Language Learning Diary - Part Four

Linas Vepstas

Sept 2021 - Dec 2021

Abstract

The language-learning effort involves research and software development to implement the ideas concerning unsupervised learning of grammar, syntax and semantics from corpora. This document contains supplementary notes and a loosely-organized semi-chronological diary of results. The notes here might not always makes sense; they are a short-hand for my own benefit, rather than aimed at you, dear reader!

Introduction

Part Four of the diary on the language-learning effort continues work on the English dataset.

Summary Conclusions

A summary of what is found in this part of the diary:

- Connector sequences or "disjuncts" are the "jigsaw pieces" extracted from maximum planar spanning-graph (MPG) parses of English sentences. How many connectors are there on a typical jigsaw piece? Answer: 2.25 on average. These are closely modeled by a log-normal distribution. This is good news for English, where we expect nouns to typically have 1, 2 or 3 connectors, and verbs and prepositions to have 2,3,4. Adjectives, adverbs and determiners typically have only 1 connector. Thus the observed distribution matches what one would expect to find.
- The number of words having any given number of connector sequences also follows a log-normal distribution.
- The merger of some top-ranked similar words is explored in detail. For example, the merger of "is"-"was" into one word-class. It seems to go well.
- It appears that there are cliques or almost-cliques of similar words. That is, not just word pairs, but groups of 3 or 4 or more words that are all similar to one-another. This suggests that such "in-groups" of similar words should be merged.

A half-dozen different such in-groups are explored in detail. They look pretty good.

- The above leads to a break-through, an important advance in the theory. One must merge together not only groups of similar words, but only those disjuncts that are shared by a majority of the words in the in-group. This is a two-step process: first, an in-group of similar words is selected. Then, one looks to see what traits (disjuncts) the members of that in-group have in common. Only those traits that most group members share are voted into the group. This solves an important clustering problem: it generalizes, without generalizing over-broadly. I think this is a major advance in the theory, here.

- I believe that the result of merging as described above corresponds (strongly) to word-senses. (Or it should correspond.) This remains unexplored.

- How big should an in-group be? The sizes of a dozen different groups are explored, as a function of the similarity between group members. It appears that, as one loosens the restrictions on group membership, the size of the group grows at first very slowly (not at all) and then grows explosively. The ideal group size is then the largest group below the explosive-growth threshold.

- A good judgment of similarity is needed. This was explored in great depth in the Diary Part Three. The mutual information (MI) survives as a good way of judging similarity. Below, it is discovered that an even better judgment is given by the average of the MI and the log-frequency. That is, extremely rare words can have a huge MI, but this is boring, because the words are rare. We want to know what words are common (frequent) and also have a large MI. This is provided by the average. Pulling it all together, this results in a square-root in the expression:

$$\text{common}MI(w,u) = \log_2 \frac{f(w,u)}{\sqrt{f(w)\,f(u)}}$$

  where f is effectively a dot-product. See below for details. This is... surprising, unheard-of in the literature (I've never seen it before). Its got some nice properties, including being scale free (thus suggesting it lives on a projective space. Perhaps some information-theoretic analog of the Fubini-Study metric? Mathematically, its quite intriguing.) Again, this is a pretty big break-through, as compared to earlier efforts.

- The above ranked-MI is explored in considerable detail for the English dataset. It looks pretty good.

- This part of the diary ends with the first mega-merge, where the top two-thousand most similar and most highly ranked words (similarity measured by ranked-MI) are merged together according to the in-group algo above. It "works", in that it doesn't crash, and the results still look reasonable after a few thousand merges (and many days of CPU time.) Thus, the time is ripe for exploring in detail the results of this merge. This is done in the next part, Diary Part Five. But first,

an assortment of "minor" (but difficult) bugs have to be fixed (and mostly have been!? Tune in next week for the continuing adventures...)

To recap: two important things are found. First, there appears to be a major breakthrough: the in-group shared-trait merge algorithm is discovered, and it seems very promising. Next, averaging together the MI with the word-frequency gives the rankd-MI similarity, which seems to provide an ideal way of ranking frequent similar words.

TODO List

Some items previously explored, but worth looking at again, with the latest datasets.

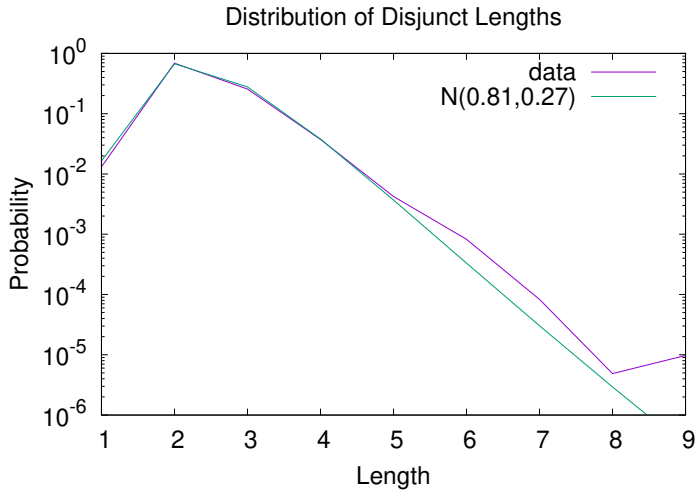
- Words with the lowest self-MI - what are they? What do they mean?

Expt-4/5/6 – Miscellany – Sept 2021

Diary Part Three carried out the bulk of experiment-4 with trimming until it was discovered that trimming to a minimum support of two is even better. Assorted data analysis was done. There are still a few more interesting questions about those datasets.

Length of disjuncts

For a given length of disjunct, how many sections have a disjunct of that length? (Counted with multiplicity) Previously, in Diary Part Two page 14 this was a Gaussian, centered at 8. But this was for a fake language, not for English. Based on what we expect for English, 8 is an alarmingly large number. So, some graphs:[1]



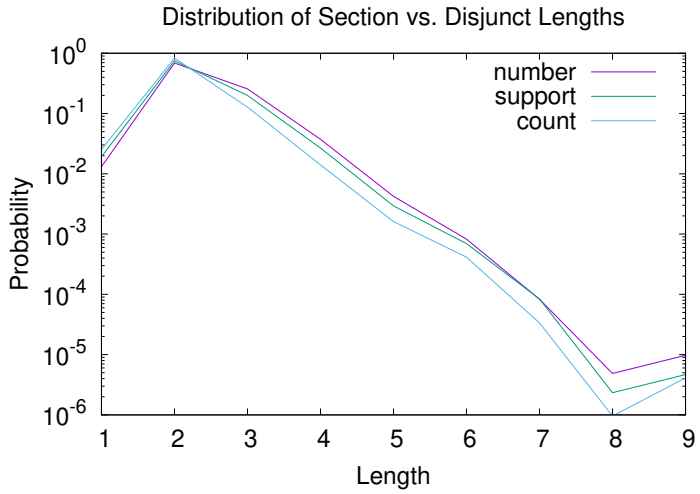
[1] Data for graphs prepared with 'utils/similarity-graphs.scm'

The above shows the number of disjuncts of a given length. That is, fixing the length, just count how many disjuncts there are. Also shown is an eyeballed fit using the log-normal distribution. As before, this is defined as

$$N(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln x-\mu)^2}{2\sigma^2}\right)$$

In this graph, shown is $\mu = 0.81$ and $\sigma = 0.27$. Note that $\exp 0.81 \approx 2.25$ and so we can take the average length of a disjunct as 2.25. This is really pretty nice for English, I guess – we expect common-sense lengths: transitive verbs with a length of 2 (subject, object); common nouns with a length of 2 (determiner, verb-connector) or 3 (determiner, adjective, verb connector); both determiners and modifiers (adjectives, adverbs) should have a length of 1. Punctuation, ditransitive verbs, quotations, etc. have more complex structure. So above looks healthy.

How about the observation count of the number of words that have some disjunct length? Below:



So this shows three very similar curves. These are:

number: How many disjuncts there are of a given length.

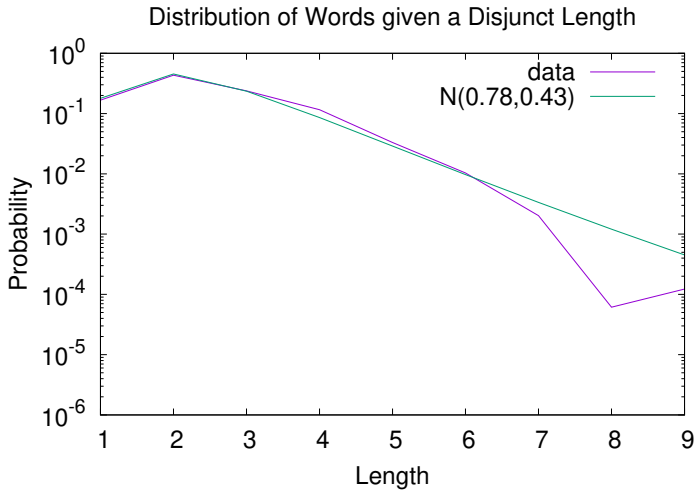
support: How many sections there are, with that section having a disjunct of the given length.

count: The number of observations of sections having a disjunct of the given length.

Recall that a "section" is just a word-disjunct pair. Thus, "support" is when we tally 1 if a section exists, else 0, and "count" is how many times that particular pair was observed.

Looking at that graph, all three have almost the same log-normal distribution, with the mean inching ever so slightly lower. The nice behavior persists.

One last graph: for any given disjunct length, how many unique words are there that have a disjunct with that length? Shown below.



So its similar to the above, but wider. Note that any given word might be counted repeatedly, since it might have disjuncts of various different lengths. So, to read this graph correctly: there are very few words having disjuncts of length 8 (two of them, to be precise). Most words don't have that many disjuncts. Here's the table:

length	num words
1	5446
2	14146
3	7762
4	3776
5	1080
6	338
7	66
8	2
9	0
10	2
11	2

That is, there are 5446 words that have disjuncts of length 1, and 14146 words with disjuncts of length 2, and so on. Recall this dataset has a total of 15083 words, and so apparently, there are some words that do NOT have any disjuncts of length 2! This is somewhat surprising. I wonder what those are. Lets take a quick look.

There are $937 = 15083 - 14146$ words that do NOT have disjuncts of length two. Looking over the list, they are junk. Here's the top-10, their observation count and their rank in the list of all words.

word	count	rank
DR	128	1996
**********	100	2344
Trelawney	97	2390
).=	96	2404
*********	79	2704
U. S	76	2786
http://pglaf.org	74	2844
THORNE	73	2868
CLOUSTON	66	3095
.....	58	3348

So .. these do look fairly skanky. The rankings are ... well, out of 15K words, these rankings are fairly high. They are legitimate words, and are obviously seen fairly often in the texts. Yet, presumably, they occur in some fairly specialized constructions. For example, pglaf almost certainly appears in boilerplate, and as such this text will be very rigid in its structure. Perhaps, then, not surprising that such rigidity might be missing a common disjunct size.

Anyway, we conclude that the distributions look healthy and behave as expected. There are no obvious faults to be seen.

Rankings of mergeable words

That top-40 list of words to merged – what is the ranking of those words? That is, when we go to compute similarities, how deep should they be computed? Here, we assume common-MI provides the merge suggestion, and ranking is by count, as always.

Ugh. Yes, I could, should provide graphs here. I'm lazy, so not today. Basically, the large majority (maybe 90%?) of the top-100 ranked pairs involved words in the top-200 count rankings. Exceptions are (i,ii) and (Old, New) the words of which are ranked around 500, and (don't,don't), and (didn't,don't) whose words are ranked around 300, (ii,iv) and (iii,iv) and (------,--------) whose words rank around 1000.

Conclude: it is safe to compute common-MI out to about 200 or so, and then merge the top-100 out of this list.

Word Similarity with Shapes

Consistent merging requires that merging must be done with shapes. But we judged word similarity without shapes. What happens if we judge word similarity with shapes? After a quick look, it appears to be substantially similar. Many (most?) of the same pairs show up in the top-60 list, with similar rankings. There are some notable differences, some apparently making things worse, others making things better.

Word Similarity without Shapes

FYI. Of the top-ranked 1200 words, there were 51990 word pairs with no overlap whatsoever, and 668610 with some overlap. The total is as expected, a triangle-number:

668610 + 51990 = 720600 = 1200 * 1201 / 2.

Expt-7 – Merging – Sept 2021

Time to open a new chapter. We are ready to merge word-pairs, with a mostly-trustworthy dataset. The goal here will be to merge just one word at a time, by hand, and see what happens. We're doing this because of the disastrously bad July 2021 merge attempt, which generated complete garbage, quite in contrast to the merges from previous years.

Part of the reason for the failure of the July merge was that the merge threshold was set much too low. Part of the problem is that the datasets had too much grunge in them; they needed to be trimmed. Part of the problem is that the default merge used a proxy for the common-MI value, based on ranking. Probably most of the failure can be ascribed to the first part: the merge threshold was set too low.

We've never studied the merges carefully, before. Mostly, they seemed to "just work", and we unleashed the machinery, and let it go. The first time, it worked, the second time, it failed. Anyway, now that we've got a better dataset, its worth looking at these in detail.

Merge is-was

The top word-pair is "is-was", from table on page 23 in Diary Part Three. Lets characterize this. Before the merge:

property	is	was	pair(is,was)	total
$\log_2 \mathrm{tr} MM^T$				33.624
$\log_2 p\,(\mathrm{word})$	-7.763	-7.594	-	
self-MI	5.888	6.067		
MI(is, was)			5.824	
common-MI(is, was)			-1.854	
support(word)	7482	9324	-	
support union			11707	
support intersection			5099	
overlap			0.4355	
count(word)	71894	94214	-	
total Sections				855718

Now, merge the word, but do NOT merge the connectors. That will be a distinct experiment. Issues that need to be addressed: Assorted marginals need to be recomputed. These include:

- The support. This is done automatically, its in the code.
- The MMT marginals. This is done in the supplementary routines.

- New cached similarity score, or at least wipe out the old one. Not just for this pair, but for all pairs with the contributing words. Not done yet.

After the merge, we have:

property	is	was	pair(is,was)	total
$\log_2 \mathrm{tr} MM^T$				33.610
$\log_2 p(\mathrm{word})$	-14.295	-11.957	-6.707	
self-MI	11.560	9.318	5.974	
MI(word, word)	$-\infty$	$-\infty$	$-\infty$	
common-MI(is, was)	$-\infty$	$-\infty$	$-\infty$	
support(word)	2383	4225	5099	
fraction of support moved	0.6815	0.5469		
support union			6608	
support intersection			0	
overlap			0	
count(word)	8554	20808	136746	
fraction of count moved	0.8810	0.7791		
total Sections				850619

Perhaps the most interesting result is that more than 3/4ths of the count was moved over to the merged class. Compared to how much of the support moved, its clear that the merger involved the most-frequently observed sections.

Mystery result: Total number of disjuncts dropped from 205003 to 202922 (as measured by '(pcs 'right-basis-size)') .. how can this happen? This suggests some bug!? That's a difference of 2081 ... oh, I see. The basis shrank, because pcs is NOT looking at WordClassNodes... so there are 2081 disjuncts that got moved out of any word, and into the new WordClassNode. ('pcs' is from 'make-pseudo-cset-api') OK. Unexpected, but OK.

Merge is-was and connectors too

Merging connectors requires using shapes. This is, in fact, the right thing to do, and was hard-won. But this changes everything. So lets track what's going on.

Task	Stats
time to load Sections	183 secs
RAM RSS after load	1.5g
time to explode	125 secs
RAM RSS after explode	3.4g
num words	15083
num disjuncts	205003
num shapes	838580
right basis size	1043583
num Sections	855718
num CrossSections	1922250

So ... How many CrossSections do the words “is” and “was” appear in? What are their counts?

	is	was	union	intersection
Section support(word)	7482	9324	11707	5099
Section count(word)	71894	94214		
CrossSection support(word)	18556	23671	35260	6967
CrossSection count(word)	167112	203939		
Section+Cross support	26038	32995	46967	12066

Clustering merges 46967 sects+cross, as expected (this is the union support size; but in the end only the intersection is merged).

Issue: the merge code wishes to recompute the MMT. The merge code was given shapes to work with so it wants to recompute MMT w/ shapes, but this is not what we’re using for similarity. At any rate, the pipeline needs to be redesigned to use common-MI, instead of using an MI cutoff.

Expanded Clusters

Once a cluster has been formed, it will expand. How does this look like? The top 60 list contains pairs that merge punctuation: .–? and !–? and .–! There is also: It–He, It–There, It–She, This–It, She–He, There–He, There–there. The question is: how does the MI/common-MI change, as the cluster grows?

The exploration below will not merge connectors.

word-pair		MI	common-MI
It	He	4.174	-1.880
It	There	4.251	-2.571
It	She	4.124	-2.705
It	This	4.351	-2.677
She	He	4.385	-2.822
There	He	3.851	-3.349
There	there	-3.459	-12.022

So, It–He gets merged first, and the MI’s are recalculated.

word-pair		MI	common-MI
It	He	$-\infty$	$-\infty$
It	There	6.316	-3.175
It	She	-2.143	-11.642
It	This	0.753	-8.946
cluster	There	4.000	-2.507
cluster	She	4.206	-2.307
cluster	This	4.240	-2.473
She	He	4.007	-5.679
cluster	there	-4.130	-11.231

Comparing, we see that the mergability scores of There, She, This improve, and that is a good thing. The MI of what's left of It has dropped to more of the other words, although the affinity to There improves.

Next in line is She–cluster, according to the new rankings. So lets merge that.

word-pair		MI	common-MI
It	She	2.664	-9.239
cluster	There	3.979	-2.442
cluster	She	$-\infty$	$-\infty$
cluster	This	4.214	-2.414
cluster	It	$-\infty$	$-\infty$
cluster	He	$-\infty$	$-\infty$
She	He	8.815	-3.276
cluster	there	-4.213	-11.228

Interestingly, the remainder of what's left to She-He has the MI shoot way up, and the mergability of this remainder is relatively high. The MI between the cluster and There, This is unaffected, or even goes up. The common-MI scores improve. Looks like This is next, so do it.

word-pair		MI	common-MI
It	This	5.047	-3.176
cluster	There	3.989	-2.375
cluster	This	$-\infty$	$-\infty$
cluster	there	-4.015	-10.973

As before, two things pop out: the mergability of cluster-There improves, and what is left of It-This is quite highly mergeable (just like He-She, last time).

What's going on with these remainders? It's worth taking a look. Below the diagonal are the MI's after the formation of the cluster. Above the diagonal are the initial MI's, before clustering.

	It	He	She	There	This
It		4.174	4.124	4.251	4.351
He	$-\infty$		4.385	3.851	4.049
She	2.664	8.815		3.798	3.980
There	9.015	2.443	2.868		4.081
This	5.047	2.168	2.178	3.906	

Before merging, these were all in the same ballpark. After merging, we see that the He–She MI really popped! So did It–There. The rest are meh.

And again, the common-MI:

	It	He	She	There	This
It		-1.879	-2.705	-2.571	-2.677
He	$-\infty$		-2.822	-3.349	-3.358
She	-9.239	-3.276		-4.178	-4.203
There	-1.826	-8.586	-8.861		-4.095
This	-6.799	-9.865	-10.555	-7.766	

Before the merge, the It–He mergability suggestion was rather quite high; the rest are OK, but lower down.

After the merge, the remaining It–There mergability ranking is now very high, among the highest. The mergability ranking for He–She is decent... but a second-order priority.

Revised clustering strategy

The above analysis suggests a revised clustering strategy. Let's call it the "democratic" strategy, or the "peer" strategy. It goes like this:

1. Find a word pair with the highest common-MI.
2. For each word in that initial pair, look for other words that have a regular MI that is close to that particular word-pair, or better. "Close" means "no worse than 1.0 less". Of these, maybe reject those with a common-MI of more than 5.0 less.
3. The above list of similar words defines the initial peer group.
4. Given the peer group, determine which disjuncts belong to it by voting: if more than half the words share some common disjunct, then this disjunct will be admitted into the merged cluster.
5. Perform the merge, merging all of the words into the cluster, and accepting those disjuncts that have been voted on.

The above would seem to create a more balanced cluster, as opposed to the agglomerative merge. The problem with agglomeration was demonstrated above: what was left on She–He and on It–There was really quite a lot, and probably should have been a part of the main cluster. It wasn't, because agglomeration threw these away, when forming the initial He–It cluster. The democratic voting of step 4 appears to solve this problem.

Democratic voting also appears to address another old bugaboo: how to perform "generalization". The pure-intersection clustering performs no generalization whatsoever. The democratic voting allows generalizations to be admitted, when they weren't before. So this seems like a big win.

An open question is how general the generalization should be. Should it be a majority vote? Perhaps it should be an evaluation of the intra-cluster MI after the generalization is made? Could there be some relaxation process, which picks and chooses what disjuncts get admitted, so as to maximize the intra-cluster MI and to minimize the extra-cluster (inter-cluster) MI? Right now, voting is easiest. Perhaps some kind of relaxation algo might be better, in the future. The relaxation algo is certainly harder.

We can steal some ideas from the Tonnoni Phi spin-glass canon: the extra-cluster MI should be best described by an ultrametric, so that the distance from the cluster, to anything else, has the ultrametric property. Recall the definition of an ultrametric: it is the distance between two leaves on a tree. Two leaves are close if they are on the same branch; they are distant if they are on different branches. We want all words in a cluster to not only be close, but to be "maximally far away" from anything else, where "maximally far" s intended in the ultrametric sense. (The ultrametric appears in the theory of spin glasses. It is the "obvious" metric for describing human brains in a social network: Every neuron in my head is much more closely connected to every other neuron in my head, than it is to your head. The only connection between my head and yours is the fact that you are reading these words. That is both a spin-glass ultra-metric, and also describes there the lest cruel cut can be made. Phew. Sorry for the digression, but this seems important.)

Word-sense disambiguation

Issue: If we compute MI and common-MI using only Sections, and not Shapes, but then do a proper merge (i.e. merging connectors) then we have a post-merge issue: the new sections will have have disjuncts with the clustered connectors in them.

This appears to cause several issues, but does it?

First issue: When the MI's get recomputed, the shape object will be used. So this is a bug, we need to use the non-shape object when recomputing the MI. So this needs to be passed around as a distinct argument to the functions.

Second issue: When the MI's get recomputed, it might seem that connector sequences that should have been comparable are no longer being compared, because one connector sequence has a plain word, while another has a WordClass. Upon further reflection, this is NOT an issue. Connector merging means that all of the various connector sequences have been updated ("correctly") with the new WordClass, so the sequences remain comparable. Those that have NOT been updated MUST be considered as belonging to a distinct word-sense.

In particular, that means that the correct connector merge code automatically handles word-sense disambiguation: whatever has not been merged, including unmerged connectors, MUST be understood to belong to a word-sense that is distinct from the WordClass.

Conclude: no issue here, except to be careful in recomputing the MI.

Conclusion to Expt-7

Looks like everything works great. The functional review provoked a code review, both the functional review and the code review pass, it seems that merging works great. Caveat: at least the non-connector-merge variant works great. The connector-merge code is really complicated, and was developed and debugged during the random-corpus exploration phase. It was completed, with no known bugs, but it is just complicated enough and squonky enough that perhaps there are issues. There is a test-suite of 13 unit tests. These pass. So I guess its pretty tight?

Examination of the agglomerative clustering strategy exposes it's weakness (well-known in the industry) However, unlike the conventional clustering algos, we have the ability to pick and choose, via "democratic voting", which disjuncts get admitted into the cluster. This appears to address an important open issue, of (a) a way of performing generalization, and (b) a good way way of performing generalization. The earlier fuzzy-mixin performed generalization, but it was fairly non-selective, and some naive MI calculations seemed to indicate that it always made things worse. In particular, it made word-sense disambiguation ugly. So this new democratic algo appears to be a major conceptual improvement.

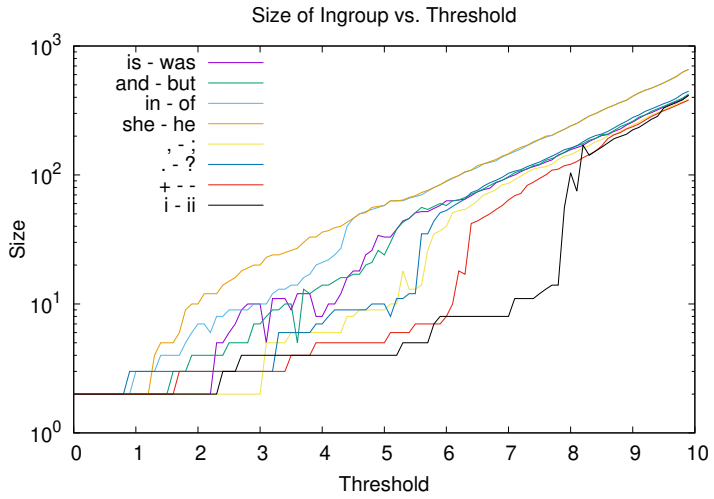
Clique Sizes (Oct 2021)

The democratic-voting idea suggests that an initial in-group needs to be selected. This in-group will be a clique or an almost-clique. But how are we to find it? How similar do words need to be, in order to belong to an in-group? Here, we explore the size of the clique, vs. the allowed variation of MI within the clique.

Consider some given initial word-pair, and a threshold ε. The initial pair has some initial value of common-MI S_0. A search is performed to find words that have a common-MI greater than $S_0 - \varepsilon$ to at least 70% of the members of the current in-group. To admit a third word into the ingroup, the common-MI must be greater than $S_0 - \varepsilon$ to both the current members. To admit a fourth word, the threshold must be exceeded to at least two of the three current members, and so on. The formula is $vote = \lfloor 0.7n + 0.5 \rfloor$ where n is the number of members in the current in-group. In general, the larger that ε gets, the larger the ingroup will be. The process is not monotonic, since it depends on the order in which words are added. For small ε, a tight-knit grouped might be formed. As ε gets larger, one of the early members might be quite different, and becoming a peer to it might be a challenge, thus decreasing the size of the in-group! These decreases are visible in the chart below. To avoid this effect, the words are ranked by frequency first, so that the most frequent words are considered first.

Common-MI in-groups

The chart below shows the results for eight different initial word-pairs. All eight of these appear in the top-15 list of pairs with the highest common-MI. Many are punctuation; this is inevitable, as punctuation is very frequent.



So... interesting ... There are two asymptotes: "she-he" and "in-of" are one, and all the others land on the other. Note that the asymptotes are just lists of garbage words. Still, its curious that there are two. Not shown in the graph: experimentally, the asymptote is almost exactly given by $\exp(\varepsilon/2)$. Missing theory: why? Why the natural exp? Why not something else, like 2^{ε}?

Some clusters grow easily - "she-he'" and "in-of" both grew quickly. "Is-was" took longer to take off. Here are some samples, taken at the maximum value of ε at which the results look good. That is, for larger ε, the ingroup starts looking ugly.

epsilon	size	ingroup
3.0	10	might must may should will could would had is was
3.0	7	when so for as that and but
3.4	13	from by with at on for as not that to and in of
1.6	5	they it I she he
5.0	9	: ! _ " " . ###LEFT-WALL### , ;
5.0	10	: ! _ " ; " ###LEFT-WALL### , . ?
5.9	7	] ([* _ + —
6.7	8	R M $ D N p i ii
2.3	13	Yes We This They You What She And The He I It There

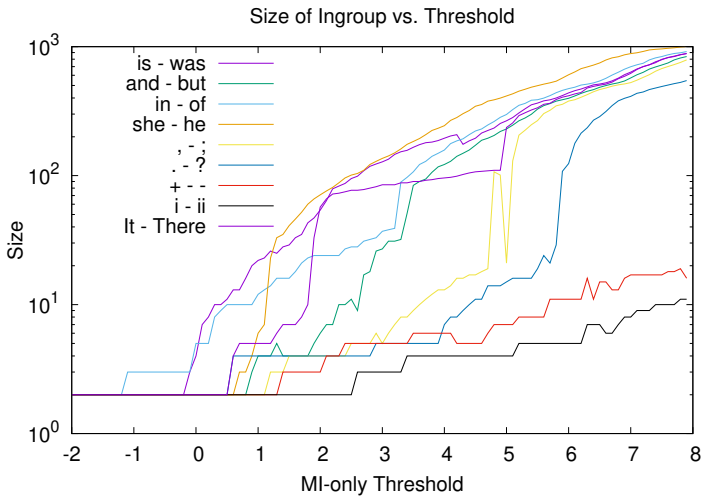
The initial seeds that start the ingroup appear as the last two members of the lists. Note that some lists end up being similar, even though the seed-pairs are different.

The two collections of punctuation started with different seed pairs, but converged to the same place. One started with comma-semicolon, the other started with period-question-mark. Similar results for the prepositions. The "and-but" forms the seed in the first group, but most of that group reappears in the ingroup around the seed "in-of". The last one is curious – its all sentence-starter words (That's why they are capitalized).

The above does not teach a clear lesson in how to pick ε. Picking a fixed value does not seem like a good idea. A reasonable merge strategy might be to start with $\varepsilon = 1$, merge what there is, and then enlarge some more. As always, the question is when to stop enlarging.... no answer for that, yet. There are two indicators: stop enlarging just before the size of the group increases rapidly, and stop enlarging when the size of the group suddenly drops. This seems to be viable – the calculations are relatively fast.

Plain-MI in-groups

Wait ... we did the wrong thing. The above used common-MI to determine similarity to the in-group. Perhaps just the regular-MI should be used. That is, common-MI is great for identifying the initial word-pair to form the seed of the clique, but perhaps regular MI is better suited to actual similarity. Here are the graphs, again, this time using regular MI. Note that the threshold goes negative. For example, for $\varepsilon = 0$ for the initial seed "is-was", there ae already four members of the ingroup. This means that there ar two other words whose MI to "is" and "was" is higher than the MI between "is" and "was". Cool! Here's the graph:



So this is very different than the earlier graph. Still all over the place. Lets take a sample:

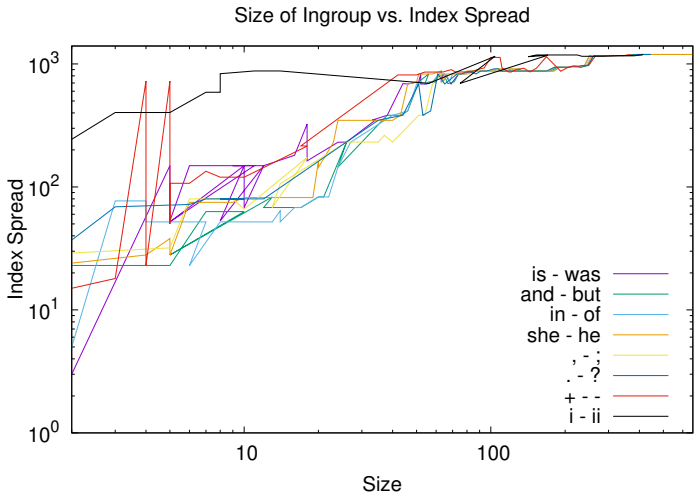
epsilon	size	ingroup
0.4000	10	doesn't takes isn't appears wasn't wasn't seems seemed is was
2.3	10	nor till until since than or for as and but
0.4	9	toward towards against through upon into from in of
1.1	7	thou it's it's we they she he
4.0	13	– ...] — : ! _ " " . ###LEFT-WALL### , ;
5.4	16	– ... . . . '] — : ! _ " ; " ###LEFT-WALL### , . ?
5.1	7	– {] ([+ —
7.9	11	b R Y $ E C D N p i ii
1.8	10	These Here That We This They She He It There

So this has a very different flavor. With common-MI, we saw words like "could would should" being judged similar to "is was". Here, we see "appears seems" showing up, while "could would" is entirely absent (and continues to be absent until the ingroup has size 25!) This means that – appearing and seeming is much more like being, but desires and statements of ideals, like "could should" occur far more frequently in the text. (High frequency words boost the common-MI but have no effect on plain MI.) So these are two very different semantic directions to go off in! How to resolve this? What, exactly, are we doing here?

The other groups seem to have a slightly different flavor, but nothing quite as distinct. Gut impulse is to say that common-MI is a better classifier, even though there's some semantically interesting stuff happening with regular-MI. But we are not yet ready to pursue semantics; we're still working on syntax.

In-group ranking spread

Words are added to the in-group one at a time, starting with the most-frequently occurring words. By the time that the construction of the in-group is done, what is the difference between the highest-ranked word in the in-group, and the lowest-ranked? This is graphed below. It's done with common-MI to determine in-group membership.

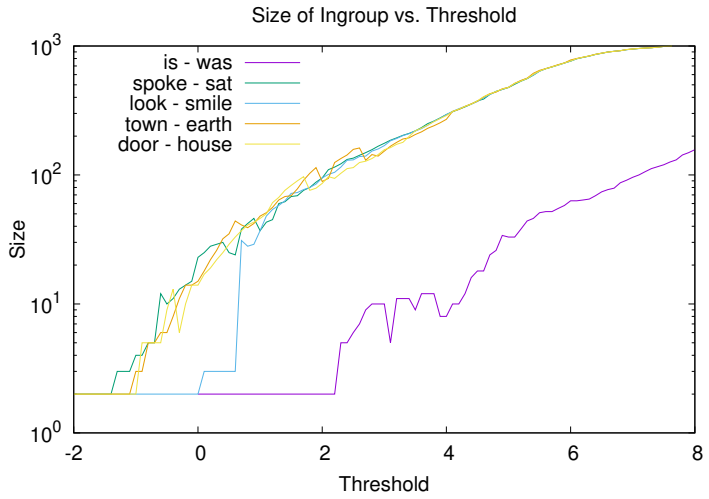


What this chart shows is that when the in-group has fewer than 10 members, all of those members are within 100 of each other in the word-frequency ranking (with exceptions as visible in the graph.) Note this is *not* a function, the lines can zig-zag left and right. Some cliques are tighter than others.

Recommendations: if we're going to compute similarity, do so in a band of about 200 from the diagonal.

Mid-level pairs

The above results were all for seed word-pairs that were top-ranked in terms of common-MI. How about something from the middle? Here are some seed word-pairs whose common-MI is about 4 less than the top-ranked pairs above. They are approximate in the 10K'th position of all ranked pairs (ranked by common-MI). They still look pretty good, in terms of English. Those that have an MI of 8 below the top-ranked pair start looking like garbage. The only problem with this exploration is that ... pairs that are 10K down are ... far far from consideration. Words in them are likely to get swept up much earlier.



Shown for comparison is the old "is-was" group. Note that the ingroups are quite large even for negative ε. This just says that these seed-pairs are likely to be swept up in earlier in-group formations.

Let's assess their health.

epsilon	size	ingroup
-0.7	5	looked had was spoke sat
0.6	3	glance look smile
-0.3	11	church United sea city country people house world most town earth
-0.6	5	end head same door house

Well ... mixed bag. In the first row, "had was" are likely to be swept up in some other group, leaving "looked spoke sat" fairly healthy. In the third row, "most" is likely be be swept up in some other in-group; what's left looks OK. The last row looks haphazard. Presumably "same" will end up in some other in-group, but what about "end"?

At this point, there seems to be no choice: we have to run the earlier clusters, and then see what's left over. Perhaps this can be done manually? So we can get a better idea? That requires running 10-20 clusters by hand, and then repeating this exercise.

Expt-8 Merging Farther (Oct 2021)

Resuming where expt-7 left off. This time, do merge with shapes, and get further down the process. Lay groundwork for the cliques above.

One major issue is that, after merging connectors, all of the previous similarity computations are rendered invalid. We need to loop over all merged connectors, and

recompute those similarities, if they are/were in the range of computable similarities... then, after that we need to re-rank, to find the next items in the list.

Done. Fixed some bugs, seems to work well, happy and healthy ... code being created in ‘agglo-rank.scm‘ based on results and outline. Nothing notable pops out, other than it looks great! Yahoo! We're rolling again!

Expt-9 Ranked Merge Development (Oct 2021)

So since expt-8 went so swimmingly, ranked merge development should go smoothly, too, right? Heh. Not so lucky.

Confusion. We are calculating the marginal similarity as

$$P(w) = \frac{\sum_d P(w,d) P(*,d)}{\sum_d P(*,d) P(*,d)}$$

The normalization here is so that $P(*) = 1$ which is what we want so that we can interpret these things as probabilities.

What is the value of $\sum_d P(*,d) P(*,d)$? I think we want to add this as a constant to common-MI. As a constant, it changes nothing, but it prevents common-MI from being always negative. We don't have it handy because we are not storing P's we are storing N's; we need to compute it. It is

$$Q = -\log_2 \sum_d P(*,d) P(*,d) = -\log_2 \frac{\sum_d N(*,d) N(*,d)}{\left(\sum_d N(*,d)\right)^2}$$

Wow. The mind boggles. Numerically, for the t1234 dataset, it is 11.946 which is actually a healthy value.

The mind boggles because this is... well, its a scale parameter. It's also a mean-square. It's a screwy matrix norm of some kind. It feels like there should be some kind of physical interpretation for it. Some kind of graph spectral foo-bar, but I don't know what that is.

There's a generic problem here: we've got some kind of graph spectral something-or-other here, but I don't know that theory. There's a hint that its simple, elegant, and should have some obvious interpretation. Fer Chris sake, the last figure of Part Three was a Gaussian! There's got to be some kind of generic theorems on this stuff.

Note the common-MI was scale invariant. It was defined as

$$\text{common}MI(w,u) = \log_2 \frac{f(w,u)}{\sqrt{f(w) f(u)}}$$

with

$$f(w,u) = \sum_d P(w,d) P(u,d)$$

It's scale invariant in that we can multiply f by any constant, and the common-MI is unchanged. So its projective...

Projective suggests that this is some information-theoretic analog of the Fubini-Study metric, or something like that.

Ranked MI

OK, so the above ruminations suggest that using ranked-MI is better than using common MI. We define this as

$$\text{ranked}MI\,(w,u) = \text{common}MI\,(w,u) + Q$$

Ranking cutoffs

As a practical application, ranking works best if we add cut-offs: reject all pairs with an MI of less than 4.0. This knocks out some pairs that have a high ranked-MI but otherwise crappy MI.

There's also another *de facto* filter at play: if we compute sims only for the top-200 words, any highly similar words that aren't in the top-200 simply won't appear. For example, the roman numeral '(Word "i")' is ranked 458'th in the list of words, '(Word "ii")' is ranked 701'th. They are highly similar: MI=12.52 and a ranked-MI of 9.05 which would have placed it fourth in the table below. But they are too far away from the top-200 and so are never considered.

Worse: they are not even within the diagonal band that is 200 wide, and so will never be considered... perhaps the band needs to get wider over time....

So, with that in mind: looking only at the top-200 words, there are 1338 pairs with an MI of greater than 4.0. Accepting only those, here's the list of the top-20 word pairs:

word-pair		ranked-MI	MI
—	+	9.1671	9.3888
;	,	9.1620	5.0377
is	was	9.0605	4.6242
and	but	9.0319	4.8387
.	?	8.9357	5.8409
!	?	8.9218	7.1206
It	He	8.8665	4.9165
[	+	8.8158	9.7555
”	"	8.2227	5.2267
No	A	8.1991	4.7015
in	of	8.1951	4.4320
It	There	8.1493	4.6029
!	.	8.0668	5.3339
‘	“	8.0408	4.5713
the	his	7.9914	4.2834
She	It	7.9629	4.6488
This	It	7.8279	4.4001
–	’	7.8212	5.3334
She	He	7.8169	4.8770
”	’	7.7406	4.4218

I'm enjoying that both the left-slanting and the right-slanting single and double quotes show up in this list correctly paired with each-other.

Lets take a look at what we missed with the MI cutoff. There are 310 words pairs with an MI between 3 and 4. The top-5 in this list are:

word-pair		ranked-MI	MI
she	he	8.1922	3.7175
and	as	8.1079	3.4660
had	was	7.7289	3.7725
and	for	7.6961	3.6477
they	he	7.5188	3.1174

Notable is the-a, which appears 8th in this list, with ranked-MI = 7.4818 and MI = 3.1485. Capitalized The A appear 15th with ranked-MI = 7.2114 and MI = 3.4420.

One more time, for MI between 2 and 3:

word-pair		ranked-MI	MI
###LEFT-WALL###	,	7.9065	2.2994
he	I	7.6618	2.0831
it	he	7.4155	2.3143
You	I	7.2956	2.0871
then	he	7.2354	2.4393S

So this looks pretty reasonable until we get to the last entry, which is pretty crappy. Except for that one, the next 15 look pretty decent. So ... things seem healthy.

What if we are more demanding? What if we insist on an MI of greater than 6, which would knock out most of the words in the early list? Here we go:

word-pair		ranked-MI	MI
—	+	9.1671	9.3888
!	?	8.9218	7.1206
[	+	8.8158	9.7555
—	[	7.5186	8.0631
should	might	7.1761	6.2910

Yikes! The next four entitles are also punctuation. I think the corpus had a bunch of ASCII-formatted tables in it. These are reasonable suggestions, but are actually .. boring. So setting MI=4 for this dataset seems like the right thing to do.

Of course, long-term, this MI threshold needs to be auto-discovered. However, 4 has been working great for me for the last 5-6 years, so .. hey. It's dataset-size dependent, and all my datasets end up being as big as I can make them and still be manageable.

Duplicate Cluster Names

OK, so, during merge, after 26 previous merges, we hit a merge of (‘###LEFT-WALL###‘, ‘:‘) which is reasonable, except that we’d already merged these two before, at iteration 12. So what is being merged now is presumably the left-overs of that earlier merge. Is this sane? Is this a bug? Some details.

Previous: at round 12, merged‘###LEFT-WALL###‘, ‘:‘ from a start of ranked-MI = 7.7349 MI = 4.1172.

At round 26, we merge (‘]‘, ‘:‘) because they have ranked-MI = 6.9107 MI = 8.5029. This apparently splits off enough of the colon that what is left over is now similar to the left-overs from LEFT-WALL. (Although I’m confused? Didn’t the overlap merge pre-emptively remove all traces of similarity? So how did this come back?)

So, at round 27, we get ranked-MI = 7.0566 MI = 6.5282 for ‘###LEFT-WALL###‘, ‘:‘ and so these dregs get merged again. Surprise!

The confusion: the overlap merge should have clobbered the dot product, and left the dregs completely orthogonal to the cluster. Because overlap merge means to sum over all basis elements that are non-zero in common, which are the *only* basis elements contributing to the dot product. So, post-merge, the cosines should be exactly zero. All three of them: the cluster should be orthogonal to each of the contributing words, which are orthogonal to each-other. Further stripping out disjuncts from the colon should not have changed it’s orthogonality to either the first cluster, or to the left wall.

So it seems like we’ve got a bug somewhere. Is that correct? Is it possible that connector merging brought these two back in parallel? Lets find out...

word-pair		cosine	overlap
###LEFT-WALL###	:	0.3317	8.746e-4
class	:	0.3584	0.0103
###LEFT-WALL###	class	0.4809	9.610e-4

So some of the overlaps are tiny, but the cosine is huge! The only way this can happen is if most of the disjuncts have very small counts, so that the vector length is unaffected by them (these appear in the denominator of the cosine), while the overlapping disjuncts are the ones with the high counts.

And indeed that is what happened! There are 56 entries in the overlap of left-wall and colon, out of about 46K. All 56 of these are CrossSections. All have large counts. All have Shapes involving a WordClassNode. Thus, although merger will orthogonalize vectors, later connector merges can add back parallel components. Again: this is a sheaf, and its got some kind of torsion or curvature. The connector-merge is a kind-of parallel transport around a loop, and bends things around. I’m pretty sure this is the right way to think about it, but actually writing out the formal mathematical details of this would be quite the project. Almost surely a good exercise, as it will clarify what the heck is going on, but .. a difficult one.

On with the show!

In-group Merge Results

Some early results. This uses quorum=0.7, so that a disjunct is placed in the cluster of 70% of the in-group shares that disjunct in common. The ingroup is selected according to a bunch of hard-coded parameters in the code, see ‘cliques.scm‘ in the 12 Oct 2021 git tree for details.

in-group	nwords	top-rank-MI	Q
	-	-	11.946
] ([* _ — +	7	9.1671	7.555
; ,	2	9.1620	9.131
is was	2	9.0605	9.416
as that and but	4	9.0319	10.046
! " ###LEFT-WALL### . ?	5	8.9357	10.466
There I It He	4	8.8665	10.650
They She He It There	5	8.6036	10.675
from to in of	4	8.1951	
’ ‘ “	3	8.0408	
this he a the his	5	7.9914	
my her the his	4	7.8097	
my this her a the his	6	7.8352	

The Q reports the Q value defined up above. (its called ‘mmt-q‘ in the code). The Q is that after the merge is performed. Thus, the first row is the initial Q of the starting dataset.

There's actually a bug in the code w.r.t. Q – because it is changing from merge to merge, any cached similarity scores will use the old Q from earlier rounds. These will no longer be correct for the new Q value, even though the MI is unchanged and the $\log P$ marginals are unchanged. So we have to stop adding Q into the cached scores. Ooops!

There's another bug with voting. After an in-group is selected, as judged by MI, it turns out that only a small number of disjuncts are shared by more than 70% of the ingroup. Sometimes less than 1%! These are clustered, but then, out of what's left, a new very similar in-group is created, because the MI is still high. This is seen in the last three lines of the table above, and also in the It-He lines in the table.

Heh. Now its stuck in an infinite loop, merging 0 sections each time around!

Solution would seem to be to kick out the most recent members from the in-group, until at least 20% of the disjuncts of the smallest member are shared by all. The 20% figure allows the smallest member to have five different word-senses (i.e. to belong to five different clusters.)

We shall call this number the “commonality”, and request that the ingroup have a minimal commonality. The code documentation explains this better than here.

In-group commonality

OK, so this needs further study. The failing in-group was ‘There‘ ‘It‘ ‘He‘ ‘They‘. Initially, the following ranked-MI's are observed: [2]

	There	It	He	They
There	8.142	8.149	7.246	6.007
It		9.551	8.866	5.710
He			8.6763	5.969
They				6.938

After this, there are 5 rounds that merge as documented in the previous section.

At round 6, ‘He‘ ‘It‘ ‘I‘ ‘There‘ are merged, seeded by ‘It‘ and ‘He‘. At this time, ranked-MI = 8.8665 MI = 4.9165 (‘It‘, ‘He‘) At this merge, approx 600 out of 6000 sections each are merged.

Round 7 is seeded by ranked-MI = 8.6036 MI = 10.873 (‘It‘, ‘There‘) and merges ‘There‘ ‘It‘ ‘He‘ ‘She‘ ‘They‘. Approx 20 sections each get merged, so this is already a small number.

Round 8 is seeded by ranked-MI = 8.6293 MI = 10.875 (‘It‘, ‘There‘) and merges ‘There‘ ‘It‘ ‘He‘ ‘They‘ with less than 20 sections each contributing.

Round 9 starts with ranked-MI = 8.6296 MI = 10.878 (‘It‘, ‘There‘) and attempts to merge ‘There‘ ‘It‘ ‘He‘ ‘They‘. The result of this attempt is zero sections contributing! How is this possible?

If we stop there and look, we get:

	There	It	He	They
There	9.099	8.630	2.696	4.897
It		9.830	5.202	-3.004
He			8.292	2.590
They				5.672

The negative ranked MI at ‘It‘, ‘They‘ shows the problem. BTW, the regular MI is -2.264 for these two. Recall that the ingroup is determined by stopping the addition of members just before membership explodes. It says nothing about all of the members in the in-group having to have a high MI. Perhaps it should?

[2] Use this code:

```
(load-atoms-of-type 'SimilarityLink)
 (define sap (add-similarity-api covr-obj #f "shape-mi")
 (sap 'pair-count (Word "There") (Word "It"))
```

The first number is MI, the second is ranked-MI.

Its clear that backing off the one word ‘They‘ will resolve the issue. It’s the one halting progress. The quorum is set at 0.7, so majority vote requires $\lfloor 4 \times 0.7 + 0.5 \rfloor = 3$ agreements. Obviously, none of these will come from ‘They‘, and so ‘There‘ ‘It‘ ‘He‘ would have to be unanimous. But they can’t be: they were already merged in round 7 and 8.

Conclude: we have two solutions: reject ingroup members with a negative MI to other in-group members. Also apply the commonality requirement.

In-group merge results, restarted

As above, but restarting with a commonality of 20% and an absolute lower bound of 1.0 for the ranked-MI. The Q value is the Q at the start of the merge. After the merge is completed, Q is recomputed. Here, “kick” in the in-group column means the last member is being kicked out. That is, in-groups start large, and then members are kicked out until a suitable commonality is reached.

Here are the first four merges.

mrg	in-group	nwrd	rank-MI	Q	overlap	tot dj	commonality
1	— + “ ” _ ’)	7	9.1671	11.946	106	42880	0.25%
	kick)	6			181	40648	0.45%
	kick ’	5			107	38983	0.27%
	kick _	4			170	31074	0.55%
	kick ”	3			644	18246	3.53%
1	— +	2		11.946	121	5014	2.41%
2	; ,	2	9.1620	5.506	14201	148717	9.55%
3	is was	2	9.0605	8.449	11776	46508	25.32%
4	but and that as	4	9.0319	8.851	3180	94563	3.36%
	kick as	3			8829	83719	10.54%
4	but and	2			4842	55787	8.68%

Here are some merges further down the queue. Merge 18 appears to be the first one accepting three items. If we had set commonality to 10%, then merge 4 would have been the first.

mrg	in-group	nwrd	rank-MI	Q	overlap	tot dj	commonality
18	could should will may	4	7.4589	10.908	1230	13959	8.81%
	could should will	3			2617	12707	20.59%
19	are were had did	4	7.3664	10.912	1180	32338	3.65%
	kick did	3			4666	29402	15.87%
19	are were	2			3436	14476	23.74%
20	there There	2	7.3611	10.922	210	6136	3.42%
21	There It This That	4	8.3244	10.951	115	4914	2.34%
	kick That	3			409	4244	9.64%
21	There It	2			124	2732	4.54%
22	There She What ...	8	7.2281	10.950	34	12058	0.28%
	kick When	7			71	10820	0.66%
	kick We	6			91	9855	0.92%
	kick This	5			45	8386	0.54%
	kick They	4			71	7341	0.97%
	kick You	3			196	5184	3.78%
	There She	2			102	3639	2.80%

And still further down:

mrg	in-group	nwrd	rank-MI	Q	overlap	tot dj	commonality
50	have had is	3	6.4826	11.228	452	29731	1.52%
	have had	2			1	16896	0.01%
51	saw am took	3	5.6057	11.227	361	5112	7.06%
	saw am	2			61	3761	1.62%
52	is used had ...	5	5.6081	11.230	8	26595	0.03%
	kick still	4			15	24714	0.06%
	kick nothing	3			338	23026	1.47%
52	is used	2			27	14055	0.19%
53	1 2 |		7.0466	11.226	219	2353	9.31%
	1 2				216	853	25.32%
54	the whom which if ...	5	5.5613	11.226	52	100367	0.05%
	kick what	4			122	96193	0.13%
	kick if	3			1405	91768	1.53%
54	the whom	2			87	81711	0.11%

And again, further down:

mrg	in-group	nwrd	rank-MI	Q	overlap	tot dj	commonality
95	H F	2	5.4067	11.374	110	519	21.19%
96	asked saying ")	4	5.4208	11.373	4	14972	0.03%
		3			55	12288	0.45%
96	asked saying	2			9	1474	0.61%
97	On on In during	4	4.9823	11.375	20	11105	0.18%
		3			239	10517	2.27%
97	On on	2			120	9212	1.30%
98	S O	2	4.9756	11.374	38	634	5.99%
99	no been not nothing	4	4.9733	11.373	64	20906	0.31%
		3			709	19476	3.64%
99	no been	2			30	7193	0.42%
100	didn't didn't took seemed	4	4.9436	11.372	64	3650	1.75%
		3			384	2785	13.79%
100		2			351	1257	27.92%

Hmm. Thoughts about specific merges:

- There are some close shaves, e.g. merge 96 which proposes something ugly. Then there's merge 99 which doesn't seem right. And there's merge 100 which is prettiest in its final form. Note that these are all late-stage merges.
- Merge 53 is due to some books including many tables.

General remarks:

- A commonality of 20% sometimes seems to high. Sometimes seems just right. Worth trying 10% or 5%.
- A quorum of 0.7 is probably too high! Perhaps 0.5 would be better. That is, half the members share at least 20% of traits.
- We want a quorum of the members to share a commonality of traits .. but the relationship of both of these to the MI seems very unclear. These groups start with very high MI, and yet, seem to have little in common. How is that? Is it because we are counting commonality wrong?
- We are counting commonality by the *number* of disjuncts shared. But the MI is computed by weighting high-count disjuncts highly, and low-count disjuncts low. Is there a way to weight the commonality score likewise? But how, exactly? How do we turn this into a scale-free number? The problem is that, by definition, the count will be zero for some (this is how generalization happens.) I can't think of a worthy alternate variant of commonality.

Possible solutions:

- Looks like sometimes commonality drops as words are removed. This is is interesting and .. useful. Seems like a good strategy would be to accept the largest group before commonality drops! This is cool stuff!

- Perhaps it is time to reintroduce the noise floor, so that disjuncts with low counts are swept up during the merge. ... DONE.

Sooo...

More In-group merge results

Again. Set quorum=0.5, commonality=20% and noise-floor = 4. The first five merge results:

mrg	in-group	nwrd	rank-MI	Q	overlap	tot dj	commonality
1	— + " " _ ')	7	9.1671	11.946	83	14287	0.58%
	kick)	6			160	13677	1.17 %
	kick '	5			815	13152	6.20 %
	kick _	4			388	11162	3.48 %
1	— + " " _	5					
2	; ,	2	9.1620	3.762	3687	48332	7.63 %
3	is was	2	9.0605	7.036	4184	14950	27.99 %
4	but and that as	4	9.0319	7.483	4377	30763	14.23 %
	kick as	3			3151	27116	11.62 %
4	but and that as	4					
5	? . LEFT-WALL to it :)	7	8.9357	8.143	54	75929	0.07 %
	kick	6			354	75324	0.47 %
	kick	5			2629	74470	3.53 %
	kick	4			2051	65759	3.12 %
5	? . LEFT-WALL to it	5					

Clearly visible are two effects: the climb-back-up because commonality dropped, and the much smaller total dj number, which means most dj's really had a count of 4 or less. And we're taking the vote away from them.

To be clear: the noise-floor says that any section with a count of equal or less than the floor will be ignored during voting, and will be automerged, no matter what the outcome of voting. (Perhaps we should have two distinct noise-floors: one for adjusting counting, the other for merging?)

Nov-Dec 2021: The Big Merge

Ran a long-running merge. Working with Expt-9 as above, with the "final" merge algo fully implemented and "debugged", started a long-running merge sometime in Oct, Nov and back-burnered it, kicking it along every now and then, monitoring progress. Now its time to examine it and see what we got.

Some stats:

- Starting dataset was ‘run-1-t1234-tsup-1-1-1.rdb‘. This is copied to ‘run-1-t1234-shape.rdb‘ and shapes are added. This is copied to ‘r9-sim-200.rdb‘ and the first 200x200 similarities are precomputed. This is copied to ‘r9-merge-h.rdb‘ and similarities start being computed.
- Data file: ‘r9-merge-h.rdb‘ there’s also an earlier one with about 2/3rds the total merges: ‘r9-merge-h-save1.rdb‘
- As above, parameters were quorum=0.5, commonality=20% and noise-floor = 4. AKA ‘(in-group-cluster covr-obj 0.5 0.2 4 200 100)‘ in the current codebase per late-Oct mid-Nov git repo contents. This command was run maybe 20 or 30 times (??) by hand; each run was maybe 12 hours (??), I’d scan it, look for obvious failures (there weren’t any), and then run it again. Each run picked up where the last left off, and would run deeper.
- 18977 minutes total CPU (= 316 hours = 13 days) Of this time, 1988 minutes spent in gc, or about 10.5%.
- 126 GB RAM resident (this grew slowly). The guile heap size is 7GB of which 6.7GB is free, so 387 MB in actual use. The RAM blowup is due to pattern queries not being erased. (See below)
- Total of 124820805 atoms (thats 125M). RAM is 131717108 KB so thats 1.05KB/Atom so that’s utterly normal.
- Most of the atoms are search queries that were not deleted! So that’s a bug. After deleting the junk, have 7129235 atoms (that’s 7M), so very tolerable.
- 5190946 SimilarityLinks, (that’s more than 5M of them) so similarities between approx 3222 words.
- 4467 MemberLinks and 1393 WordClassNodes.

General impressions: Glancing at output during run, can see just a handful of disjuncts being peeled of each word, and getting merged. Often just one percent. Sometimes see merges where 2/3rds of the disjunct of one word are merged with just a few percent of the other words. Neither word class seems to be very desirable.

The last few dozen merges were ... curious. They were:

merged class	Comments
line series	synonyms
S Ap	stray markup ?
$ 1.00	tables?
P 1.00	stray markup ?
1 1.00 B	numbers
1.50 1	numbers
† 12mo	tables?
\\\\\\\\\\\\\\"that --	garbage punctuation
S Ag O	Periodic table?
S R C 2	tables?
S 1 4	?
S No W	compass rose points?
Leewit captain	fiction character title?

So these are not bad, but involve a lot of undue attention on stray markup. For the 'line series' merge, the print says:

```
In-group size=2 overlap = 1 of 87 disjuncts, commonality= 1.15%
------ Assign: Merged 3 of 73 sections on ‘line‘ in    0.0 secs
------ Assign: Merged 1 of 17 sections on ‘series‘ in    1.0 secs
```

So although 'line series' seems like a reasonable pair of synonyms, just a small handful of sections are being peeled off of each one. The 'Leewit captain' situation is a little better:

```
In-group size=2 overlap = 3 of 37 disjuncts, commonality= 8.11%
------ Assign: Merged 19 of 25 sections on ‘Leewit‘ in    0.0 secs
------ Assign: Merged 3 of 31 sections on ‘captain‘ in    0.0 secs
```

This has the "unbalanced" feel to it: most of 'Leewit' gets merged into a handful of sections on 'captain', so that's not very impressive.

Things worth exploring.

Great. So what have we actually got? Some questions about the dataset:

- Distribution: size of word-class vs. rank. This was previously examined in "diary part one", page 99, for a different collection of merge algorithms (and earlier, different datasets). Do we still get something similar?
- Above, "size" might mean "number of words in the word-class" or it might mean "number of disjuncts in word-class" or it might mean "number of disjuncts in word-class with count weighting". Oof. It would be nice to have a dashboard for this, instead of lots of manual work.
- Distribution of word-senses. That is, how many words participate in more than one word-class? If they do participate in more than one word-class, what is the weight in each class?

- For words that belong to word-classes, what fraction of their weight remains unassigned to any word-class?
- Distribution of MI of pairs of word-classes. We might hope that this is low, so that different word-classes are different from one another.
- Distribution of self-MI of word-classes. One might hope that this is high, so that the word-classes do not share much in common with other words or word-classes.
- As above, but distribution of MI of pairs consisting of a word-class, and a word.
- Prior to starting the merge, there's an MI between words and disjuncts. I don't recall examining that in detail, before. Then, after the merge, how does this change?

That's a lot of questions. Not clear which ones should be answered first.

Issues with the dataset

There are problems. Upon halt and reload of the dataset, the left-right partial sums are not balanced. Total counts are off by a factor of two. This is a bug. Is it serious?

- Starting dataset was 'run-1-t1234-tsup-1-1-1.rdb'.

From diary part three, page 6, we had this as the starting point:

filename	nwords	ndisjuncts	total pairs	sparsity	obs/pair	MM^T ent
run-1-t1234-tsup-1-1-1	15083	205003	855718	11.819	8.5462	16.352

The above is without shapes. With shapes, it was:

	run-1-t1234-tsup-1-1-1	run-1-t1234-shape
nwords	15083	15083
ndisjuncts	205003	1043583
total pairs	855718	2777968
sparsity	11.819	12.468
total obs		22942644
obs/pair	8.5462	8.2588
MM^T ent	16.352	18.222

But the file 'r9-merge-h.rdb' shows

filename	nwords	ndisjuncts	total pairs	sparsity	obs/pair	MM^T ent
r9-merge-h	16452	1110708	?	14.491	20.836	18.246

The following differences are notable:

- This shows 16452-15083=1369 which is almost but not quite the number of word classes, so wtf?
- The number of disjuncts is 1110708 - 1043583 = 67125 more of them. Seems plausible.
- The reported pair count is inconsistent: 793550.0 817444.0
- Sparsity is about the same as the pre-merge value.
- Observations per pair is probably just wrong.

How does one recompute the reported stats, again? ... by running ‘marginals-mst-shape.scm‘ aka ‘run/3-mst-parsing/compute-mst-marginals.sh‘

Expt-10 Merge exploration (Dec 2021)

The above generated a lot of confusion ... A number of bugs were found and fixed in the merge code, and debugging continues. Once everything seems fully debugged and stable, analysis of merge results will resume in the next part of the diary.

I did try restarting by copying ‘run-1-t1234-tsup-1-1-1.rdb‘ and recomputing marginals. This solved nothing; the marginals (and the similarities) appear to be fine. I’m currently using ‘r9-sim-200.rdb‘ for marginals plus the similarities for the top 200 most frequent words.

Objdump (Dec 2021)

And now for something *completely* different. So, this is a learning system. It should be able to learn “anything”. This section describes a wild-n-crazy idea of pumping disassembly through the system. Just to see what will happen. This might seem irrational, and I don’t have any great hopes for it, but I’m curious to see ... what will pop out.

We’ll pump disassembled elf64 binaries through the system. Disassembly to be done by objdump. It’s cleaned up a bit with split-objdump.pl in the run-common directory. Yes, one could develop a very sophisticated graph of the control flow. This won’t be done, as the short-term goal here is to see what’s possible with very little effort, instead of building in complex *a priori* knowledge about what assembly is.

The End

This is the end of Part Four of the diary. The next part is Part Five.