\
&= -\frac{2}{3} - \log_2\frac{1}{3} = 0.9182958
\end{aligned}$$

which can be compared to the random-pair entropy of $h^{red}_{PR-RAND} = -\log_2 1/16 = 4$. The comparable reduction is

$$h^{red}_{PR-RAND} - h^{red}_{PAIR} = 3.081704 > 3.0588935 = h_{PR-RAND} - h_{PAIR}$$

So again, this wins, but not by a lot. Re-ordering, this can be written as:

$$h_{PAIR} - h^{red}_{PAIR} = 0.6666667 > 0.643856 = h_{PR-RAND} - h^{red}_{PR-RAND}$$

So we seem to have two ways of winning: reducing the overall entropy, for for single letters, and for pairs, and also finding reductions that are strong, even compared to the reduced vocab.

Reductio ad absurdum? No.

What if we continue on this path, and (incorrectly) reduce to $N = 3$ letters, with $\delta = \{\gamma,\eta,D\}$ where $\eta = \{B,E\}$? Then $p(\eta) = p(B) + p(E) = 1/2$

$$\begin{aligned}
h^{rr}_{SING} &= -\sum_{w\in\delta} p(w)\log_2 p(w) \\
&= -\frac{1}{6}\log_2\frac{1}{6} - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{2}\log_2\frac{1}{2} \\
&= \frac{2}{3} - \frac{1}{2}\log_2\frac{1}{3} = 1.4591479
\end{aligned}$$

and the reduction inequality is

$$h^{red}_{SING} - h^{rr}_{SING} = 0.4591479 > 0.4150375 = h^{red}_{RAND} - h^{rr}_{RAND}$$

So this inequality allows an inappropriate reduction to take place. That implies that we must not use the SING inequality to obtain reductions!

For the pairs, $p(\gamma,\eta) = p(\gamma,B) = 2/3$ and $p(D,\eta) = p(D,E) = 1/3$ and everything else being zero. Thus one gets:

$$\begin{aligned}
h^{rr}_{PAIR} &= -\sum_{w_1,w_2\in\delta} p(w_1,w_2)\log_2 p(w_1,w_2) \\
&= -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} \\
&= -\frac{2}{3} - \log_2\frac{1}{3} = 0.9182958
\end{aligned}$$

so that

$$h^{red}_{PAIR} - h^{rr}_{PAIR} = 0 \not> 0.830075 = h^{red}_{PR-RAND} - h^{rr}_{PR-RAND}$$

Here, nothing is gained, so the pair inequality blocks the inappropriate reduction. Consider a differnt inappropraite reduction to $N = 3$: let $\varepsilon = \{\zeta, B, E\}$ with $\zeta = \{D, \gamma\}$ Then the pair probabilities are $p(\zeta, B) = p(\gamma, B) = 2/3$ and $p(\zeta, E) = p(D, E) = 1/3$ and again, there is no entropy reduction. The other groupings look to be equally ineffective.

Finding Patterns, General Formula

OK, recast the above section for the (semi-)general case of word-pairs (not structures in general). So, given a vocabulary of N words, we have $h_{RAND} = -\log_2 \frac{1}{N} = \log_2 N$ and $h^{red}_{RAND} = \log_2(N-1)$ so that for large N, $h_{RAND} - h^{red}_{RAND} = \log_2 N/(N-1) = \log_2(1 + 1/(N-1)) \approx 1/N$ and so we have a word-combine winner if we can combine words A and C into a cluster $\gamma = \{A, C\}$ such that

$$\begin{aligned}\frac{1}{N} &\lesssim h_{SING} - h^{red}_{SING} \\ &= -\sum_{w\in\alpha} p(w)\log_2 p(w) + \sum_{w\in\beta} p(w)\log_2 p(w) \\ &= -p(A)\log_2 p(A) - p(C)\log_2 p(C) + p(\gamma)\log_2 p(\gamma) \\ &= p(A)\log_2\left(1 + \frac{p(C)}{p(A)}\right) + p(C)\log_2\left(1 + \frac{p(A)}{p(C)}\right)\end{aligned}$$

where $p(\gamma) = p(A) + p(C)$. What's not clear: is this inequality *ever* broken? Or does it always hold? At any rate, from the previous example, it seems clear that we should not use the SING inequalities to obtain clusters.

For pairs, its clear that $h_{PR-RAND} - h^{red}_{PR-RAND} \approx 2/N$ which follows as above, given that $h_{PR-RAND} = 2\log_2 N$, etc. The corresponding inequality is now

$$\begin{aligned}\frac{2}{N} &\lesssim h_{PAIR} - h^{red}_{PAIR} \\ = &-\sum_{w_1,w_2\in\alpha} p(w_1,w_2)\log_2 p(w_1,w_2) + \sum_{w_1,w_2\in\beta} p(w_1,w_2)\log_2 p(w_1,w_2) \\ = &-\sum_{w\in\alpha\setminus A,C} [p(A,w)\log_2 p(A,w) + p(C,w)\log_2 p(C,w) - p(\gamma,w)\log_2 p(\gamma,w)] \\ &-\sum_{w\in\alpha\setminus A,C} [p(w,A)\log_2 p(w,A) + p(w,C)\log_2 p(w,C) - p(w,\gamma)\log_2 p(w,\gamma)] \\ &\quad - p(A,A)\log_2 p(A,A) - p(C,A)\log_2 p(C,A) + p(\gamma)\log_2 p(\gamma) \\ &\quad - p(A,C)\log_2 p(A,C) - p(C,C)\log_2 p(C,C)\end{aligned}$$

So...

8 March 2014

Morphology

Notes: https://en.wikipedia.org/wiki/Nonconcatenative morphology

25 March 2014

Information-Theoretic Clustering

New references:

- http://www.cs.utexas.edu/users/inderjit/public_papers/kdd_cocluster.pdf Information-Theoretic Co-clustering Inderjit S. Dhillon, Subramanyam Mallela, Dharmendra S. Modha
- http://pdf.aminer.org/000/472/364/a_generalized_maximum_entropy_approach_to_bregman_co_clustering_and.p A Generalized Maximum Entropy Approach to Bregman Co-clustering and Matrix Approximation Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, Dharmendra S. Modha Journal of Machine Learning Research 8 (2007) 1919-1986

30 March 2014

The below was going to be a breif note, but I'm turning it into a rough draft blog post. But after sleeping on it, it seems silly.

Freedom and Constraint

The concepts of freedom and constraint are central to the definition of algebra in mathematics. So for example, in group theory, the algebraic symbols denoting the elements of the group may be arranged freely, in any order desired. A given group is then defined as a 'presentation', a set of equivalences between different orderings. Thus, there is the notion of a 'free group', which is merely a set of symbols that can be written in arbitrary order, and no further constraints other than those of it being a group. Groups that aren't free are presented by a collection of equations, which state that one certain order of symbols is equivalent to another. One says that groups are 'equationally presented'.

A more complex example is the term algebra, where the terms may be arranged in free order; but the combination of the written symbols on the page are constrained to those of the 'signature' of the algebra. One then has the notion of an 'equational theory', which is a term algebra with additional equations between expressions, indicating which expressions should be taken as equivalent.

These have strong, and even precise analogues in linguistics. But first, continuing with the mathematical observations: the signature of a term algebra can be viewed as defining the 'syntax' of the symbolic notation: a Turing machine, tasked with the need to recognize the 'language' of the term algebra, would process input symbols one by one. It would appear that term algebras have a context-free syntax, and are thus recognizable by a push-down automata. That is, one must recognize the function symbol, the open and close parens, the commas separating arguments, and the constant symbols. The arbitrary-depth recursiveness is the only reason why the push-down is needed; otherwise the language seems 'almost regular'. (Hmm ... is there any formal definition/distinction of this case? i.e. for very simple constext-free languages, vs. 'more complex' ones? Not that I know of ...)

In linguistics, similar notions of freedom and constraint arise, but seem to be more of a surprise and mystery to linguists. Thus, for example, in [And12], Anderson describes the syntax and morphotactics of Kwakw'ala, a Wakashan language of coastal British Columbia. The syntax of the language (that is, the order in which the words can appear in a sentence) is very strict: the verb must be followed by a subject, optionally followed by the object, and then a prepositional phrase. Similarly adjectives must always preceed the noun. The language also has a rich morphology: words are assembled from stems and suffixes. The rules for assembling a word out of stems and affixes is refered to as the 'morphotactics'. In Kwakw'ala, it would appear that the morphotactics is utterly distinct from the syntax: here, object-denoting prefixes can preceed verbs, adjective-denoting suffixes follow a noun. Anderson finds this quite remarkable: the language has two distinct kinds of structure-imposing systems: the syntax and the morphtactics, and they are quite different. He notes that this dual structure in turn allows the same thing to be said in multiple ways. One may take meaning-parts, as morphemes, and glue them together morphtactically into words, and aranging these in a sentence. Allternately, one may take the meaning-parts separately, as individual words, and glue them together into a sentence, having a different sequence of the meaning-parts.

The part that struck me with Anderson's analysis is the similarity of the phenomena to the analogous behaviour formalized in mathematics. Lets first look at a second example: Lithuanian has a rich morphotactical structure: verbs and adverbs are conjugated, nouns and adjectives are declined; the rules for doing so are rather fixed and uniform, making adjustments mostly for phonological reasons (i.e. with exceptions based on constraints that come from the natural flow of the sound sequences constrained by the use of vocal cords, mouth, tongue and lips). Curiously, Lithuanian is almost devoid of syntactic constraint: word-order can be chosen freely (in the mathemaitcal sense!), and the meaning of the resulting sentences are esentially the same (if I am allowed to gloss over the notion that different word orders can serve to highlight or emphasize different themes and rhemes). So again: a language with very distinct syntax and morphtactics; in this case, the syntax being almost absent.

I used the theory of Link Grammar for performing structural linguistic analysis. The theory was originally developed to model syntactic structure, but it also appears to be entirely adequate for morphotactic analysis as well (certainly, for 'agglutinative' or 'concatenative' languages, with ongoing research into more complex morphologies). From the point of view of a linguist, Link Grammar appears to be 'just another theory of syntax', being a kind of dependency grammar. From the point of view of a mathematician, the situation is entirely more remarkable. It appears that the mathematical defintion of what constitutes a 'link grammar' is isomorphic to that of a 'categorical grammar', and that the correspondance is immediate and direct. Categorical grammars are interesting because they have a direct, formal mathematical definition that is studied and classified by mathematicians: roughly speaking, categorical grammars are 'non-symmetric compact closed monoidal categories'. The precise definition here has been championed by Bob Coecke ref [xxx]

It takes some study of category theory to understand what this means, but, roughly speaking, it means that sequences of sounds, morphemes, words are analyzed in sequential order: by means of short-distance groupings of left-right arrangements. This may sound silly, as, of course, sequential things occur in a sequence, but it helps highlight the difference between dependency grammars and phrase-structure grammars, or computer-science grammars in general. An example of a 'computer-science grammar' is the so-called 'context-free grammar'. A hallmark of such grammars is that they allow recursion to arbitrary depths. An English-language example would be the sequence of sentences: "This is a house", "This thing is a house." "This thing is a thing that is a house." "This thing is a thing that is a thing that is ... a house." The example is silly because no one ever talks that way. The phrase-structure analysis of this would be "(S (VP (NP (VP (NP ...)))))", with the heirarchical arrangement emphsized. Dependency grammars can also parse such sentences, but here, the arrangment of dependencies are in the form of arrows that point from head word to dependent word; the arrows are only rarely long-range, and usually point to the immediately-surrounding words. There is strong psycho-linguistic evidence for such local structure in language, see for example [xxxx]. That is, the workings of the human mind is not recursive in nature, pushing and popping an arbitrarily deep stack as each new noun-phrase or verb-phrase is enountered. Indeed, psychological studies with constructed sentences similar to the above, but varying the 'thing' and 'house' at each depth, show that humans quickly loose track after just two or three nestings [need ref]. In essence, the human mind is adapted for linear sequential analysis, and long-range order between words is challenging: this is the psycho-linguistic argument for dependency grammars. From the mathematical point of view, the statement is that human languages are not so much context-free, as they are non-symmetric compact closed monoidal categories. That Link Grammar is an example of the latter is why it seems so appropriate to use for syntactic and morpho-tactic structural analysis.

Which theories of langauge are mathematically isomorphic? That is, Link Grammar and categorical grammars seem to be isomorphic because there is a simple way of translating the one into the other, and vice-versa (although no formal mathematical proof of this has been written down). A mathematical proof of equivalence is a mechanical device: given one representation, one turns a crank to obtain the other. More generally, its been argued that phrase-structure grammars and dependency grammars are equivalent in the same sense: there is an algorithm that converts the one into the other, and v.v.[where's the ref for this?]. Does this mean that non-symmetric compact closed monoidal categories have context-free grammars as their internal language, and that every context-free language has a corresponding monoidal category? I think not, but the answer to that, the 'why not', and the 'what, then, is the difference?' is entirely unclear. Clarifying these relationships seems important for putting language study on a firmer basis.

Anyway, the point here was to clarify the boundaries between freedom and constraint. Traditional phrase-structure grammars were inspired by notions from 1960's-era computer science, but now seem slavishly wedded to the same ideas, to the detriment of closer linguistic understanding. Dependency grammars seem to be more psycho-linguistically valid, but have suffered from a lack of mathematical formalism that ellucidates freedom and constraint. This lack of formalism makes it hard to explain why some constructions are grammatically correct, and others are not. It also seems to draw an artificial and confusing line between syntactic and morphotactic structure, when, in fact, these really should be taken as a part of a continuum of structure. I see no reason why a single grammar could not also describe the allomorphic variations in pronunciation. After all, these are just a set of rules that govern how a morpheme is pronounced, and this is essentially a linear, sequential phenomenon, with only (mostly?) nearest-neighbor morphemes affecting one-another. The nearest-neighbor aspect of this fairly well screams out 'dependency'.

Another curious and interesting language-constraint structure emerges with the study of idioms and institutionalized, set phrases. Because these are 'phrases', built of 'words', it would naively seem that tese lie in the domain of syntax. But this is misleading. Institutionalized utterances are those where neither the word-choice nor the word-order are directly governed by syntax alone, but instead seem to be frozen into a fixed form. So, one talks of the 'time of day', but never of 'pressure of air' or 'height of mountain' – "What pressure of air should I put in this tire?" "What height of mountain do you plan to climb?" "What time of day do you expect to come over?". There is nothing syntactic that prevents such a choice of wording, and the semantic meaning is more or less clear: its just that such word arrangements simply don't happen. Its as if the lexis for English has a phrase in it: "time-of-day", which should be treated as a single word, rather than the three words it is written as. This provides the first hint of the role of probability in this discussion: the probability of seeing the phrase 'height of mountain' in English approaches zero: in fact, this text that you are reading right now just might be the only place ever in the history of the world in which this phrase has appeared ... despite it being 'grammatically valid'. Freedom and constraint aren't just governed by true-false distinctions, but by probabilities. The question then is, 'what is the most natural way in which to express such probabilities?'

The last is not just some idle intellectual question, but in fact, an engineering question: the proper structure should have an immediate and direct effect on how well, and how quickly a language could be learned, via unsupervised machine-learning algorithms. A universal but naive attitude in the artificial-intelligence community is that 'oh. everything is a neural net, and we should use neural nets to build AI.' Less frequently, one may seem a similar attitude regarding Hidden Markov Models (HMMs). The fact that such naive approaches lead to algorithms that fail to converge quickly leads to ideas such as 'deep learning': a modification that explicitly splits a problem into layers, with explicit feedback between layers. Another variation used to escape the trap is to explicitly model what is un-known: this is the notion of maximum entropy (MaxEnt). Traditional AI was also founded on logic and reasoning, and, for many decades, AI was dominated by the exploration of boolean-valued logic. By this I mean anything with crisp, sharp truth values: whether first-order logic, boolean satisfiability, satisfiability-modulo-theories, stable-model semantics, and so on. Another corner was fuzzy logic, but that didn't seem to have legs. Notions of maximum entropy and probability can be unified: thus, one has Markov Logic Networks (MLN). What I'm wodering about here is that maybe none of these approaches are correct, because they are ignoring the actual structure that is in front of us.

So, perhaps, the correct approach is not to marry maximum entropy with first-order logic, but to marry maximum entropy to dependency grammars (or, equivalently, to appropriate monoidal categories). The question then becomes: what is the appropriate monoidal category? Picking the wrong one will lead to disasterous machine learning performance (this, I think, is the lesson from neural networks). Picking something too easy doesn't get you far enough (the lesson of HMM's – excellent for certain classes of problems, but lacking in scale). There are more choices than that: but the chices, and their inter-connectedness, and trade-offs, seem to be unarticulated. For any given monoidal category, there would seem to be some probabilistic model corresponding to that category's internal langauge. That is, there is a way of describing the transition probabilities from state to state. Indeeded, (finite) monoidal categories, in the form of acts, can be partly understood to be finite state machines acting on a set. The probabilistic generalization of this leads both to probabilistic and quantum finite automata, with the former having a strong resemblance, if not identity, to Markov chains, with the corresponding acts being HMM's. My hypothesis is that probabilistic dependency grammars will lead to machine learning algos that converge more rapidly than the similar-but-different HMM that can also be mapped onto the same problem. Unfortunately, my hypothesis is impedded by my lack of understanding of precisely, exactly how the different approaches named above may be equivalent, isomorphic, or merely similar.

2 April 2014

Link Grammar and Finite State Transducers

Claim: Finite state transducers, such as those used for morphological analysis, can be mapped to a Link Grammar. This implies that Link Grammar parsing can be used for morphological analysis, thus unifying syntactic parsing and morphological analysis into a unified framework.A finite state transducer (FST) is defined as:

- A set of states Q
- A set Σ of input symbols (surface form)
- A set Γ of output symbols (lexicalized form)
- A transition function $\delta \subset Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$

A member $(r, a, b, s) \in \delta$ should be thought of as the arrow from state r to state s, the arrow being taken when the input symbol is a and as a result producing the output symbol b. The corresponding link-grammar dictionary entry for this would be

```
a.b: r- & s+;
```

This states that no linkage is possible, unless the previous link resulted in the emission of the r+ connector. No transition to the next state is possible, unless that state has an s- connector on it.

The current link-grammar notation a.b is awkward for printing, and perhaps some new style is needed to distinguish the output to be printed from the input that is recognized. Thus, perhaps, it would be better to invent a new notation, perhaps a$b to denote that a is recognized, and that b is printed.

Note that the above definition of link-grammar rules results in a very simple, linear linkage: state transitions follow one-another in linear order. Link grammar allows richer, more complex linkage diagrams, and so the question arises: can a given FST be compactified into a smaller system by making use of the richer possibilities that link-grammar offers? How can this compactification be acheived?

Suppose that the FST δ includes as a subset the state transitions $\{(r,a,?,s),(s,\varepsilon,\varepsilon,t),(s,b,?,t),(t,c,?,u)\}$. The symbol ? is used here as a don't-care state, as it is irrelevant to the discussion that follows. The above state transitions indicates that when the system is in state s, it may spontaneously transition to state t, or may do so upon reading b. That is, the presence of b is optional in the state transition. The "natural" way of indicating this with link-grammar notation is using the link-grammar dictionary entries:

```
a: r- & s+;
b: t+;
c: {t-} & s- & u+;
```

Because the transition $(s,\varepsilon,\varepsilon,t)$ reads no input, and produces no output, the state transitions would more likely be written as $\{(r,a,?,t),(r,a,?,s),(s,b,?,t),(t,c,?,u)\}$, that is, by collapsing the transition $(s,\varepsilon,\varepsilon,t)$ into the prior state. This would have the entries

```
a: r- & (s+ or t+);
b: s- & t+;
c: t- & u+;
```

How should it be understood? These are, in fact, two distinct, inequivalent LG grammars, as can be seen by considering the parse of the strings "ac" and "abc" for the two cases.

When would weighting schemes interfere? when would output interfere?

15 April 2014

Elegant Normal Form

Or, more precisely, "Minimal Normal Form". Instead of writing out LG disjuncts in long strings of DNF or CNF, where they blow up into the thousands or tens of thousands, we really need to write then in Craig Holman's "Elegant Normal Form", (http://www.patterncraft.com/Blog/Blog-080609.html#ElegantNormalForm) format. This is to be done by entropy minimization, in two different ways: first, ENF reduces the total count of terms, for just one single expression. Second, and maybe more important: different words will share significant subsets of the ENF expression. So, for example, the LG English dicts define:

```
    <verb-rq>: Rw- or ({Ic-} & Q- & <verb-wall>) or [()];
```

which is (1) in ENF, not DNF or CNF, and (2) shared by several dozen words. There should be a strong push to discover such common sub-expressions across many words.

28 April 2014

Isotopy

The concept of "isotopy" (https://en.wikipedia.org/wiki/Isotopy_(semiotics)) was introduced by Algirdas Greimas in 1966. Example: "I drink some water", with the meanings of "drink" and "water" re-inforcing one-another. But this is eactly what the Mihalcea WSD algo does, eh?

14 May 2014

Tree similarity

"Similarity Evaluation on Tree-structured Data" Rui Yang Panos Kalnis Anthony K. H. Tung SIGMOD 2005 June 14-16, 2005. Quote from abstract: "propose to trans- form tree-structured data into an approximate numerical multidimensional vector". Funny – that's what Bob Coecke proposes for any kind of monoidal category: vector spaces being a special monoidal cat. Hmmm.

Approaches:

- Tree-edit distance: many variants proposed, all high cpu/memory intensive.
- convert tre to pre or post-oerder, and use string edit distance.
- Convert to binary tree. For combo trees, this makes sense, due to the associative property of most of the operators. In particular, in combo any oper that can have multi-sibilings is also associative and thus convertible to binary tree. What's more, trees with binary branch distance of zero really are equivalent for us: See Figure 4 in above reference. Yay! this fits very very well with combo!.

29 June 2014

Morphology Basic Claims

We have two tasks to adress: the automated discovery of morpheme boundaries, and the automated discovery of "morphtactics", the syntax of connected morphemes. We make two claims: first. the automated discovery of morpheme boundaries can be accomplished by searching for breaks between word-parts that have the lowest mutual information. Second, the discovery of morphotactics is identical to the discovery of syntax, as outlined above.

The simplest approach to finding the breaks between morphemes is to randomly break up words into two parts. A worked example of this is given below. Several questions present themselves:

- To discover morphemes of words that split into three or more parts, is it better to always split pairwise, and then perform recursion, or is it easier to split into multiple parts immediately? Perhaps the answer is language-dependent?
- Does one obtain better mophological splits by immediately including morphtactic analysis, or can this be deferred?

Morphology Worked Example

OK, this will be tedious, but I see no alternative. Suppose we have the corpus “test gift tester testy gifty tester gifter” so that “tester” appears twice in the corpus. Explore all possible splits into two parts. The 4-letter splits split 3 ways, the 5-letter splits split 4 ways, etc. so there is a total of N(*,*)=3+3+5+4+4+5+5=29 pairs. All pairs appear once, except for tester, which appears twice. Viz.

P(x,y)=1/29 for (x,y) in {(t,est), (te,st), (tes,t), (g,ift), (gi,ft), (gif,t), (t,esty), (te,sty), (tes,ty), (test,y), (g,ifty), (gi,fty), (gif,ty), (gift,y), (g,ifter), (gi,fter), (gif,ter), (gift,er), (gifte,r)}

and

P(x,y)=2/29 for (x,y) in {(t,ester), (te,ster), (tes,ter), (test,er), (teste,r)}

There is a bit of a procedural error in the above; we would like to discover the “null suffix”, that is, that “test”, “gift”, with nothing following it, are morphemes, so that the possible suffixes are “-y”, “-er” and “-nothing”. However, the above failed to count this possibility separately. Thus, given the above data, what we expect to find are two roots: “gif-” and “tes-” and three suffixes: “-t”, “-ty” and “-ter”. This is not so bad. If we did split and count in such a way as to allow a null suffix, it would be ambiguus as to whether the stems end with a “t” or not. That is, the with-t and without-t stems would have been equally likely... Anyway, moving on... the possible splits are shown in the table below 1:

Next, lets do the partial sums. Recall the notation for the partial summation of pairs. writing P(x,y) for the probability of observing the *ordered* pair of items (x,y), the partial sums are:

$$P(x,*) = \sum_{y \in Y} P(x,y)$$

and

$$P(*,y) = \sum_{x \in X} P(x,y)$$

The left-hand sums are the column totals in the table above, table 1

P(t,*) = (1+1+2)/29 = 4/29 = P(te,*) = P(tes,*)

Table 1: Word Split Table

	g	gi	gif	gift	gifte	t	te	tes	test	teste	row total
ifter	1										1
fter		1									1
ter			1					2			3
er				1					2		3
r					1					2	3
ifty	1										1
fty		1									1
ty			1					1			2
y				1					1		2
ift	1										1
ft		1									1
t			1					1			2
ester						2					2
ster							2				2
esty						1					1
sty							1				1
est						1					1
st							1				1
column total	3	3	3	2	1	4	4	4	3	2	29

The above is a sparse matrix showing the possible word splits. empty cells contain a count of zero.

P(g,*) = (1+1+1)/29 = 3/29 = P(gi,*) = P(gif,*)

P(test,*) = (1+2)/29 = 3/29

P(teste,*) = 2/29

P(gift,*) = 2/29

P(gifte,*) = 1/29

Next, the right-hand partial sums. These are the row totals for the table above, table 1:

P(*,est) = 1/29 = P(*,st) = P(*,esty) = P(*,sty) =P(,ift) = P(*,ft) = P(*,ifty) = P(*,fty) = P(*,ifter) = P(*,fter)

P(*,t) = (1+1)/29 = 2/29 = P(*,ty) = P(*,y)
P(*,ester) = 2/29 = P(*,ster)
P(*,ter) = (1+2)/29 = 3/29 = P(*,er) = P(*,r)

Now, the compute the MI (we use log=log_2 in all cases below, for measuring the entropy in units of bits). Recall the definition of mutual information for *ordered* pairs, previously discussed and given above:

$$MI(x,y) = \log_2 \frac{P(x,y)}{P(x,*)P(*,y)}$$

So, working these by hand:

MI(t,est) = log P(t,est)/P(t,*)P(*,est) = log (1/29)(29/4)(29/1) = log(29/4) = 2.857981 = MI(te,st) = MI(t,esty) = MI(te,sty)

MI(g,ift) = log P(g,ift)/P(g,*)P(*,ift) = log (1/29)(29/3)(29/1) = log(29/3) = 3.273018 =MI(gi,ft) = MI(g,ifty) = MI(gi,fty)=MI(g,ifter)=MI(gi,fter)

MI(tes,t) = log P(tes,t)/P(tes,*)P(*,t) = log (1/29)(29/4)(29/2) = log(29/8) = 1.857981 =MI(tes,ty)

MI(gif,t) = log P(gif,t)/P(gif,*)P(*,t) = log (1/29)(29/3)(29/2) = log(29/6) = 2.273018

MI(test,y) = log P(test,y)/P(test,*)P(*,y) = log (1/29)(29/3)(29/2) = log(29/6) = 2.273018

MI(gif,ty) = log P(gif,ty)/P(gif,*)P(*,ty) = log (1/29)(29/3)(29/2) = log(29/6) = 2.273018

MI(gift,y) = log P(gift,y)/P(gift,*)P(*,y) = log (1/29)(29/2)(29/2) = log(29/4) = 2.857981

MI(gif,ter) = log P(gif,ter)/P(gif,*)P(*,ter) = log (1/29)(29/3)(29/3) = log(29/9) = 1.688056

MI(gift,er) = log P(gift,er)/P(gift,*)P(*,er) = log (1/29)(29/2)(29/3) = log(29/6) = 2.273018

MI(gifte,r) = log P(gifte,r)/P(gifte,*)P(*,r) = log (1/29)(29/1)(29/3) = log(29/3) = 3.273018

MI(t,ester) = log P(t,ester)/P(t,*)P(*,ester) = log (2/29)(29/4)(29/2) = log(29/4) = 2.857981
= MI(te,ster)

MI(tes,ter) = log P(tes,ter)/P(tes,*)P(*,ter) = log (2/29)(29/4)(29/3) = log(29/6) = 2.273018

MI(test,er) = log P(test,er)/P(test,*)P(*,er) = log (2/29)(29/3)(29/3) = log(58/9) = 2.688056

MI(teste,r) = log P(teste,r)/P(teste,*)P(*,r) = log (2/29)(29/2)(29/3) = log(29/3) = 3.273018

Phew. I think that's all of them. So, what can we conclude? The basic claim is that the morpheme boundaries occur at the places where the letters are the least sticky, the most likely to be de-correlated, i.e. those with the lowest MI. In the above, these are: MI(gif,ter)=1.69 followed by MI(tes,t)=MI(tes,ty)=1.86. These are the most likely splits for these three words. Lets look up each possible split, for each word. We get:

Word	Split	MI	Split	MI	Split	MI	Split	MI	Split	MI	Best
gift	(g,ift)	3.27	(gi,ft)	3.27	(gif,t)	2.27					(gif,t)
gifty	(g,ifty)	3.27	(gi,fty)	3.27	(gif,ty)	2.27	(gift,y)	2.86			(gif,ty)
gifter	(g,ifter)	3.27	(gi,fter)	3.27	(gif,ter)	1.69	(gift,er)	2.27	(gifte,r)	3.27	(gif,ter)
test	(t,est)	2.86	(te,st)	2.86	(tes,t)	1.86					(tes,t)
testy	(t,esty)	2.86	(te,sty)	2.86	(tes,ty)	1.86	(test,y)	2.27			(tes,ty)
tester	(t,ester)	2.86	(te,ster)	2.86	(tes,ter)	2.27	(test,er)	2.69	(teste,r)	3.27	(tes,ter)

The best results from the above table are summarized below

Word	Lowest MI split(s)	MI
gift	(gif,t)	2.27
gifty	(gif,ty)	2.27
gifter	(gif,ter)	1.69
test	(tes,t)	1.86
testy	(tes,ty)	1.86
tester	(tes,ter)	2.27

What looks like the best split has been found; it certainly matches what was expected. Yay! After this, link-type clustering proceeds just as before, as if these were distinct words. That is, the above has 6 distinct link types; clustering will then proceed discover one link type, between the cluster {gif, tes} and {t,ty,ter}.

Morfesssor

An alternative algorithm is presented in:

- Mathias Creutz Krista Lagus, "Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0" http://users.ics.aalto.fi/mcreutz/papers/Creutz05tr.pdf

That algorithm works only for concatenative languages, and does not provide a morphotactic structure; that is, it cannot learn the grammar governing the morphemes. It also requires several (plausable) assumptions about Bayesian priors. One assumption is that morpheme frequency follows a modified Zipfian distribution, this is used to make estimates for morphemes that are observed only once in the corpus. Another assumption is is that the morpheme length distribution can be approximated by either a Poisson or a (two-parameter) gamma distribution.

12 July 2014

Link-type discovery, worked example

In keeping with the previous, lets look at a super-simplified version of link-type discovery, continuing imediately from the previous morpheme-discovery example. We begin with the intial observations, given in the table below:

Pair	Initial Link Type	# observations
gif–t	GA	1
gif–ty	GB	1
gif–ter	GC	1
tes–t	TA	1
tes–ty	TB	1
tes–ter	TC	2

The "initial link type" is handed out randomly; the actual letter string has no bearing on the outcome. Notice the above has 6 different, unique link types. These correspond to the following link-grammar dictionary, written in the classic link-grammar notation:

Algorithm 1 Morpheme grammar

```
    gif.=: GA+ or GB+ or GC+;
    tes.=: TA+ or TB+ or TC+;
    =t: GA- or TA-;
    =ty: GB- or TB-;
    =ter: GC- or TC-;
```

From the above initial dictionary, we want to deduce that a single link type is sufficient to full describe what is happening. That is, we wish to discover the following dictionary:

```
    gif.= tes.=: LL+;
    =t =ty =ter: LL-;
```

This is inuitively obvious, because the morphemes obviously form a clique: each stem has been observed with each suffix. Technically, this is a bipartite clique or complete bipartite graph of order (2,3). Here, we see it immediately; however, in general, it is very hard to search for bipartite cliques in a grammar; general algorithms are provably NP-complete and run in exponential time.

So how should we find grammar reductions? How is this to be done?

Out vocabulary consists of N=5 morphemes α ={gif.=, tes.=. =t, =ty, =ter}. We begin by recomuting the MI for observed pairs, once-again starting with the initial corpus “test gift tester testy gifty tester gifter”, same as before, with “tester” appearing twice in the corpus. This time, se split strictly according to the learned morphology. The word split table is:

	gif	tes	row total
ter	1	2	3
ty	1	1	2
t	1	1	2
column total	3	4	7

Note that this table is a stric subset of the previous table; the column and row totals are completely unchanged. However, the total number of observations has diminished from 29 to 7, and so all P amd MI values need to be recomputed. Proceeding long-hand, as before:

P(x,y)=1/7 for (x,y) in {(tes,t), (gif,t), (tes,ty), (gif,ty), (gif,ter)}

and

P(x,y)=2/7 for (x,y) in {(tes,ter)}

The partial sums are:

P(gif,*) = (1+1+1)/7 = 3/7

P(tes,*) = (1+1+2)/7 = 4/7

P(*,t) = 2/7 = P(*,ty)

P(*,ter) = 3/7

The MI values are all different, as well:

MI(gif,t) = log P(gif,t)/P(gif,*)P(*,t) = log (1/7)(7/3)(7/2) = log (7/6) = 0.222392 = MI(gif,ty)

MI(gif,ter) = log P(gif,ter)/P(gif,*)P(*,ter) = log (1/7)(7/3)(7/3) = log (7/9) = -0.362570

MI(tes,t) = log P(tes,t)/P(tes,*)P(*,t) = log (1/7)(7/4)(7/2) = log (7/8) = -0.192645 = MI(tes,ty)

MI(tes,ter) = log P(tes,ter)/P(tes,*)P(*,ter) = log (2/7)(7/4)(7/3) = log (7/6) = 0.222392

Note that three of the MI values are negative, and three are positive.

Following the previous formulas, we compute the total pair entropy:

$$\begin{aligned} h_{PAIR}^{observed} &= -\sum_{w_1,w_2\in\alpha} p(w_1,w_2)\log_2 p(w_1,w_2) \\ &= -\frac{5}{7}\log_2\frac{1}{7} - \frac{2}{7}\log_2\frac{2}{7} = 2.521641 \end{aligned}$$

This is a bit of a misnomer, or misleading; we are actually computing the link-entropy: so the set is actually $\beta = \{GA, GB, GC, TA, TB, TC\}$ the first five of which were observed once, and the last was observed twice. So really we should write:

$$h_{PAIR}^{observed} = -\sum_{t\in\beta} p(t)\log_2 p(t)$$

with $p(t)$ being the probability of observing link-type t.

The above is the observed entropy, given the corpus, and the grammar shown in listing 1. However, this grammar does not have any probability indicators attached to it, so that if it was used to generate a corpus, the entropy would be different. Basically, the probability of observing any of the link-types would be identical, and so the entropy would be:

$$\begin{aligned} h_{PAIR}^{generated} &= -\sum_{t\in\beta} p(t)\log_2 p(t) \\ &= -\frac{6}{6}\log_2\frac{1}{6} = 2.584963 \end{aligned}$$

This is obtained by observing that there are 6 link types in the set β and so, if chosen equi-probably, the resulting entropy is just $\log_2 6$. For a given number N of link types, the entropy of the generated grammar will be $\log_2 N$, for this extremely simply type of grammar, where all disjuncts have only one connector in them. The generated entropy will always be maximal for the grammar, as the observed distribution will surely never be equi-distributed. Thus, we have as a general principle:

$$h^{observed} \leq h^{generated}$$

Note that the equi-distributed link-types is the same as having each of the words in the corpus appear with equal frequency. The morphemes, however, do NOT appear with equally frequency (although individually, all stems do, and all suffixes do).

Link type reductions can be many ways. In each case, we look to see if adding a new word to a category improves the score. The possibilites are:

1. Group =ter and =ty together.
2. Group =ter and =t together.
3. Group =ty and =t together.
4. Group gif.= and tes.= together.

After this, we have more reductions:

1.a. Add =t to {=ter, =ty}, and finally group together gif.= and tes.=
1.b. Group together gif.= and tes.=, and finally, add =t to {=ter, =ty}
2.a, 2.b. 3.a 3.b variations of above
4.a. Group =ter and =ty together, then add =t.
4.b. Group =ter and =t together, then add =ty.
4.c. Group =ty and =t together, then add =ter.

This gives 9 different orders in which the reductions can take place. Actually, only 6: case 1b and 4a are the same, as are 2b=4b and 3b=4c. Lets do at least some of them.

Case 1. Let $\gamma = \{$=ter, =ty$\}$. Then the link types GB and GC need to be consolidated: GG={GB, GC} and likewise TT={TB, TC}. The dictionary becomes

```
gif.=: GA+ or GG+;
tes.=: TA+ or TT+;
=t: GA- or TA-;
=ty =ter: GG- or TT-;
```

The observed pair probabilities become:

p(GA) = p(gif,t) = 1/7 = p(TA) = p(tes,t)

p(GG) = p(gif,ter) + p(gif,ty) = 2/7

p(TT) = p(tes,ter) + p(tes,ty) = 3/7

So the observed entropy is now

$$h_{PAIR}^{red1.} = -\frac{2}{7}\log_2\frac{1}{7} - \frac{2}{7}\log_2\frac{2}{7} - \frac{3}{7}\log_2\frac{3}{7} = 1.842371$$

The generated entropy is $h_{gen}^{red1.} = \log_2 4 = 2$ since there are four total link types in the reduced grammar. Pursuant to equation 6, we should add the log of the cardinality of the word-sets. Here, only one word-set has a cardinality greater than one: {=ty, =ter}. So, one gets:

$$h_{gen}^{wrds1.} = \log_2 2 = 1$$

The conditional entropy, based on the textual observations, is

$$h_{obs}^{wrds1.} = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.970951$$

Case 1.a. Let $\delta = \{$=ter, =ty, =t$\}$. Then the link types GA and GG need to be consolidated: G={GA, GG} and likewise T={TA, TT}. The dictionary becomes

```
 gif.=: G+;
 tes.=: T+;
=t =ty =ter: G- or T-;
```

The observed pair probabilities become:

p(G) = p(gif,t) + p(gif,ty) + p(gif,ter) = 3/7

p(T) = p(tes,t) + p(tes,ty) + p(tes,ter) = 4/7

So the observed entropy is now

$$h_{PAIR}^{red1.a.} = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.985228$$

The generated entropy is $\log_2 2 = 1$ since there are only two link types in the grammar. The word-counting entropy for the set δ contributes an additional

$$h_{gen}^{wrds1.a.} = \log_2 3 = 1.584963$$

while the observed entropy is

$$h_{obs}^{wrds1.a.} = -\frac{4}{7}\log_2\frac{2}{7} - \frac{3}{7}\log_2\frac{3}{7} = 1.556657$$

Case 1.b. Let $\gamma = \{$=ter, =ty$\}$ as before, and $\varepsilon = \{$gif.=, tes.=$\}$. The link types consolidate: EA={GA, TA} and EM={GG, TT}. The dictionary becomes

```
gif.= tes.=: EA+ or EM+;
=t: EA−;
=ty =ter: EM−;
```

The observed pair probabilities become:

p(EA) = p(gif,t) + p(tes,t) = 2/7

p(EM) = 5/7

So that the entropy is

$$h_{PAIR}^{red1.b.} = -\frac{2}{7}\log_2\frac{2}{7} - \frac{5}{7}\log_2\frac{5}{7} = 0.863121$$

The generated entropy is $\log_2 2 = 1$ since there are only two link types in the grammar. The word-set counting probability adds

$$h_{gen}^{wrds.1.b.} = 2\log_2 2 = 2$$

while the observed probabilities are

$$h_{obs}^{wrds.1.b.} = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} - \frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 1.956179$$

Other cases. Case 2.a. and case 2.b. are identical to cases 1.a. and 1.b. because =t and =ty are interchangeable, from the probability point of view.

Case 3.a. and case 3.b. are similar, but with different probabilities.

Case 4. and the subcases are different, but not illuminating.

Final Case. The final consolidation gives $\gamma = \{$=ter, =ty,=t$\}$, and $\varepsilon = \{$gif.=, tes.=$\}$. The dictionary becomes

```
gif.= tes.=: LL+;
=t =ty =ter: LL−;
```

The observed pair probabilities become:

p(LL) = 7/7

So that the entropy is

$$h_{PAIR}^{final} = -\frac{7}{7}\log_2\frac{7}{7} = 0$$

The generated entropy is $\log_2 1 = 0$ since there is only one link type in the grammar. The word-set counting probability adds

$$h_{gen}^{wrds.fin} = \log_2 2 + \log_2 3 = 2.584963$$

while the observed word count is

$$h_{obs}^{wrds.fin} = -\frac{4}{7}\log_2\frac{2}{7} - \frac{3}{7}\log_2\frac{3}{7} - \frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 2.541885$$

Summary. The table below summaries these results. The sum columns show the entropy according to the equation 6 for the observed frequencies, and the generated frequencies.

	h_{obs}^{red}	h_{gen}^{red}	h_{obs}^{wrds}	h_{gen}^{wrds}	$h_{obs}^{red}+h_{obs}^{wrds}$	$h_{gen}^{red}+h_{gen}^{wrds}$
Initial	2.521641	2.584963	0	0	2.521641	2.584963
Case 1.	1.842371	2	0.970951	1	2.813322	3
Case 1.a.	0.985228	1	1.556657	1.584963	2.541885	2.584963
Case 1.b.	0.863121	1	1.956179	2	2.819299	3
Final	0	0	2.541885	2.584963	2.541885	2.584963
Case 3.	1.950212		1.0		2.950212	
Case 3.a.	0.985228		1.556656		2.541884	
Case 3.b.	0.985228		1.985228		2.970456	
Case 4.	1.556657		0.985228		2.541885	

Arghhh. Such a simple case, so much complexity... anyway, the case 3 and 4 are computed from the script “link-type/gifty.scm” in this same directory.

Conclusions: based purely on entropy maximization, all cases advance, but none go to the final case. But we are not imposing any ’complexity penalty’ on this.

Results on some alternate distributions, for this ranking: “tester testy test gifter gifty gift”

- Pure Zipf: $(rank)^{-1.0}$: none advance ($h_{initial} = 2.281979$ and $h_{final} = 2.293598$)
- Zipf $(rank)^{-1.05}$:none advance ($h_{initial} = 2.251204$ and $h_{final} = 2.263603$)
- Zipf $(rank)^{-1.5}$:none advance ($h_{initial} = 1.930661$ and $h_{final} = 1.948128$)

None of these advance because the initial and final entropies are so very close. But, as before, there are advnces, with the biggest ones to case 4.c and 3.b. The alternative rankings “tester testy test gift gifty gifter” and “tester gifter testy gifty test gift” give only slightly different results.

Link-type discovery, better example

In the previous, the unified link-type discovery is inevitable, so a more complex version is needed, with a less-obvious outcome. So lets take the original example of link-type discovery, and add some confounding link types. We begin with the intial observations, plus some extras, given in the table below:

Algorithm 2 Example morpheme grammar

```
gif.=: GA+ or GB+ or GC+;
tes.=: TA+ or TB+ or TC+;
blo.=: BB+ or BF+;
=t: GA− or TA−;
=ty: GB− or TB− or BB−;
=ter: GC− or TC−;
=fu: BF−;
```

A more complex grammar showing morpheme linkages.

Table 2: Example Link Frequency Table

Pair	Initial Link Type	# observations
gif–t	GA	1
gif–ty	GB	1
gif–ter	GC	1
tes–t	TA	1
tes–ty	TB	1
tes–ter	TC	2
blo–ty	BB	3
blo-fu	BF	1

Example distribution of link frequencies obtained from an example corpus.

The addition of "bloty" to the link table, and with a strong weight, will tend to derail the consolidation of the =ty suffix with the others. The addition of "blofu" helps make sure that there's some confusion about the "blo=" stem.

The corresponding link-grammar dictionary is:

From the above initial dictionary, we hope to deduce one word class that contains gif.= and tes.= and another that contains =t and =ter; eactly how the rest plays out is unclear. Lets begin by starting with the un-clustered entropy, and then see what hapens if we try various different clusters. So, as before, let $\beta = \{GA, GB, GC, TA, TB, TC, BB, BF\}$ and write:

$$\begin{aligned} h_{PAIR}^{observed} &= -\sum_{t\in\beta} p(t)\log_2 p(t) \\ &= -\frac{6}{11}\log_2\frac{1}{11} - \frac{2}{11}\log_2\frac{2}{11} - \frac{3}{11}\log_2\frac{3}{11} \\ &= 2.845351 \end{aligned}$$

with $p(t)$ being the probability of observing link-type t. Since there are 8 different link types, the generated entropy is $h_{PAIR}^{generated} = \log_2 8 = 3$. The different between these two is $h^{gen} - h^{obs} = 0.154649$. The observed corpus also has 8 words in it (not counting multiplicity): this is by design; before reduction, there is always exactly one link type for each morpheme pair.

Lets look at several cases:

1. Group gif.= and tes.= together.
2. Group gif.= and blo.= together.
3. Group =t and =ty together.
4. Group =ty and =fu together.
5. Group =ter and =fu together.

Here, we expect case 1 to go easily, cases 2 and 3 to to be ambiguous or blocked, case 4 to be weakly blocked, and case 5 to be strongly blocked. So, proceeding:

Case 1. Group gif.= and tes.= together. Let $\gamma = \{$gif.=, tes.=$\}$. Then the link types G* and T* need to be consolidated: A=$\{$GA,TA$\}$ and likewise B=$\{$GB,TB$\}$ and C=$\{$GC,TC$\}$. The dictionary becomes

```
gif.= tes.=: A+ or B+ or C+;
blo.=: BB+ or BF+;
=t: A−;
=ty: B– or BB−;
=ter: C−;
=fu: BF−;
```

The observed pair probabilities become:

p(A) = p(gif,t) + p(tes,t) = 2/11 = p(B) = p(gif,ty) + p(tes,ty)

p(C) = p(gif,ter) + p(tes,ter) = 3/11

p(BB) = p(blo,ty) = 3/11

p(BF) = p(blo,fu) = 1/11

So the observed entropy is now

$$h_{PAIR}^{red1.} = -\frac{4}{11}\log_2\frac{2}{11} - \frac{6}{11}\log_2\frac{3}{11} - \frac{1}{11}\log_2\frac{1}{11} = 2.231270$$

The generated entropy is $h^{gen} = \log_2 5 = 2.321928$. The difference is $h^{gen} - h^{obs} = 0.090658$. This clearly brings the entropy closer to the theoretical (equidistributional) maximum; the grouping goes. However, $h^{lang} = \log_2 8 = 3$ as before, since the generated language still has 8 words in it.

Case 2. Group gif.= and blo.= together. Let $\delta = \{\text{gif.=, blo.=}\}$. Then the link types GB and BB can be consolidated, because they share the common suffix =ty: B={GB,BB}. No other link consolidation is possible, without permitting impermissible (previously unseen) linkages. The dictionary becomes

```
gif.= blo.=: GA+ or B+ or GC+ or BF+;
tes.=: TA+ or TB+ or TC+;
=t: GA- or TA-;
=ty: B- or TB-;
=ter: GC- or TC-;
=fu: BF-;
```

Note that this dictionary does allow several previously unobserved words: giffu, blot, bloter. This is what happens when one hypothesizes unions between classes that merely overlap, instead of being subsets. What happens next depends on whether the overlap was large, or small.

The observed pair probabilities become:

p(GA) = p(gif,t) = 1/11 = p(GC) = p(TA) = p(TB)

p(TC) = p(tes,ter) = 2/11

p(B) = p(gif,ty) + p(blo,ty) = 4/11

p(BF) = p(blo,fu) = 1/11

So the observed entropy is now

$$h^{red2.}_{PAIR} = -\frac{5}{11}\log_2\frac{1}{11} - \frac{2}{11}\log_2\frac{2}{11} - \frac{4}{11}\log_2\frac{4}{11} = 2.550341$$

The generated entropy is $h^{gen} = \log_2 7 = 2.807355$. The difference is $h^{gen} - h^{obs} = 0.257014$. The entropy is not getting closer to the equidistributional maximum; this grammar is rejected.

Case 3. Group =t and =ty together. Let $\varepsilon = \{\text{=t, =ty}\}$ Then we may group G={GA, GB} and T={TA, TB}. The corresponding link-grammar dictionary is:

```
gif.=: G+ or GC+;
tes.=: T+ or TC+;
blo.=: BB+ or BF+;
=t =ty: G- or T- or BB-;
=ter: GC- or TC-;
=fu: BF-;
```

The above again allows a new, unobserved word: "blot". The observed pair probabilities become:

p(G) = p(gif,t) + p(gif,ty) = 2/11 = p(T) = p(tes,t) + p(tes,ty)

p(GC) = p(gif,ter) = 1/11

p(TC) = p(tes,ter) = 2/11

p(BB) = p(blo,ty) = 3/11

p(BF) = p(blo,fu) = 1/11

So the observed entropy is now

$$h_{PAIR}^{red3.} = -\frac{6}{11}\log_2\frac{2}{11} - \frac{2}{11}\log_2\frac{1}{11} - \frac{3}{11}\log_2\frac{3}{11} = 2.481715$$

The equidistributional entropy is $h^{gen} = \log_2 6 = 2.584963$. The difference is $h^{gen} - h^{obs} = 0.103248$. This difference means we are getting closer to the maximum; the grouping is acceptable! Its really not much worse than case 1, which was unambiguous.

Case 4. Group =ty and =fu together. Let $\zeta = \{$=ty, =fu$\}$. Then we must group B={BB,BF} together. The dictionary is:

```
gif.=: GA+ or GB+ or GC+;
tes.=: TA+ or TB+ or TC+;
blo.=: B+;
=t: GA− or TA−;
=ty =fu: GB− or TB− or B−;
=ter: GC− or TC−;
```

No new unobserved words are allowed by this grouping! The observed pair probabilities are:

p(GA) = p(gif,t) = 1/11 = p(GB) = p(GC) = p(TA) = p(TB)

p(TC) = p(tes,ter) = 2/11

p(B) = p(blo,ty) + p(blo,fu) = 4/11

The observed entropy is then:

$$h_{PAIR}^{red4.} = -\frac{5}{11}\log_2\frac{1}{11} - \frac{2}{11}\log_2\frac{2}{11} - \frac{4}{11}\log_2\frac{4}{11} = 2.550341$$

Curiously, this entropy is identical to the completely different case 2. The equidistributional entropy is $h^{gen} = \log_2 7 = 2.807355$ and the difference is thus $h^{gen} - h^{obs} =$

0.257014 which is sharply further away from the equidistributional maximum. Thus, this grouping is rejected. This is perhaps surprising ... First, this grammar did not generate any new unobserved words; thus, it is a faithful grammar. Also, it suceeds in reducing the total number of link-types, and thus is naively acceptable for that reason. However, the frequency distribution of th generated grammar move away from the observed frequency distribution, leading to the rejection. This begs a question: when and how might we annotate the grammar with frequency information?

Case 5. Group =ter and =fu together, so that $\eta = \{$=ter, =fu$\}$. It does not appear that any link types get consolidated! This is not much of a grouping, then ...

```
gif.=: GA+ or GB+ or GC+;
tes.=: TA+ or TB+ or TC+;
blo.=: BB+ or BF+;
=t: GA- or TA-;
=ty: GB- or TB- or BB-;
=ter =fu: GC- or TC- or BF-;
```

Many new, unobserved words are allowed: bloter, giffu, tesfu. The observed pair probabilities are:

p(GA) = p(gif,t) = 1/11 = p(GB) = p(GC) = p(TA) = p(TB)

p(TC) = p(tes,ter) = 2/11

p(BB) = p(blo,ty) = 3/11

p(BF) = p(blu,fu) = 1/11

The observed entropy is then:

$$h_{PAIR}^{red5.} = -\frac{6}{11}\log_2\frac{1}{11} - \frac{2}{11}\log_2\frac{2}{11} - \frac{3}{11}\log_2\frac{3}{11} = 2.845351$$

This is identical to the unreduced entropy: no surprise, because no link consolidation was performed. The equidistributional entropy is the same as well: $h^{gen} = \log_2 8 = 3$ since there are still 8 link types. The language entropy increased: there are now 11 possible words in the language, so $h^{lang} = \log_2 11 = 3.459432$. This is a very unsatisfying situation: the difference in entropies is no better or worse than the starting point, and so this seems like a reasonable sideways slide, and yet, this grammar allows a bunch of nonsense words to be generated. That seems wrong.

Summary Of the 5 cases, three are blocked (cases 2,4,5), and two are acceptable (1,3). Case 1 looks to be the best. Lets see what might happen next:

- Case 1a. Group =t and =ty

- Case 1b. Group =t and =ter
- Case 1c. Group =ty and =ter

Case 1a resembles Case 3 so we expect it to advance. Likewise for case 1c. Its reasonable to guess that case 1b will be the strongest. Lets try some of these.

Case 1a. Group =t and =ty. The link merges are T={A,B}; the resulting gramar is:

```
gif.= tes.=: T+ or C+;
blo.=: BB+ or BF+;
=t =ty: T- or BB-;
=ter: C-;
=fu: BF-;
```

This grammar allows a new unobserved word: “blot”. The observed pair probabilities become:

p(T) = p(gif,t) + p(tes,t) + p(gif,ty) + p(tes,ty) = 4/11

p(C) = p(gif,ter) + p(tes,ter) = 3/11

p(BB) = p(blo,ty) = 3/11

p(BF) = p(blo,fu) = 1/11

So the observed entropy is now

$$h^{red1.}_{PAIR} = -\frac{4}{11}\log_2\frac{4}{11} - \frac{6}{11}\log_2\frac{3}{11} - \frac{1}{11}\log_2\frac{1}{11} = 1.867634$$

The generated entropy is $h^{gen} = \log_2 4 = 2$. The difference is $h^{gen} - h^{obs} = 0.132366$ which is not closer than the previous delta of 0.090658, so this is rejected.

Casse 1b. Group =t and =ter. This consolidates links T={A,C} and so the dictionary becomes

```
gif.= tes.=: T+ or B+;
blo.=: BB+ or BF+;
=t =ter: T-;
=ty: B- or BB-;
=fu: BF-;
```

This does not generate any new words. The observed pair probabilities become:

p(A) = p(gif,t) + p(tes,t) + p(gif,ter) + p(tes,ter) = 5/11

p(B) = p(gif,ty) + p(tes,ty) = 2/11

p(BB) = p(blo,ty) = 3/11

p(BF) = p(blo,fu) = 1/11

So the observed entropy is now

$$h^{red1.}_{PAIR} = -\frac{5}{11}\log_2\frac{5}{11} - \frac{2}{11}\log_2\frac{2}{11} - \frac{3}{11}\log_2\frac{3}{11} - \frac{1}{11}\log_2\frac{1}{11} = 1.789929$$

The generated entropy is $h^{gen} = \log_2 4 = 2$. The difference is $h^{gen} - h^{obs} = 0.210071$ which does not get closer; the best still stands at 0.090658. This is surprising: it sems to be blocking the discovery of the clique.

Case 1c. Group =ty and =ter. This consolidates T={B,C}, so the dictionary becomes

```
 gif.= tes.=: A+ or T+;
 blo.=: BB+ or BF+;
=t: A−;
=ty =ter: T− or BB−;
=fu: BF−;
```

This does not generate any new words. The observed pair probabilities become:

p(A) = p(gif,t) + p(tes,t) = 2/11

p(C) = p(gif,ty) + p(tes,ty)+ p(gif,ter) + p(tes,ter) = 5/11

p(BB) = p(blo,ty) = 3/11

p(BF) = p(blo,fu) = 1/11

The changes are the same as for case 1b. Again, this is blocked.

Case 1f. This is the “final” case: group together =t =ty =ter into one. This consolidates T={A,B,C} so that

```
 gif.= tes.=: T+;
 blo.=: BB+ or BF+;
=t =ty =ter: T− or BB−;
=fu: BF−;
```

This allows new words “blot”, “bloter”. The observed pair probabilities become:

p(T) = p(gif,t) + p(tes,t) + p(gif,ty) + p(tes,ty)+ p(gif,ter) + p(tes,ter) = 7/11

p(BB) = p(blo,ty) = 3/11

p(BF) = p(blo,fu) = 1/11

The observed entropy is

$$h_{PAIR}^{red1.} = -\frac{7}{11}\log_2\frac{7}{11} - \frac{3}{11}\log_2\frac{3}{11} - \frac{1}{11}\log_2\frac{1}{11} = 1.240671$$

Hmm. 3 link types

Summary Movement to Cases 1a, 1b and 1c are all blocked. This seems surprising. The relatively high-frequency observation of =ter makes the distribution of the consolidated grammar to deviate strongly from the distribution of the observed corpus. This seems like an undesired effect, as the point of learning how to simplify the grammar is to obtain a smaller grammar, rather than to preserve the the distribution of the corpus. Mostly.

Intuition suggests that the grammar for "common" cases should be consolidated. The grammar for quite rare cases should indeed be handled distinctly. To avoid this seemingly perverse outcome, perhaps the grammar should contain frequency information, which is to be consolidated appropriately. This is truly tedious, but seems to be neccessary. So we have to start from scratch.

And we do, below, and its a total failure, as now, the corpus frequencies are recorded faithfully, so the consolidation process doesn't tell us anything we didn't know. Its the same calculation done differently.

Worked Link Consolidation Example, with Frequencies (XXX Fail)

(XXX The below fails, don't bother reading it). So we start all over again, using the same corpus frequences as before, namely, those of table 2. The grammar is essentially identical to that of 2, except that it is now annotated with probabilities.

```
gif.=: (GA+)(1/11) or (GB+)(1/11) or (GC+)(1/11);
tes.=: (TA+)(1/11) or (TB+)(1/11) or (TC+)(2/11);
blo.=: (BB+)(3/11) or (BF+)(1/11);
=t: GA- or TA-;
=ty: GB- or TB- or BB-;
=ter: GC- or TC-;
=fu: BF-;
```

The above only annotates the +-going links; it seems like annotating the –going links would cause double-counting. This is somewhat confusing, since the probability has nothing to do with directionality. A better notation is not obvious. Lets go through the cases as before.

Case 1. Group gif.= and tes.= together. Let $\gamma = \{$gif.=, tes.=$\}$. Then the link types G* and T* need to be consolidated: A=$\{$GA,TA$\}$ and likewise B=$\{$GB,TB$\}$ and C=$\{$GC,TC$\}$. The dictionary becomes

```
gif.= tes.=: (A+)(2/11) or (B+)(2/11) or (C+)(3/11);
blo.=: (BB+)(3/11) or (BF+)(1/11);
=t: A−;
=ty: B− or BB−;
=ter: C−;
=fu: BF−;
```

The observational probabilities are unchanged, as the dictionary probabilities have no bearing on the parsing. However, the entropy of the generated language is different, as it is no longer $\log_2 5$ but instead

$$h^{gen} = -\frac{6}{11}\log_2\frac{1}{11} - \frac{2}{11}\log_2\frac{2}{11} - \frac{3}{11}\log_2\frac{3}{11}$$

That is, it is now identical to $h^{observed}$. No surprise, as we made it like that, by encoding the frequency information in the dictionary.

Case 2. Group gif.= and blo.= together. Let $\delta = \{\text{gif.=, blo.=}\}$. Then the link types GB and BB can be consolidated, because they share the common suffix =ty: B={GB,BB}. No other link consolidation is possible, without permitting impermissible (previously unseen) linkages. The dictionary becomes

```
gif.= blo.=: (GA+)(1/11) or (B+)(4/11) or (GC+)(1/11) or (BF+)(1/11);
tes.=: (TA+)(1/11) or (TB+)(1/11) or (TC+)(2/11);
=t: GA− or TA−;
=ty: B− or TB−;
=ter: GC− or TC−;
=fu: BF−;
```

The generated entropy is

$$h^{gen} = -\frac{5}{11}\log_2\frac{1}{11} - \frac{4}{11}\log_2\frac{4}{11} - \frac{2}{11}\log_2\frac{2}{11}$$

which is identical to the corpus entropy, again. Not surprising, I guess ... we seem to be doing the same calculation, but in a different way. Dohh. Never mind ...

Alternate Distributions

Instead of looking for an equi-distribution, how about a Zipf distribution, which seems far more plausible? The distribution is

$$p(k,n) = \frac{1}{kH_n}$$

where the normalization is $H_n = \sum_{k=1}^{n} 1/n$. The entropy is then

$$\begin{aligned}
h_n^{Zipf} &= -\sum_{k=1}^{n} p(k,n) \log_2 p(k,n) \\
&= \frac{1}{H_n} \sum_{k=1}^{n} \frac{\log_2 kH_n}{k} \\
&= \log_2 H_n + \frac{1}{H_n} \sum_{k=1}^{n} \frac{\log_2 k}{k}
\end{aligned}$$

and the first few values are shown below. For comparison, $h_n^{equi} = \log_2 n$ is also shown.

n	H_n	h_n^{Zipf}	h_n^{equi}
2	1.5	0.918296	1
3	1.83333	1.435371	1.584963
4	2.033333	1.792488	2
5	2.283333	2.063860	2.321928
6	2.45	2.281979	2.584963
7	2.592857	2.463914	2.807355
8	2.717857	2.619715	3

The question is then: how would the above cases go if this was used as the deciding factor? This is shown below:

Case	# lnk	$h^{observed}$	h^{equi}	$h^{eq} - h^{obs}$	OK	h^{Zipf}	$h^{Zipf} - h^{obs}$	C1	C2
Base	8	2.845351	3	0.154649		2.619715	-0.225636		
1.	5	2.231270	2.321928	0.090658	Y	2.063860	-0.16741	Y	N
2.	7	2.550341	2.807355	0.257014	N	2.463914	-0.086427	Y	N
3.	6	2.481715	2.584963	0.103248	Y	2.281979	-0.199736	Y	N
4.	7	2.550341	2.807355	0.257014	N	2.463914	-0.086427	Y	N
5.	8	2.845351	3	0.154649		2.619715	-0.225636		
1a.	4	1.867634	2	0.132366	N	1.792488	-0.075146	Y	N
1b.	4	1.789929	2	0.210071	N	1.792488	0.002559	Y	N
1c.	4	1.789929	2	0.210071	N	1.792488	0.002559	Y	N
1f.	3	1.240671	1.584963	0.344292	N	1.435371	0.1947	N	N

There seem to be two different decision criteria to apply:

1. Does the reduced entropy come closer to the Zipfian entropy?
2. Does the reduced entropy increase, relative to the Zipfian entropy?

The first is shown in column C1, the second in C2. Naively, C2 seems like a better chooser. Does it also work for the simple case (with the original 7-word corpus)? Lets see:

Case	#lnk	$h^{observed}$	h^{equi}	$h^{eq} - h^{obs}$	OK	h^{Zipf}	$h^{Zipf} - h^{obs}$	C1	C2
Base	6	2.521641	2.584963	0.063322		2.281979	-0.239662		
1.	4	1.842371	2	0.157629	N	1.792488	-0.049883	Y	N
1a	2	0.985228	1	0.014772		0.918296	-0.066932	N	Y
1b.	2	0.863121	1	0.136879		0.918296	0.055175	N	N

Basically, this is really irritating.

Thoughts

What have we learned from the above?

- The problem of condensing together morphemes into classes which share common link types is the bipartite clique problem. It is a known-hard problem.
- Bad grammars increase the size of the language. This could be acceptable, if the increase is small. What's the criteria? Unclear.

Consciousness - 27 July 2014

Two works:

- Masafumi Oizumi, Larissa Albantakis, Giulio Tononi, "From the Phenomenology to the Mechanisms of Consciousness: Integrated Information Theory 3.0" (2014) PLOS Computational Biology, http://www.ploscompbiol.org/article/info%3Adoi%2F10.1371%2Fjournal.p
- Max Tegmark, Consciousness as a State of Matter (27 Feb 2014) arXiv:1401.1219v2 [quant-ph]

Curious points and thoughts:

- CEI – "Cause-effect information" – Tononi – sound like a time-ordered variant of mutual information. How should this be defined? Answer: my guess is that its just like the mutual information defined in eqn5, right? Because the relational complexity can deal with arbitrary structures, so that seems appropriate.

13 Sept 2014

The Zipfian distribution is typical of a scale-free network. So why is language scale-free? Crudely, because we attempt to recycle existing concepts/words.

Next, from this:

- Christoph Adami "Information-Theoretic Considerations Concerning the Origin of Life" http://arxiv.org/pdf/1409.0590v1.pdf

Come the following thoughts:

- Never assume a uniform distribution of parts. This makes it very unlikely that an imprtant assemblage of parts can arise at random. For Adami, this is used to argue that biotic and abiotic strings should have very siilar distributions (or rather, the converse: a non-uniform abotic distribution makes it much more likely to find a replictor with a similar distribution.)

- The information content of (grammatical sentences of length L is

  $$I_{grammatical} = -\log_2(N_{grammatical}/N_{total})$$

  where N_{total} is the total number of sentences of length L, assuming a *uniform* distribution of words picked from a vocabulary of D words. That is, $N_{total} = D^L$. But this is weird ... because the vocabulary isn't really a constant, and the natural distribution is not uniform, so its not clear what kind of "information" the above actualy is ...

Thermodynamics - 24 March 2015

Some quick short notes: blog post: "Thermodynamics with Continuous Information Flow" https://johncarlosbaez.wordpress.com/2015/03/21/19395 with arxiv paper: http://arxiv.org/pdf/1402.3276v3.pdf Jordan M. Horowitz and Massimiliano Esposito study the master equation for a probability p(x,y) over two distributions X,Y, which are connected via a bipartite graph. The total system is also connected to a thermal bath. The eqn is

$$\frac{dp(x,y)}{dt} = \sum_{x',y'} H_{x,x'}^{y,y'} p(x',y') - H_{x',x}^{y',y} p(x,y)$$

i.e. its Markovian; we've written two indexes, which makes it clearer when H is bipartite i.e.

$$H_{x,x'}^{y,y'} = \begin{cases} H_{x,x'}^{y} & \text{if } y = y' \\ H_{x}^{y,y'} & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases}$$

The interesting part is the entropy, and the thermal bath, which is not in the master eqn(!) The total entropy is $S_{tot} = S_{XY} + S_{env}$. Per usual, the information entropy is $S_{XY} = -\sum_{x,y} p(x,y) \log p(x,y)$. Two tricks now happen: (1) taking the timer derivative of S_{XY} results in something that naturally splits into an X piece and a Y piece. Trick (2) is that S_{env} cannot be written down directly, but its time derivative can be; it is proportional to the heat current: $\dot{S}_{env} = -\dot{Q}/T$ Observer the tiny dot over S,Q these are the usuaul rate-of-change dots, (i.e. just rates, not functions we are taking time derivative of). Q is heat, Q-dot is heat flow, T is temp. Local detailed balance requires that

$$\log \frac{H_{x,x'}^{y,y'}}{H_{x',x}^{y',y}} = \frac{-(E_{x,y} - E_{x',y'})}{kT}$$

is the change in energy due to a state transition: the change in energy is supplied by the heat reservior. Where does this mystery equation come from? Answer:

Detailed balance requires that, when the system reaches equilibirum, that the transition rate into and out of the equlibrium state $p_i = \pi_i$ are equal:

$$H_{ji}\pi_i = H_{ij}\pi_j$$

(there is NO repeated-index summation). Then, just write $\pi_i = \exp -E_i/kT$, and turn the crank. The general principle: *the log of the ratio of the forward and backward transition rates between two states must be proportional to the energy difference between those states!*

BTW the detailed-balance equation resembles Bayes Theorem, in that, if we wrote $H_{ji} \to P(j|i)$ and $\pi_i \to P(i)$, then detailed balance is written as $P(j|i)P(i) = P(i|j)P(j)$. So the master equation describes "non-equilibrium Bayes statistics", in a strange sense. Hmm. But, of course, this is just a Markov chanin/process.

26 March 2015

The Inverse Relationship Principle of Channel theory: "*Whenever there is an increase in available information there is a corresponding decrease in possibilities, and vice versa.*" Barwise, "Information and Impossibilities." Notre Dame J. Formal Logic Volume 38, Number 4, 488-515. Barwise, Jon and Jerry Seligman 1997. "Information Flow: The Logic of Distributed Systems". Cambridge: Cambridge University Press

Linear networks - 3 May 2015

Another Baez post: "*A Compositional Framework for Passive Linear Networks*" blog: https://johncarlosbaez.wordpress.com/2015/04/28/a-compositional-framework-for-passive-linear-networks/

So first, we have the table:

	mechanics	electronics	information geometry	geometric mechanics
q	position	charge	entropy	point on manifold
$\dot{q}$	velocity	current	entropy change	tangent vector
p	momentum	flux linkage	temperature momentum	covector (vector in cotangent bundle)
$\dot{p}$	force	voltage	temperature	map from tangent bundle to cotangent bundle
	principle of least action	principle of least power dissipation	?	principle of least action

This table is slightly oversimplified; the first four columns show only the linear case. The fifth column makes clear that force isn't really p-dot; that only holds when the manifold is flat. Anyway..

Key concepts: (*) monoidal categories are needed, and (*) symplectic geometry is needed.

Baez does the linear passive-component electronics example, viz a network of passive resistors, capacitors, inductors. For the resistor network, voltages at each node are taken from the field $\mathbb{F} = \mathbb{R}$ while for the inductive network, the field is the field of rational functions of one variable $\mathbb{F} = \mathbb{R}(t)$ with t time: i.e. voltage varying over time. A Dirichlet form is a quadratic form

$$P(\phi) = \frac{1}{2} \sum_{i,j} \frac{(\phi_i - \phi_j)^2}{r_{ij}}$$

where r_{ij} is the reistance (impedance) between nodes i and j, and $\phi_i = \phi(i)$ is the voltage at node i. (Actually, we should be summing over edges, so as to handle parallel resistors). Note that the space of Dirichlet forms is smaller than the space of quadratic forms: Dirichlet forms do not have diagonal entries. Note that P is (half) the power dissipation.

The principle of least power dissipation is this: Given fixed voltages ψ on the boundary of the network, i.e. on the input/output terminals, the actual power dissipated is

$$Q(\psi) = \min_{\phi \in \mathbb{R}^N, \phi|_{\partial N} = \psi} P(\phi)$$

Notation: there are N nodes, so voltages live in $\mathbb{R}^N$. The boundary of the network (input/output terminals) is written as ∂N and the voltages are held fixed at the boundary. Note that Q is also a Dirichlet form. Viz its a map $Q : \mathbb{R}^{\partial N} \to \mathbb{R}$. The black-box principle of equvalent resistor networks is that any two resistor networks are black-box equivalent when they have the same Q.

For the correct generalization to impedance, it is not enough to just replace $\mathbb{F} = \mathbb{R}$ by $\mathbb{F} = \mathbb{R}(t)$ because this fails to deal with the time variation correctly. Put it another way: for the pure resistor network, we are free to fix voltages at both the input and output terminals arbitrarily; the internal currents are determined entirely by these. For the general case with impedance, we are not free to fix both voltages and currents at both the input and output terminals. Out of the total set of $2\dim(\partial N)$ voltages and currents, we can fix only half the set, i.e. a mixture of voltages, currents of $\dim(\partial N)$.

To handle this, we need to construct a symplectic vector space, with a symplectic form on it, and work in the Lagrangian subspace of it. Thus, we have $\psi \in \mathbb{F}^{\partial N}$ as the potentials at the network terminals, and $dQ_\psi \in \left(\mathbb{F}^{\partial N}\right)^*$ as the conjugate currents. Out of the total space $\mathbb{F}^{\partial N} \oplus \left(\mathbb{F}^{\partial N}\right)^*$ of states, the subspace of actually attainable states is

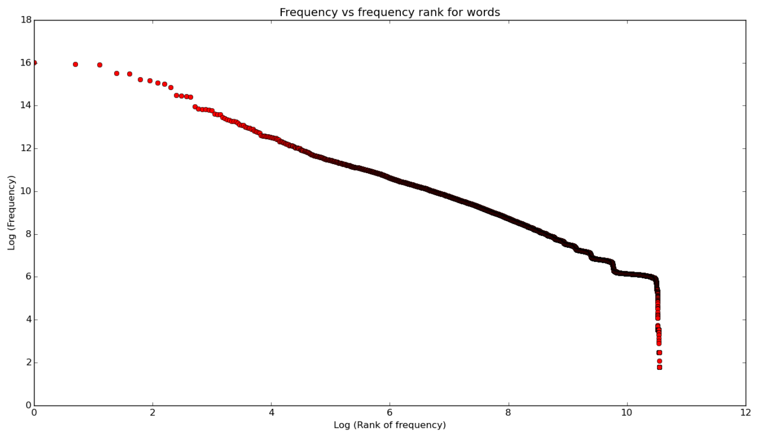
$$\text{Graph}\,(dQ) = \left\{ \left(\psi, dQ_\psi\right) | \psi \in \mathbb{F}^{\partial N} \right\} \subseteq \mathbb{F}^{\partial N} \oplus \left(\mathbb{F}^{\partial N}\right)^*$$

The set of Lagrangian subspaces is an algebraic variety, the Lagrangian Grassmanian.

Baez primary result on impedence networks is that the black box is describable by the symplecitification of .. OK I don't get it.

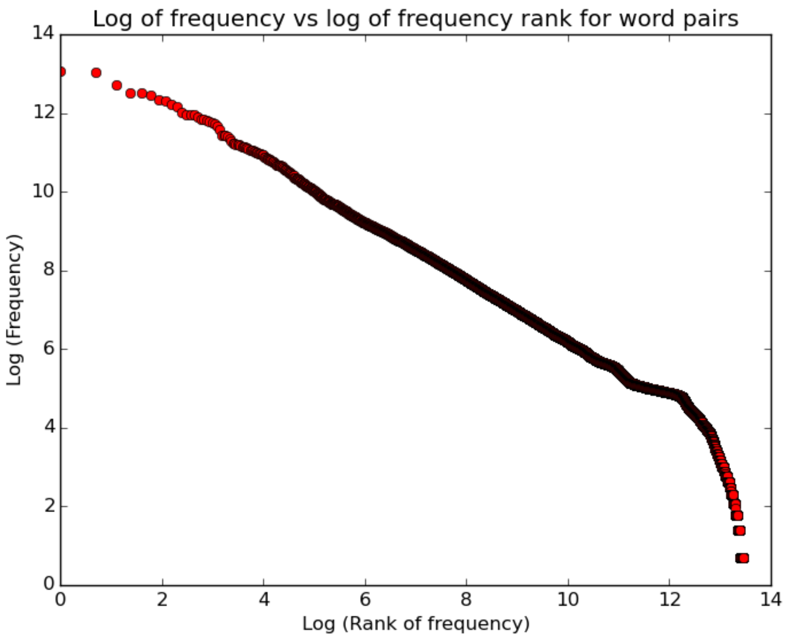
MI graphs – 31 May 2015

Results from Rohit:

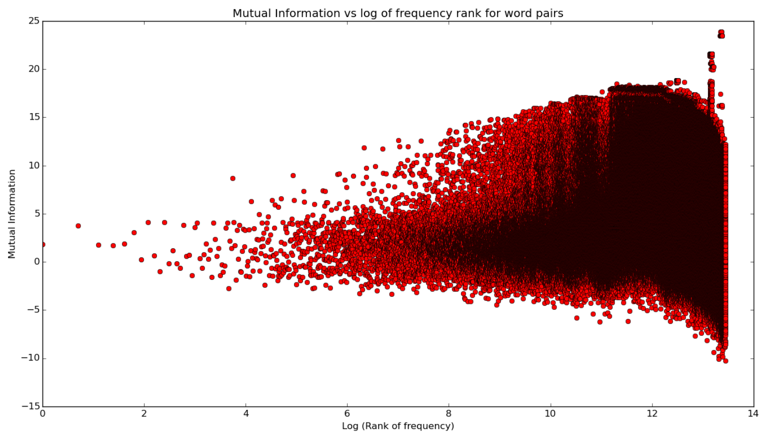


Above is for single words. Frequency is the number of times the word was observed.

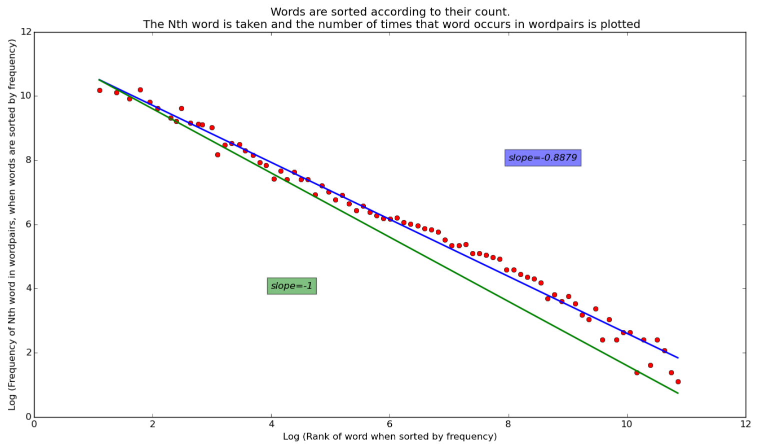
The log is the natural log. Below is for word-pairs.



Below is a scatterplot for mutual information of word-paris vs rank.



number of word pairs as function of word rank



Summary: these are more or less exactly as expected. Will need to make cuts to get rid of the low-frequency word pairs...

Mining Grammatical Categories – 20 June 2015

Now that we have a database filled with disjunct statistics, how do we datamine that for grammatical categories, which is, after all, the main point of this exercise? Let me explain in several steps; at first illustrative, and then, more precisely. So first, consider a corpus containing these sentences:

```
the big tree
a green tree
the big bush
```

```
a green bush
```

I want to conclude that "tree" is a lot like "bush", and the two should be considered as being "similar enough to be merged into a common grammatical category". That is, the words "tree" and "bush" always occur in similar contexts, or even the same contexts. The word “context” here means “the dependency parse context”, and not “the n-gram context”. More precisely, it means “the accumulated statistics for the disjuncts obtained from MST dependency parses”.

Suppose the following parses were observed:

```
     +---MA---+
     |    +-MB-+
     |    |    |
    the  big  tree

     +---MC---+
     |    +-MD-+
     |    |    |
     a  green tree

     +---ME---+
     |    +-MF-+
     |    |    |
    the  big  bush

     +---MG---+
     |    +-MH-+
     |    |    |
     a  green bush
```

Recall that the above parses were obtained by performing a Maximum-Spanning-Tree (MST) parse based on word-pair mutual information (MI). The MST is obtained by considering the graph clique joining all words in the sentence, and then keeping only those edges that have the greatest MI between pairs of words. This is the “Yuret parse”. The Yuret parse does not have labelled edges, and so we assign arbitrary (but unique!) link labels to the edges that were kept. Every unique word pair gets a unique link type. Then, using the standard Link Grammar theory, each link is broken into a + and a - connector, and the ordered set of connectors on a word are called a disjunct.

The disjuncts extracted from the above parses would then be:

```
    tree:  (MA- & MB-)  or  (MC- & MD-)
    bush:  (ME- & MF-)  or  (MG- & MH-)
```

No two disjuncts are alike, so naively, these seem completely uncomparable. Of course, this is wrong; we need to compare the “decoded disjunct”. The “decoded disjunct” is NOT a part of the standard Link Grammar theory, so let me explain it here: it is simply the disjunct where the connector is replaced by the word or word-class that it connects to. For example, `MA-` connects to the word “the”, so the “decoded connector” for `MA-` is `$the$-`. So, the decoded disjuncts are then:

```
tree: ($the$- & $big$-) or ($a$- & $green$-)
bush: ($the$- & $big$-) or ($a$- & $green$-)
```

Now we can see that the decoded disjuncts are identical, for this example. Based on this, we conclude that perhaps “tree” and “bush” indeed belong to the same grammatical category. The remainder of the clustering algorithm is now “obvious”: rewrite the dictionary so that it has a single entry for both words:

```
tree bush: (MA- & MB-) or (MC- & MD-)
```

This leaves the ME+, MF+, *etc.* connector dangling: thus, we need to search for all occurances of ME+ and replace it by MA+, and likewise all occurances of MF+ need to be replaced by MB+, and so on.

Similarity metrics

The above conveys the general idea, but is over-simplifies a few aspects. First of all, it is very unlikely that two words will appear in sentence contexts that are exactly identical. Secondly, some constructions may be very common, and others, very rare; that is, some disjuncts may be very common, and some very rare. So, for example: suppose we read a text which used the phrase “*the big idea*” a lot, but we also read an obscure linguistics text that said that “*a green idea sleeps furiously*”. It would probably be a mistake to lump “idea” in with “tree, bush”, given that “green idea” is a very rare construction. Thus, we need a better way of comparing collections of disjuncts.

One obvious way is to treat a collection of decoded disjuncts as a vector in a high-dimensional vector space. The similarity between two vectors could be given by the cosine between two vectors. Alternately, perhaps the vectors could be treated as points, and similarity be given by the distance between points. There are other possibilities; the best choice is not obvious; several need to be explored.

Thus, for example, let $\{e_1, e_2, e_3, \cdots\}$ be the basis of a high-dimensional vector space. For the previous example, we let e_1 correspond to the decoded disjunct `($the$- & $big$-)` while e_2 corresponds to `($a$- & $green$-)`. The word “tree” is then some vector ... what vector should it be? There are several choices. Suppose that `($the$- & $big$-)` was observed with a frequency p_1 and that `($a$- & $green$-)` was observed with frequncy p_2. The corresponding vector is then obviously $p_1e_1 + p_2e_2$ and we can construct another vector that corresponds to the the word “bush”, say, for example: $q_1e_1 + q_2e_2$.

The dot-product between “tree” and “bush” is then given by $p_1q_1 + p_2q_2$, so that the larger the product, the closer the two words are. The cosine angle is $(p_1q_1 + p_2q_2)/|p||q|$ where $|p| = \sqrt{p_1^2 + p_2^2}$ and so on. The closer that the cosine is to 1.0, the closer the two words are. There are other possibilities: we have the Cartesian distance

$$dist(tree, bush) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

and we can contemplate *lp*-metrics as well.

None of the above metrics take into account the mutual information (MI) of the disjunct. This is almost surely a mistake. Due to the vagaries of MST parsing, there will be many disjuncts with a low MI value. This is not uncommon in sentences with prepositions, where MST gives some poor choices for the links to the prepositions, and thus results in disjuncts with low MI values. Recall, the higher the MI, the stronger the structure is. Thus, perhaps a better vector for "tree" might be

$$tree = e_1 m_1 p_1 + e_2 m_2 p_2$$

The above seems to be the most entropic-like in its expression. However, the probabilites might weight the terms too strongly, and so a weaker weighting would be the below. It is not yet clear to me which of these expressions are the most "elegant", or which work the best...

$$tree = e_1 (m_1 - \log_2 p_1) + e_2 (m_2 - \log_2 p_2)$$

Here m_1 and m_2 are the mutual information of the disjuncts `(MA- & MB-)` and `(MC- & MD-)`, respectively. The last two seem to be closer to the intended spirit of the maximum entropy principle. There are even more possibilities, though.

Frequency and Mutual Information

The above section makes explicit use of the frequency and the mutual information of a disjunct. It is useful to define these. Given a disjunct `(MA- & MB-)` let N`(MA- & MB-)` be the number of times that this disjunct has been observed. It will usually be an integer (except when obtained in certain unusual situations not discussed here). Let N`(*- & *-)` be the numer of times that any two-connector disjunct has been observed, as long as both connectors point in the - direction. That is,

$$N(*-\&*-) = \sum_{c_1 \in -, c_2 \in -} N(c_1 \& c_2)$$

the summation taking place over all connectors in the - direction. The frequency of observing `(MA- & MB-)` is then

$$p(MA-\&MB-) = \frac{N(MA-\&MB-)}{N(*-\&*-)}$$

The mutual information associated with the disjunct is then

$$m(MA-\&MB-) = -\log_2 \frac{p(MA-\&MB-)}{p(MA-\&*-)p(*-\&MB-)}$$

The reason for this possibly unexpected form was developed earlier in this diary.

Semantics

There is another interesting issue that arises in the above discussion: the problem of syntax-semantics correspondance. Consider, for example, the sentence *"the dog treed* *the squirrel*". Here the word "tree" is used as a verb, meaning "the dog chased the squirrel up into the tree". Such sentences will cause the the word "tree" to accumulate disjuncts that the word "bush" will not have. Likewise, "*I'm bushed*" is a verb usage that has no analogous "tree" version. Thus, not only do the words "bush" and "tree" have different sets of disjuncts, but the differences are hiding semantic differences ...

There are several strategies that can be used to deal with this. More on this later.

Finding word pairs

We need a good way of finding word-pairs that are likely to be related. I think that perhaps the pattern matcher may be ideal for this. Details are TBD... but the basic idea is that the hypergraph for "tree: (MA- & MB-)" is connected to "big" because MB- is connected to "big", and "big" is connected to other lg-connectors, which in turn are connected to other disjuncts, which are then connected to other words. Thus, we search the local neighborhood of "tree", which causes us to dsicover the word "big", and then we search the neighborhood of "big" to find candidates such as "bush" which might be comparable to "tree". This search graph is not small, but it is not large: There may be thousands of words that are two hops away from "tree", but not millions.

Putting it all together

These are the things that need to be done:

1. compute the MI for the disjuncts
2. pick a common noun, compute the similarity scores for that word and every word that is linked to it. created ranked graph of similarity.
3. repeat step 2 for several different similarity formulas
4. repeat steps 2,3 for several verbs, several adjectives, several adverbs, several determiners, several prepositions.
5. Write code for creating grouping words into grammatical clusters.
6. Pick the most promising metric, and start clustering in bulk.

Step 5 requires writing a lot of code; it can all be written before the final metric has been determined.

The end.

That's all for now. More later.

Not LSA – 1 July 2015

NotLSA – a way to do LSA-like things without actually using LSA (Latent Semantic Analysis). Two very low-brow approaches, maybe well-known in the industry; I have no idea. Both of these approaches attempt to automatically extract keywords from documents. What cool about this is that its ... unsupervised; requires no training, and is based on very simple, proven ideas. Obvious, even: compute the mutual information between pairs of things ... between words and documents, between words and word-pairs, etc. Heh.

But how do we do this? How do we compute the MI between a page of text, and a word? No way to answer this without diving into the details.

Text-keyword correlation

Lets take a text, say – 1000 pages of .. something. Some corpus. We want to compute the mutual information between the page itself, and the words on the page. We do this by analogy to MI of word pairs.

Call the k'th page g_k. Count the number of times that word w_m appears on this page; let this count be N_{mk}. Define $N_m = \sum_k N_{mk}$ be the total number of times that the work w_m appear in the document, and let $N = \sum_m N_m$ be the total number of words in the document. Then, as usual, define probabilities, so that

$$p_m = P(w_m) = N_m/N$$

is the frequency of observing word w_m in the entire corpus, and

$$p_{mk} = P(w_m|g_k) = N_{mk}/\sum_m N_{mk}$$

be the (relative) frequency of the same word on page g_k. Notice that the definition of p_{mk} is independent of the page size. Pages do not all have to be of the same size. Define the mutual information as

$$\text{MI}(g_k, w_m) = -\log_2 \frac{p_{mk}}{p_m} = -\log_2 \frac{N_{mk}N}{\sum_m N_{mk} \sum_k N_{mk}} = -\log_2 \frac{p(m,k)}{p(m,*)p(*,k)}$$

This is essentially a measure of how much more often the word w_m appears on page g_k as compared to its usual frequency. The highest-MI words are essentially the topic words for the page. The right-most form introduces a new notation, to make it clear that it resembles the traditional pair-MI expression. The notation is

$$p(m,k) = \frac{N_{mk}}{N}$$

so that

$$p(m,*) = \sum_k p(m,k) \qquad \text{and} \qquad p(*,k) = \sum_m p(m,k)$$

are the traditional-looking pair-MI values.

TODO: – this does not have the feature-reduction/word-combing aspects of LSA...

Variants

Instead of working with words, we could work with word-pairs, which is a stand-in for working with (named) entities. Thus, we can identify if a named entity occurs in a document more often than average.

Unsupervised Morphology Learning References

Here's some:

- John Goldsmith, "The unsupervised learning of natural language morphology", Journal Computational Linguistics archive Volume 27 Issue 2, June 2001 Pages 153-198 MIT Press http://delivery.acm.org/10.1145/980000/972668/p153-goldsmith.pdf
- John Goldsmith, "An algorithm for unsupervised learning of morphology" Natural Language Engineering Volume 12 / Issue 04 / December 2006, pp 353-371 Cambridge University Press DOI: http://dx.doi.org/10.1017/S1351324905004055 http://people.cs.uchicago.edu/~jagoldsm/Papers/algorithm.pdf
- Survey Article Unsupervised Learning of Morphology Harald Hammarström Lars Borin http://www.mitpressjournals.org/doi/pdf/10.1162/COLI_a_00050

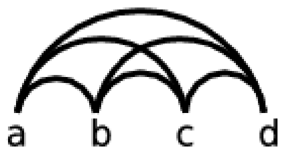
Predicate-Argument structure

Here's one:

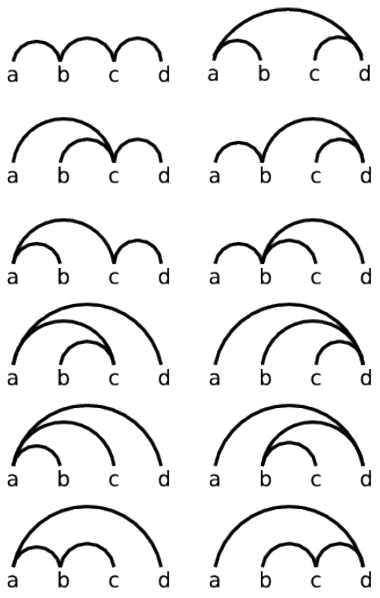
- The Darwinian evolution of natural language comes from a combination of Expressive FSM's and Lexical predicate-argument FSM's within the human brain. Shigeru Miyagawa, Robert C. Berwick and Kazuo Okanoya "The emergence of hierarchical structure in human language" Front. Psychol., 20 February 2013 | http://dx.doi.org/10.3389/fpsyg.2013.00071 http://alpha-leonis.lids.mit.edu/wordpress/wp-content/uploads/2014/01/shigeru-berwick-kaz-frontiers13.pdf

Edge-counting 27 March 2017

Counting edges in a clique is not the same as counting edges in planar trees. The diagram below shows the clique of a four-word sentence. The "words" are 'a', 'b', 'c' and 'd'. There are a total of six edges, with one edge between every possible word-pair. Each edge occurs only once.



Pair counting in planar diagrams gives different results. The diagram below shows the twelve planar trees, containing no cycles, that can be formed by parsing a sentence of four words.



There are 36 edges grand total, and these are unequally distributed. The counts are:

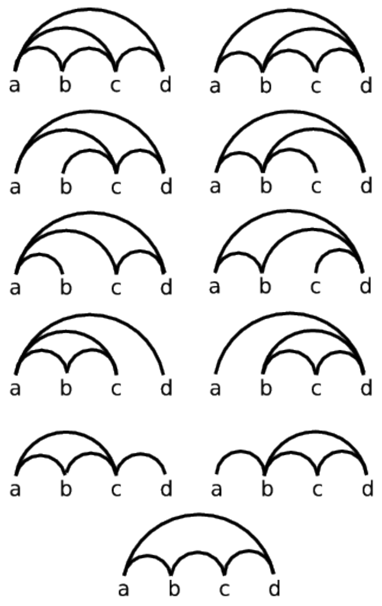
word-pair	count
(ab)	7
(bc)	7
(cd)	7
(ac)	4
(bd)	4
(ad)	7

Note that the most frequent edges occur almost twice as often as the least-frequent edges. The distribution, by length, is:

Length	Count
1	21
2	8
3	7

Note that the progressively-longer edges get less frequent.

If graphs with cycles are also allowed, (but no edge crossings) then, in addition to the above, there are eleven more diagrams. These are shown below.



Again, we count the number of edges, as before. The 'tree' column shows he counts from before; the loop count count the edges from the additional eleven diagrams; the total is just that.

word-pair	tree	loop	total
(ab)	7	9	16
(bc)	7	9	16
(cd)	7	9	16
(ac)	4	5	9
(bd)	4	5	9
(ad)	7	9	16

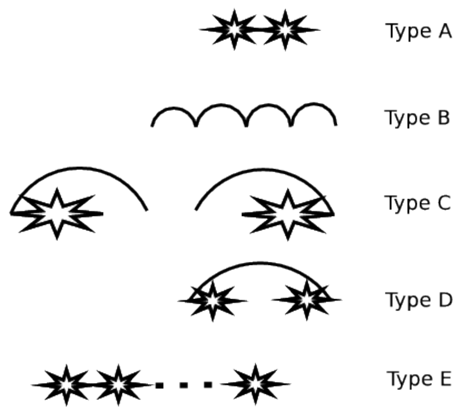
Likewise, the number of arcs of the given length is now given below:

Length	Count
1	48
2	18
3	16

What are the actual distributions, for these two cases? Bewgine by counting the number of planar trees.

Counting planar tree graphs

Let's count the number of planar tree graphs; i.e. those without any loops. First, we need a generic formula for sentences of length N. This is not so very easy. The diagram below shows one way to count. (I think what follows is correct, but I might be making a mistake. I am unaware of any literature that presents this information).



Here, the star represents some planar tree connecting all of the words of a smaller sentence. Assume that there are $T(n)$ such trees, connecting n words. Tree diagrams of of Type A are assembled by placing two adjacent smaller trees next to each other. Naively, one can then count how many such pairs there are; the issue is that the Type B diagram will occur mutiple times in this pairing; we would rather NOT count it with this mutiplicity. To avoid this problem, we should only allow pairs, such as Type A, to be assembled of sub-parts of the shapes C and D. Because of the over-arching arc, these can never result in double-counting. However, counting only pairs results in an under-counting: graphs of Type B never occur. Thus, one should count pairs, triples, and so on – graphs of Type E. Now we have a way of getting the formula. Define $D(n)$ as the count of the number of planar trees, connecting n words, having an arc connecting the first and last word: i.e. trees of type C or D. (Think "D = dome") One then has that

$$D(n) = \sum_{j=1}^{n-1} T(j)T(n-j)$$

It is convenient, here, to define $T(1) = 1$. The first and last terms of this sum then correspond to trees of Type C, while the middle terms are trees of type D.

To count trees of Type E, is is convenient to break this up into the problem of counting chains of length k, so that there are $C_k(n)$ trees, consisting of a sequence of k domes, making up a total of n words. One then has that

$$T(n) = D(n) + C_2(n) + \cdots + C_{n-1}(n)$$

It's convenient, here, to define $C_1(n) = D(n)$. Writing down the $C_k(n)$'s requires some combinatorial magic. The first one is

$$C_2(n) = \sum_{j=2}^{n-1} D(j)D(n-j+1)$$

Next comes

$$C_3(n) = \sum_{j=2}^{n-1} \sum_{m=2}^{n-j+1} D(j)D(m)D(n-j-m+2)$$

which is awkward to write down. It's easier to count partitions of sets. Thus, what really is happening here is that the sums range over all k-way partitions of sets containing $n+k-1$ elements. Not the partition is NOT over n elements: to get connected graphs, we have to identify end-points of each link in the chain. Thus,

$$C_k(n) = \Pi_\sigma \cdots$$

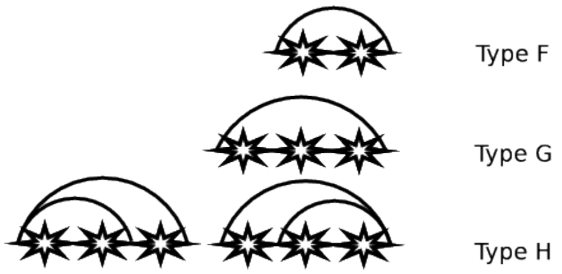
The table below summarizes the first few sums:

n	$T(n)$	$D(n)$	$C_2(n)$	$C_3(n)$	$C_4(n)$	$C_5(n)$
1	1	1				
2	1	1				
3	3	2	1			
4	12	7	4	1		
5	45	20	18	6	1	
6		123				1

Either I am computing this wrong, or the sequences are not in OEIS. Surprising!

Counting planar loop graphs

The above process can be repeated, except that this time, we consider the planar graphs containing loops. To get started, consider the diagram below.



Here, the stars represent either "domed" diagrams, or the empty set (a set containing no words and no edges. The type F concatenates two domes, and puts a dome over those, in turn. Since both of the stars are domed (or empty), it is impossible to add any additional edges to this graph. So, for graphs constructed out of a pair domes (one or both possibly empty), Type F is all that there is. For three domes in a row, there are only three ways of adding edges: these are shown in type G and H in this diagram. Again, this exhauses all possibilities. This process constructs both looped and tree diagrams. The general idea is to repeat this, for sequences of four or more stars.

The counting is similar to that before. Let $F(n)$ count the number of domed graphs, connecting n words. Let $G_2(n)$ count the number of type F graphs, made of two parts, and containing n words. Consulting the diagram, we have

$$G_2(n) = F(n-1) + \sum_{k=2}^{n-1} F(k)F(n-k+1) + F(n-1)$$

Likewise, let $G_3(n)$ count the number of graphs of type G and H, combined. Consulting the diagram, this has a more complex expression:

$$G_3(n) = \sum_{k=2}^{n-2} F(k)F(n-k) + \sum\sum F()F()F()... + \sum_{k=2}^{n-2} F(k)F(n-k) + ...$$

The total number of domed graphs having n words is then

$$F(n) = \sum_{k=2}^{n-1} G_k(n)$$

Let $S(n)$ be the count of a string of domed graphs, but NOT having connecting arcs: that is, graphs of type A or E.

A table of these is given below.

n	$L(n)$	$S(n)$	$F(n)$	$G_2(n)$	$G_3(n)$	$G_4(n)$	$G_5(n)$
1	1	0	1				
2	1	0	1				
3	4	1	3	3			
4	23				x		
5	156					x	
6	1162						x

TODO

Explain how mutual exclusion of concepts as performed by humans when learning new concepts, resembles optimal strategies for the channel coding theorem, by minimizing confusion between similar concepts.

The End

References

[And12] Steven R. Anderson. Dimensions of morphological complexity. In Greville G. Corbett Matthew Baerman, Dunstan Brown, editor, *Understanding and measuring morphological complexity*, 2012.

[iC06] R. Ferrer i Cancho. Why do syntactic links not cross? *EPL (Europhysics Letters)*, 76(6):1228–1234, 2006.

[Liu08] Haitao Liu. Dependency distance as a metric of language comprehension difficulty. *Journal of Cognitive Science*, 9(2):159–191, 2008.

[Yur98] Deniz Yuret. *Discovery of Linguistic Relations Using Lexical Attraction*. PhD thesis, MIT, 1998.