# June 2019 Restart - Mini Datasets

Linas Vepstas

Version of 19 June 2019

**Abstract**

This reports some very preliminary results on learning grammars. The algorithms being applied have a variety of missing features and imperfections but appear to be usable enough to provide at least some early-stage results. The results ar encouraging. This report brefily reviews the dataset(s), the algorithms used to obtain them, and measurement results.

## Introduction

Work was suspended in the Fall of 2018, and was restarted in June 2019. This is an attempt to pick up where things got left off. It summarizes grammar learning dataset, algorithms and preliminary results, as it stands at the moment.

- The primary dataset, `en_dj_cfive`, appears to be in good condition and should be useable for a good long while, before updates are needed.
- The algorithms are incomplete; much work remains to be done. Work on the core ideas, such as the "sheaf" idea, has not yet started in earnest, although some scaffolding has been laid.
- Existing code is sufficient to generate word-classes, word-sense disambiguation and grammars.

An eyeball inspection of the word-classes suggests that they are quite healthy, but there are no auomated tools to verify thier quality. The clustering seems to be able to split words into word-senses, placing them into correct clusters. The assignments look reasonable, when judged by eye, but there are no automated measurements. These last two results were previously reported in detail in the Winter of 2017 and Spring of 2018. This report focuses on the resulting grammars that can currently be generated, and an automated process to measure thier quality.

To repeat: these are highly preliminary, as most of the important parts of the grammar generation mechanism have not yet been implemented. The results here are a sniff-test or proof-of-concept, to show that the idea works at some minimal level.

## Baseline Dataset

The baseline is the `en_dj_cfive` dataset. Let's recall how this dataset was obtained:

1. All five "tranches" of the training corpora were used to obtain word-pair MI scores. Unlike the earlier rfive datasets, this uses a bug-fixed tokenizer.

2. All five (the same five) tranches were run through the strict-MST parser, to obtain word-disjunct pairs. The disjunct connectors are all single-word connectors. The MST parser was "strict", in that it always created a tree (no loops) that always conected every word in a sentence. Each word-disjunct pair has an associated count, which is the number of times it was seen during MST parsing.

That's it. Starting with this dataset, various subsets were created:

- **en_micro_marg** – Keep only words with observation count>500, sections with count>10; discard all sections that contain connectors that cannot link to anything.

- **en_mini_marg** – Keep only words with observation count>40, sections with count>5; discard all sections that contain connectors that cannot link to anything.

- **en_large_marg** – Keep only words with observation count>8, sections with count>3; discard all sections that contain connectors that cannot link to anything.

- **en_midj_links** – Discard all sections that contain connectors that cannot link to anything. (todo: rename to full-marg)

Recall that each dataset can be thought of as a (very) sparse matrix. Rows of the matrix are words; columns of the matrix are disjuncts. Each entry in the matrix is a word-disjunct pair; it is called a "Section" (a SectionLink in the AtomSpace). Below are the stats from the `(print-matrix-summary-report)` function.

| Size | Secs | Obs'ns | Obs/sec | Sparsity | Entropy | MI | Dataset |
|---|---|---|---|---|---|---|---|
| 1610 x 67K | 184K | 14.7M | 80.3 | 9.20 | 15.17 | 4.60 | en_micro_marg |
| 7385 x 270K | 608K | 20.8M | 34.2 | 11.67 | 16.79 | 4.96 | en_mini_marg |
| | | | | | | | en_large_marg |
| 438K x 23.3M | 31.7M | 69.2M | 2.18 | 18.30 | | | en_midj_links |
| 445K x 23.4M | 31.9M | 69.4M | 2.18 | 18.32 | 23.09 | 8.16 | en_dj_cfive |

The columns are:

- **Size** – Dimensions of the matrix. Number of words x Number of disjuncts. *Notation:* $|w| \times |d|$.

- **Secs** – Total number of Sections. That is, total number of word-disjunct pairs in the matrix. *Notation:* $|(w,d)|$ (*sometimes written as* $|(*,*)|$).

- **Obs'ns** – Total number of observations of Sections. Sum total of how many times the sections were observed. *Notation*: $N(*,*)$.
- **Obs/sec** – The average number of observations per Section.
- **Sparsity** – The log (base two) of the fraction of non-zero entries in the matrix. That is, $\log_2(|w| \times |d| / |(w,d)|)$ where $|(w,d)|$ is the number from the second column.
- **Entropy** – As usual for a matrix: $-\sum_{w,d} p(w,d) \log_2 p(w,d)$. Note that the log-base-two means this is in bits. Here, $(w,d)$ are word-disjunct pairs, *i.e.* sections. The probability is actually a frequency: $p(w,d) = N(w,d)/N(*,*)$.
- **MI** – As usual for a matrix: $-\sum_{w,d} p(w,d) \log_2 p(w,d) / (p(w,*) p(*,d))$.

Note that the MI score drops as words are trimmed from the dataset. All of those infrequently-observed words carry a lot of information, it seems.

These are promising to be a lot cleaner, and stronger than the previous *rfive* dataset.

# Grammars

A grammar is obtained by clustering words into grammatical classes. A variety of different clustering strategies and clustering parameters were used to obtain several grammars. See the files `learn/scm/gram-agglo.scm` and `learn/scm/gram-projective.scm` for details about how clustering is done. Results are reported in multiple tables.

First table: summary report for the first four survey sets. The second table is like the first; it just explores more paramter settings. These are just a rough first-cut, and are not expected to be good, for five reasons:

- These use the cosine-distance metric (I expect the MI metric to be much better).
- Parameters are not tuned (some parameters were picked that seemed reasonable to start with; the second table explores some tuning.)
- Clustering remainders are being ignored. (When a word is placed into a cluster, the counts of any shared disjuncts are transfered to the cluster. This leaves behind a left-over word-vector, containing disjuncts that did not fit into the cluster. This remainder represents a second word-sense for that word; it's part of how WSD happens "automatically" in these algorithms. Currently, these left-over bits are either being reclustered, if possible, otherwise they are being discarded at the end (I think, I'm not sure). They should not be discarded.)
- The disjunct-shapes are not being clustered ("shapes" are disjuncts with wild-cards in them; see elsewhere for a detailed explanation. I expect that cluster quality will improve when shapes are included.)
- Connectors are not being assigned to word-classes (*i.e.* the "sheaves" concept is not being used; the code for the sheaves algo has not yet been written; it needs to be coupled with the shapes code.)

All columns in the tables are obtained from the gram-class matrix report. That is, the matrix-rows are now grammatical classes, not words. The matrix columns are still word-disjuncts, not gram-class disjuncts (this is partly related to the "shapes" problem). A correct grammar would use gram-classes in the connectors as well. So again – this is a rough cut.

The table is much like the above:

- **Size** – Number of grammatical classs x Number of disjuncts. Note that many of these classes are "singleton classes", containing only one word. See the **WC** and **Sing** columns for a breakout of the number of classes with one and more than one word. **WC+Sing** equals the total.
- **Secs** – Total number of Sections. That is, total number of word-disjunct pairs in the matrix.
- **Obs'ns** – Total number of observations of Sections. How many have been observed.
- **Entropy** – As before, just using gram-classes instead of words.
- **MI** – As before, just using gram-classes instead of words.
- **WC** – Number of grammatical classes with two or more words in them.
- **Sing** – Number of singleton classes – that is, classes with only one word in them.

The table below shows only the basic-baseline datasets. There are more, which explore additional parameters.

| Size | Secs | Obs'ns | Entropy | MI | WC | Sing | Dataset |
|---|---|---|---|---|---|---|---|
| 675 x 67K | 142K | 12.0M | 13.75 | 4.03 | 135 | 540 | en_micro_fuzz_exp |
| 416 x 67K | 133K | 11.5M | 13.23 | 3.62 | 124 | 292 | en_micro_discrim |
| 3692 x 269K | 502K | 16.6M | 15.39 | 4.35 | 366 | 3326 | en_mini_fuzz_exp |
| | | | | | | | en_mini_discrim |

The datasets are:

- **en_micro_fuzz_exp** – Created with `(gram-classify-greedy-fuzz 0.65 0.3 4)`. This dataset does not have LEFT-WALL linkages in it. Created from `en_micro_marg`. Here, the paramater `0.65` is the cosine-distance cutoff; vectors with a cosine distance less than this are not clustered. The paramter `0.3` is the broadening parameter: it means that 30% of the counts of disjuncts NOT already in the cluster are added to the cluster. This makes the cluster boundaries "fuzzy", whence the name.
- **en_micro_discrim** – Created with `(gram-classify-greedy-discrim 0.5 4)`. Casual observation shows the clusters are less accurate than above. The parameter `0.5` is the cosine-distance cutoff. The broadening is not fixed;

broadening is accomplished by assigning anywhere from 0% to 100% of the non-shared disjunct counts to the cluster, varying linearly by cosine distance (*e.g.* if the distance is 0.83, then (0.83-0.5)/0.5=66% of the counts will be merged).

- **en_mini_fuzz_exp** – Just like en_micro_fuzz_exp, but with a larger vocabulary, from en_mini_marg.
- **en_mini_discrim** – Just like en_micro_discrim, but with a larger vocabulary, from en_mini_marg.

The table below is much like that above, except that it explores a greater range of learning parameters. It is trying to get a sense of where the sweat spot is.

| Size | Secs | Obs'ns | Entropy | MI | WC | Sing | Dataset |
|---|---|---|---|---|---|---|---|
| 680 x 67K | 143K | 13.2M | 14.08 | 3.95 | 136 | 544 | en_micro_fuzzier |
| 948 x 67K | 155K | 12.6M | 14.23 | 4.29 | 115 | 833 | en_micro_fizz |
| 672 x 67K | 142K | 12.1M | 13.75 | 4.00 | 135 | 537 | en_micro_diss |
| 948 x 67K | 155K | 12.5M | 14.19 | 4.30 | 115 | 833 | en_micro_dissier |
| 1260 x 67K | 168K | 13.7M | 14.66 | 4.46 | 85 | 1175 | en_micro_dissiest |

The datasets are:

- **en_micro_fuzzier** – Created with `(gram-classify-greedy-fuzz 0.65 0.6 4)`. Note the fuzz paramter is 0.6 not 0.3 as before. This means that when a words is being merged into an existing word-class, it will merge in 60% of the non-shared disjuncts from the new word. This will probably harm word-sense disambiguation. However, WSD probably plays a small role, at this time.
- **en_micro_fizz** – Created with `(gram-classify-greedy-fuzz 0.75 0.3 4)`. The fuzz paramter is the same as for en_micro-fuzz, but the discriminator is tighter – 0.75 instead of 0.65.
- **en_micro_diss** – Created with `(gram-classify-greedy-discrim 0.65 4)`. Similar to en_micro_discrim, but more discriminating (0.65 instead of 0.5).
- **en_micro_dissier** – Created with `(gram-classify-greedy-discrim 0.75 4)`. Similar to en_micro_discrim, but more discriminating (0.75 instead of 0.5).
- **en_micro_dissiest** – Created with `(gram-classify-greedy-discrim 0.85 4)`. Similar to en_micro_discrim, but more discriminating (0.85 instead of 0.5).

# Measurements

There does not appear to be any good way to assess the quality of a grammar. So instead, a trick is used that is obviously flawed, but is "better than nothing". The trick

is to compare the grammar to the LG English dictionary. That is, a sentence is parsed, once with the grammar-under-test, and once with the LG English grammar. The parses are compared side-by-side, looking to see if they have the same links between words, or not. This allows both precision and recall to be measured: A linkage is high-precision, if it does not contain link that it shouldn't. A linkage has high recall, if it contains most of the links that it should.

There are obvious problems with this:

- There is no particular reason to beleive that the above algorithms will reproduce LG dictionaries. This is a comparison of apples-to-oranges: the learned dictionaries are obtained by apply probability theory to sparse observational data of strings of words. By contrast, the LG dictionary is hand-curated, by linguists applying innate, common-sense theories of grammatical relationships. The resulting linkages are just opinions expressed by linguists as to what English must be like. That there is significant overlap between statisical results and linguist opinion is itself remarkable.

- The LG English dictionary is imperfect, sometimes creating incorrect parses. Thus, it is possible that the grammar-under-test produced a better parse, while still being scored poorly, because it failed to match LG.

- LG parses in general contain loops, i.e. are not trees. The grammar-under-test might also contain loops, but maybe less than LG. Thus, the grammar-under-test will often have a low recall, simply because of how it was built. Using a non-strict MST at earlier stages might improve the situation.

- Some links are more important than others. It is more important to get links to subject and object correct, than it is to get links to punctuation correct. In particular, LG links to punctuation are rather *ad hoc*, epecially for the end-of-sentence punctuation. It is unlikely that the grammar-under-test will handle punctuation the same way.

- If there is a combinatorial explosion of LG parses, then the parse-scoring system in LG is partially blinded, and the resulting choosen parses might not be the ones with the best scores. It might be possible to ameliorate this with changes to LG.

Thus, due to the above concerns, the actual meaning of the scores is unclear. Never the less, its a good starting point. Some future version of the measurement tool will keep track of subject and object links, and maybe other link types that are considered important.

Currently, all of the dictionaries-under-test are missing a LEFT-WALL marker. I'm not sure why; this will be revisited later. Thus, LEFT-WALL links are not compared, and are not a part of the evaluation process.

The test-corpora contain many words not in the dictionary, and vice-versa: the dictionaries contain many words not in the test-corpora. The overlap between these two vocabularies is shockingly small (but perhaps entirely normal – see extended commentary about Zipf's law, elsewhere). In particular, the "micro" dictionaries have only 1.6K words in them, and of these, less than a third show up in the test corpora. Thus, we have three testing options:

- Skip the evaluation of a sentence, if it contains words that are not in the dictionary-under-test. The test corpora contain a lot of proper names that never appeared in the training corpora, and this alone accounts for many of the sentence rejections.

- Evaluate the sentence anyway; words not in the dictioary will be null-linked (i.e. have no links going to them.) Mostly all that happens is that precision scores are mostyly unaffected, while recall scores are obviously lower.

- Perform unknown-word-guessing. This is not hard to do, and ultimately is a required ability for any dictionary usable in the real world. The current guessing strategy is to try each unknown word with each gram-class, and then let the parse-scoring system pick the one with the highest score. Note that unknown-word guessing leads to a combinatoric explostion in LG, which can cause extremely long parse-times (hours!); see notes below. Combinatoric explosions can also result in sub-optimal choice of linkage. The determination of the pertinance of this effect is TBD.

Two sets of dictionaries were created: with and without unknown words. Results for both are reported below.

Comparisons were performed for five different corpora. These are also an imperfect choice, but adequate for this intitial rough-cut evaluations. The are:

- **CDS** – The "Child Dircted Speech" corpus from the ULL project. Contains 1837 sentences uttered by parents towards children. These have a characteristically small vocabulry, and a simple grammatical structure. This corpus consistently results in the highest scores during testing, for all test dictionaries.

- **basic** – The `corpus-basic.batch` file from LG. This contains 1000 sentences, of which 421 are intentionally ungrammatical, leaving 579 valid English sentences. These contain a balanced selection of a wide variey of different kinds of English constructions; it was meant to showcase the rich variety of different sentences that LG can handle. (Note, however, that evolving fixes means that not all of these parse correctly).

- **fixes** – The `corpus-fixes.batch` file from LG. This contains 4236 sentences, illustrating fixes to the early versions of the LG dictionary. Although there is a wide variety of different kinds of English constructions, it's not particularly "balanced", and the sentences tend to be short; just long enought to illustrate some phenomenon.

- **gold** – The "Golden Corpus" from the ULL project. This contains 229 sentences. Note that the parses provided by the ULL project are NOT used for evaluation. The LG parses are used instead.

- **silver** – A subset of the "Silver Corpus" from the ULL project. This contains 2513 sentences, taken from three different Project Gutenberg books: "Kilmeny of the Orchard" by Lucy Maud Montgomery; "Peter Rabbit" by Thornton W Burgess; and a portion of "Under the Lilacs" by Louisa May Alcott.

All five corpora are available in the `learn/run/3-gram-class/test-data` directory.

The table below shows results. The columns in the table are as follows:

- **Sents** – The total number of sentences in the corpus. This repeats the number given above.
- **Skip** – Percentage of total sentences that were skipped, because they contained unknown words. (Reported only when unknown-word guessing is disabled.)
- **Parsed** – The number of sentences that were evaluated. Equal to number of sentences times (100% - Skip).
- **Diff** – The number of sentences that had parses that differed from the first parse given by the English dictionary in Link Grammar version 5.6.1.
- **Words** – The total number of words in the evaluated sentences. This varies from dictionary to dictionary, depending on whether sentences were skipped or not.
- **Vocab** – The size of the vocabulary in the evaluated sentences; that is, the number of unique words. When rejecting sentences with unknown words, this becomes the intersection of the vocabulary of the accepted sentences and the dictionary-under-test.
- **Links** – The number of links that the LG English parse generated. This varies from dictionary to dictionary, depending on whether sentences were skipped or not.
- **P** – Precision of links: TP/(TP+FP) where TP is the count of the links that both dictionaries generated, and FP is the count of the links that the dictionary-under-test generated, but LG did not.
- **R** – Recall of the links: TP/P where TP as above, and P the total number of links produced by the LG English parse.
- **F1** – The harmonic mean of precision and recall.

Results for the *micro-fuzz* dataset. This is a tiny dataset, with only 1600 words in it; its the first shot at anything even vaguely workable. Not expecting good results. But they seem passable anyway. Most sentences got skipped, because they contained unknown words, but the ones that did parse seem to not be outrageous. Notice the intersection of the test-corpus vocabulary and the dictionary size is small – about 1/3rd of the dictionary size, or less. That is a small vocabulary; so 2/3rds of the dictionary contains words that are not in the test-corpus!

| | micro-fuzz | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Sents | Skip | Parsed | Diff | Words | Vocab | Links | P | R | F1 |
| CDS | 1832 | 69% | 562 | 86% | 2738 | 176 | 2376 | 0.878 | 0.586 | 0.703 |
| basic | 579 | 86% | 82 | 99% | 666 | 209 | 650 | 0.842 | 0.499 | 0.626 |
| fixes | 4236 | 82% | 776 | 96% | 4901 | 591 | 4360 | 0.752 | 0.487 | 0.591 |
| gold | 229 | 90% | 24 | 100% | 169 | 95 | 156 | 0.802 | 0.571 | 0.667 |
| silver | 2513 | 85% | 377 | 100% | 3323 | 527 | 3225 | 0.752 | 0.497 | 0.599 |

Results for *micro-discrim* below. The gram classes are looser (the cutoff is lower -0.5 instead of 0.65 as above), and casual inspection suggests that they are lower quality (its clear that many unlike words are being classed together). This behaves in a not-too-surprising way: precision drops (more junk links produced), while recall improves (the junk links just happen to go to the right places). Overall F1 score is better, which is surprising. Same base vocab of 1600 words. The values in the Sents, Skip, Parsed, Words, Vocab, Links columns are unchanged, since both this and above are built on the *en_micro_marg* dataset.

| | micro-discrim | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Sents | Skip | Parsed | Diff | Words | Vocab | Links | P | R | F1 |
| CDS | 1832 | 69% | 562 | 82% | 2738 | 176 | 2376 | 0.838 | 0.665 | 0.7421 |
| basic | 579 | 86% | 82 | 98% | 666 | 209 | 649 | 0.820 | 0.569 | 0.672 |
| fixes | 4236 | 82% | 776 | 96% | 4901 | 591 | 4359 | 0.717 | 0.559 | 0.628 |
| gold | 229 | 90% | 24 | 100% | 169 | 95 | 156 | 0.758 | 0.607 | 0.674 |
| silver | 2513 | 85% | 377 | 99% | 3323 | 527 | 3218 | 0.730 | 0.590 | 0.652 |

Results below for *mini-fuzz*. This dictionary has a much larger vocabulary: 7385 words instead of 1600 as above. Despite the much larger vocabulary, there is not much change in F1: it is just a hairs-breadth better than the micro version, except for *ull-golden*, which is markedly worse, and *corpus-basic*, which is better. Precision ad recall are very nearly unchanged, except for *ull-golden* and *corpus-basic*. The big difference is that considerably fewer sentences get skipped due to unknown vocabulary (but still more than half). The mini dictionaries contain 7385 words, and at most 1/5th of them appear in the test corpora. The lack of overlap is remarkable!

| | mini-fuzz | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Sents | Skip | Parsed | Diff | Words | Vocab | Links | P | R | F1 |
| CDS | 1832 | 53% | 850 | 87% | 4222 | 238 | 3656 | 0.878 | 0.592 | 0.707 |
| basic | 579 | 59% | 237 | 99% | 2000 | 478 | 1927 | 0.831 | 0.531 | 0.648 |
| fixes | 4236 | 57% | 1834 | 97% | 12694 | 1463 | 11468 | 0.761 | 0.491 | 0.597 |
| gold | 229 | 72% | 64 | 100% | 449 | 228 | 407 | 0.754 | 0.460 | 0.571 |
| silver | 2513 | 62% | 943 | 100% | 9859 | 1308 | 9716 | 0.750 | 0.501 | 0.601 |

Below for *mini-discrim*. Working on it ...

| | mini-discrim | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Sents | Skip | Parsed | Diff | Words | Vocab | Links | P | R | F1 |
| CDS | 1832 | | | | | | | | | |
| basic | 579 | | | | | | | | | |
| fixes | 4236 | | | | | | | | | |
| gold | 229 | | | | | | | | | |
| silver | 2513 | | | | | | | | | |

# Unknown words

How about automated unknown words? The tactic here is to insert an unknown-word marker, pointing to each gram-class with two or more members. Basically, unknown words are guessed at by trying to assign them to each gram class – the disjunct with the best-possible fit (highest class-disjunct MI) is used. The dictionaries are the same as those above, except that UNKNOWN-WORD is added to them.

Processing is noticably slower; there appears to be an combinatoric explosion - a runtime of $O\left(N^k\right)$ for $k$ unknown words in the sentence; the $N$ is presumably the total of all disjuncts in all candidate classes, as each is tried against the next as a possible match. Sentences with 3,4,5 unknown words can take *hours* to parse. Hours is really bad. We are patient, here, but clearly, something needs to be done to avoid the bog-down.

We except 100% of the sentences to be covered, so this is a big change from before. Two columns from earlier tables are ommitted: 100% of the test-corpus is parsed and compared. Since LG will run on every sentence in each test-corpus, the total number of observed words, and the total known-vocabulary will be the same for all of the different dictionaries-under-test. These are resported just once, below.

| Dataset | Sents | Words | Vocab | Links |
|---|---|---|---|---|
| CDS | 1832 | 9274 | 300 | 7957 |
| basic | 579 | 5398 | 1201 | 5222 |
| fixes | 4236 | 32078 | 4368 | 28939 |
| gold | 229 | 1895 | 782 | 1766 |
| silver | 2513 | 30119 | 3663 | 29781 |

Here are the results for *micro-fuzz*, with unknown-word-guessing enabled. Recall, the peramaters were `(gram-classify-greedy-fuzz 0.65 0.3 4)`. Compared to the same dataset without guessing, precision drops a bit, (no surprise), recall improves a lot, enough that the F1 improves a lot, with the exception of the gold-corpus for which F1 stays the same. Conclusion: adding unknoown-word guessing vastly improves coverage, and also helps recall and F1, with just a bit of damage to precision.

| | micro-fuzz-unk `(fuzz 0.65 0.3)` | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 76% | 0.854 | 0.699 | 0.769 |
| basic | 579 | 97% | 0.789 | 0.624 | 0.697 |
| fixes | 4236 | 96% | 0.724 | 0.595 | 0.653 |
| gold | 229 | 100% | 0.721 | 0.608 | 0.660 |
| silver | 2513 | 100% | 0.719 | 0.597 | 0.652 |

For *micro-fuzzier-unk*: almost no difference at all from the above. Note that both have almost exactly the same number of grammatical classes (135 *vs.* 136) and so unknown-word guessing seems to be unaffected.

| | micro-fuzzier-unk `(fuzz 0.65 0.6)` | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 77% | 0.853 | 0.698 | 0.768 |
| basic | 579 | | | | |
| fixes | 4236 | | | | |
| gold | 229 | 100% | 0.722 | 0.607 | 0.660 |
| silver | 2513 | | | | |

For *micro-fizz-unk*: Precision went up a little, recall dropped much more. Not unexpected: the tighter discriminator makes each grammatical class more accurate, but the overall coverage (the recall) drops. There are two possible explanations for this differrence:

1. The tighter discriminator is leading to more accurate grammatical classes (?)
2. A smaller number of grammatical classes to choose from suggests that unkown word guessing might struggle more.

Disentangling these effects is not obvious.

| | micro-fizz-unk `(fuzz 0.75 0.3)` | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 81% | 0.870 | 0.636 | 0.735 |
| basic | 579 | 98% | 0.797 | 0.576 | 0.669 |
| fixes | 4236 | 96% | 0.729 | 0.556 | 0.631 |
| gold | 229 | 100% | 0.727 | 0.578 | 0.644 |
| silver | 2513 | 100% | 0.722 | 0.553 | 0.626 |

Here is for *micro-discrim-unk*; it should be compared to *micro-discrim*, which uses the same algo and paramaters, but doesn't have word-guessing built in.

Prelim, incomplete data: word-guessing helps ...

| | micro-discrim-unk (discrim 0.5) | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 73% | 0.833 | 0.741 | 0.784 |
| basic | 579 | | | | |
| fixes | 4236 | | | | |
| gold | 229 | 100% | 0.704 | 0.643 | 0.672 |
| silver | 2513 | | | | |

Here is *micro-diss-unk*. It should be compared to two different dictionaries: *micro-discrim-unk*, immediately above, which uses the same clustering algo, but a weaker cutoff, also to *micro-fuzz-unk*, which uses a different clustering algo, but the same cutoff. Note that when the cutoff is the same, both produce the same number of grammatical classes (135, either way). The looser cutoff produces fewer classes (124); presumably each class is larger. This does not resolve the question of whether it is the sharpenss of the classes that matter, or thier raw number. Perhaps these effects cannot be disentangled...

Based on prelim incomplete numbers: the algo don't matter, the discrim does.

| | micro-diss-unk (discrim 0.65) | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 77% | 0.853 | 0.699 | 0.769 |
| basic | 579 | 97% | 0.791 | 0.621 | 0.696 |
| fixes | 4236 | 96% | 0.724 | 0.593 | 0.652 |
| gold | 229 | 100% | 0.725 | 0.611 | 0.663 |
| silver | 2513 | 100% | 0.721 | 0.596 | 0.652 |

Here's *micro-dissier*: it can be compared to the above, which uses a weaker cutoff, or to *micro-fizz-unk*, which uses the same cutoff.

| | micro-dissier-unk (discrim 0.75) | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 83% | 0.864 | 0.630 | 0.728 |
| basic | 579 | 98% | 0.795 | 0.574 | 0.667 |
| fixes | 4236 | | | | |
| gold | 229 | 100% | 0.715 | 0.567 | 0.632 |
| silver | 2513 | 100% | 0.726 | 0.555 | 0.629 |

Here's dissiest:

| | micro-dissiest-unk (discrim 0.85) | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 82% | 0.893 | 0.591 | 0.711 |
| basic | 579 | 98% | 0.792 | 0.501 | 0.614 |
| fixes | 4236 | 97% | 0.741 | 0.500 | 0.597 |
| gold | 229 | 100% | 0.703 | 0.509 | 0.591 |
| silver | 2513 | 100% | 0.730 | 0.492 | 0.588 |

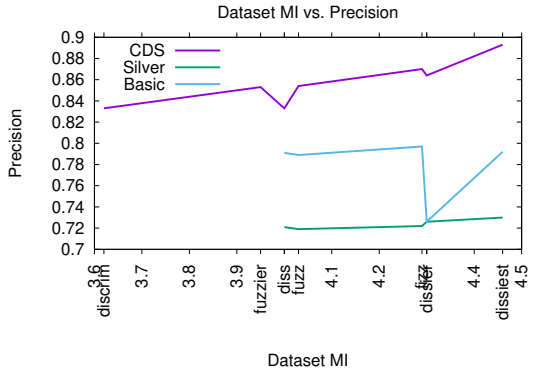
Below is *mini-fuzz-unk*. It is to be compared to *micro-fuzz-unk*, which uses the same parameters, but has a smaller vocabulary.

it appears that ... mixed results ...

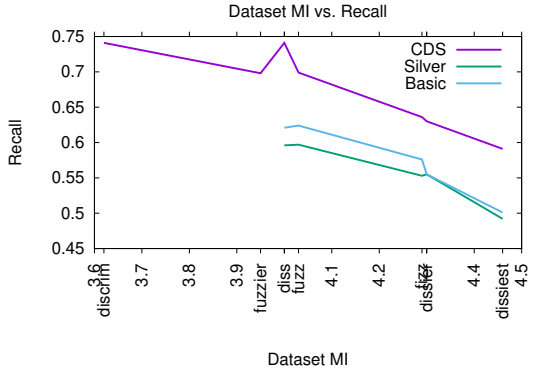
| | mini-fuzz-unk (`fuzz 0.65 0.3`) | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 78% | 0.867 | 0.676 | 0.759 |
| basic | 579 | | | | |
| fixes | 4236 | | | | |
| gold | 229 | 100% | 0.718 | 0.538 | 0.615 |
| silver | 2513 | 100% | 0.729 | 0.559 | 0.633 |

# Graphs

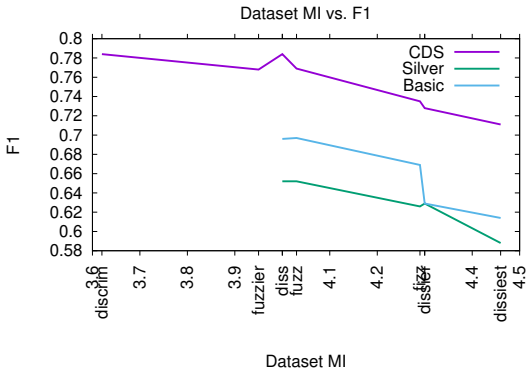
Reading the above tables is hard. It is easier to see what is going on with graphs. Here. The first figure shows precision as a function of dataset MI for three of the corpora (there is some missing data, still coming in.)




The one below shows recall as a function of dataset MI.



Finally, the reciprocal mean of the two, aka the F1-score. Judging from the slope, its now clear that the loss in recall dominates the gain in precision.




# ULL Dictionary Measurements

Same proceedures as above, applied to the ULL dictionaries. These are provided by Anton Kolonin and his crew, and are generated using completely different software pipeline, described elsewhere.

- ull-lgeng – Based on LG-English parses: obtained from http://langlearn.singularitynet.io/data/aglushchenko_parses FULL-ALE-dILEd-2019-04-10/context:2_db-row:1_f1-col:11_pa-col:6_word-space:discrete/

| | ull-lgeng | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 93% | 0.951 | 0.335 | 0.496 |
| basic | 579 | 100% | 0.945 | 0.157 | 0.269 |
| fixes | 4236 | 98% | 0.942 | 0.145 | 0.251 |
| gold | 229 | 100% | 0.982 | 0.623 | 0.763 |
| silver | 2513 | 100% | 0.957 | 0.582 | 0.724 |

- ull-sequential – Based on "sequential" parses: obtained from http://langlearn.singularitynet.io/data/aglushchenko_p FULL-SEQ-dILEd-2019-05-16-94/GL_context:2_db-row:1_f1-col:11_pa-col:6_word-space:discrete/

| | ull-sequential | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 99% | 0.898 | 0.080 | 0.147 |
| basic | 579 | 100% | 0.932 | 0.045 | 0.086 |
| fixes | 4236 | 99% | 0.839 | 0.164 | 0.274 |
| gold | 229 | 100% | 0.788 | 0.608 | 0.687 |
| silver | 2513 | 100% | 0.786 | 0.601 | 0.681 |

- ull-dnn-mi – Based on "DNN-MI-lked MST-Parses": obtained from http://langlearn.singularitynet.io/data/aglushch GUCH-SUMABS-dILEd-2019-05-21-94/GL_context:2_db-row:1_f1-col:11_pa-col:6_word-space:discrete/

| | ull-dnn-mi | | | | |
|---|---|---|---|---|---|
| Dataset | Sents | Diff | P | R | F1 |
| CDS | 1832 | 95% | 0.900 | 0.442 | 0.593 |
| basic | 579 | 100% | 0.904 | 0.262 | 0.406 |
| fixes | 4236 | 98% | 0.837 | 0.263 | 0.400 |
| gold | 229 | 100% | 0.728 | 0.547 | 0.624 |
| silver | 2513 | 100% | 0.723 | 0.532 | 0.613 |

# Conclusions

Conclusions TBD. Here's what seems to be happening so far:

- The tighter the clustering, the higher the precision. But the relationship is fairly weak, and is more than offset by loss of recall.
- The ULL precision is high – better than what I've got; recall is poor, considerably worse. Need to explore why; I don't understand
- The ULL dictionary is brought down by low recall. Root cause is unclear, needst to be explored. Perhaps it is missing vocabulary plus no unknown-word mechanism. Need to double-check what is going on here.

More examination and conclusions pending.

# The End