cTakes Dictionary Lookup Module 2 Statistics

3-04-2014

# Dictionaries

## Dictionaries

Four variations a dictionary with the standard cTakes UMLS terms were used for speed trials. Each Dictionary variation encompassed Snomed CT Terms, non-Snomed CT synonym (by CUI) Terms, and RXNorm ~ Orangebook Medication Terms. The four dictionaries varied in indexing approach, pre-filtration of Orangebook medications, and “cleanup” of Terms. All dictionaries were in Hsqldb cached table format, except for the Orangebook Lucene indexes used in the non-prefiltered drug dictionaries. All dictionaries were generated from UMLS 2011ab sources. All dictionaries only contained Terms for cTakes-relevant TUIs, those being the TUIs belonging to the standard cTakes semantic groups.

The two indexing approaches used were First Word Indexing, as used by the previous dictionary lookup module, and Rare Word Indexing, as used by the new dictionary lookup module. For details on Rare Word Indexing, see the Dictionary Lookup Help documentation.

The two First Word Index dictionaries did not have RX Norm terms pre-filtered by Orangebook, but instead the Orangebook resided in a separate Lucene index and filtering was done at run-time. This is the approach taken by the previous dictionary lookup module. The two Rare Word Index dictionaries did have medications pre-filtered by orangebook as this is the approach taken by the new dictionary lookup module.

Two variations of Term text filtration were used for both index type dictionaries. For information on Term text filtration (“cleanup”), see the Dictionary Creator Tool documentation.

|  |  |  |
| --- | --- | --- |
| **Name** | **Index Type** | **Filtered Term Text** |
| FN | First Word | No |
| FY | First Word | Yes |
| RN | Rare Word | No |
| RY | Rare Word | Yes |

Note that even the –worst- dictionary used (FN) is cleaner and smaller than the 2011ab UMLS dictionary standard used by cTakes as it only contained Terms from Snomed CT and only for standard TUIs used by cTakes. The result is that the Hsqldb dictionary used for this trial had 1,544,007 rows as opposed to the cTakes standard dictionary’s 2,848,821 rows, not including medications.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Rows (millions) | Texts (millions) | Cuis (millions) |
| **Unfiltered** | 1.5 | 1.5 | 0.7 |
| **Filtered** | 0.5 | 0.5 | 0.2 |

# Annotators

## First Word Lookup

For the tests using the First Word Index dictionaries, the previous dictionary lookup module was used, with the standard First Word annotator configuration used for UMLS dictionary lookup.

## Rare Word Lookup

For the tests using the Rare Word Index dictionaries, the new dictionary lookup module was used with the default Rare Word annotator configuration used for UMLS dictionary lookup.

Additional Tests were run using other Rare Word annotator configurations, but those will be detailed in the Accuracy Tests section.

# Speed Tests

## Sharp Corpus

Tests were run on 284 notes of the Sharp corpus. The notes chosen had gold annotations available, which were later used for analysis of the accuracy of the dictionary lookup annotators.

## Share Corpus

Tests were run on 301 notes of the Share corpus. The notes chosen had gold annotations, and though there were presently some annotation errors that made them suboptimal for accuracy analysis, the difference in note type from the Sharp corpus makes them useful in speed analysis.

## System Configuration

All tests were run on the same Linux node while it was otherwise idle. Tests were performed using the Collection Processing Engine (CPE) GUI tool to run the standard clinical pipeline, reading and writing data on disk. All times reported here are those reported by the CPE. As it was not necessary for these tests, the Assertion Module (which adds significantly to run time) was removed from the pipeline.

## Corpus Processing Time

Sharp Total Processing Time Share Total Processing Time

## Note Processing Time

Sharp Processing Time Share Processing Time

## Summary

Running with the smaller dictionary (Y variants) was faster than running with the larger unfiltered dictionary (N variants). In addition, the new annotator (R) was much faster than the previous configuration (F). The best results were obtained by running the filtered dictionary with the new dictionary lookup module (RY), running at 0.43 seconds per note for Sharp and 0.71 seconds per note for Share.

# Accuracy Tests

## Sharp Corpus

Tests were run on 284 notes of the Sharp corpus. The notes chosen had gold annotations available for all cTakes semantic groups, which made them perfect for analysis of the accuracy of the dictionary lookup annotators. To ensure adherence to the cTakes groups, a filtration step was performed to remove a small number of annotations that were of semantic types that aren’t used by the standard cTakes configuration.

## Share Corpus

Because there were some annotation errors at the time that the tests were performed, the gold annotations for Share were not usable for accuracy analysis.

## System Configuration

All tests were run on the same Linux node while it was otherwise idle. Tests were performed using the Collection Processing Engine (CPE) GUI tool to run the standard clinical pipeline, reading and writing data on disk. All times reported here are those reported by the CPE. As it was not necessary for these tests, the Assertion Module (which adds significantly to run time) was removed from the pipeline. The same runs that were used for speed tests generated the output that was used for these accuracy tests.

## Analysis

For comparison with the gold annotations, only the longest overlapping span of any cTakes semantic group was considered, allowing for evaluation of the “most specific” terms discovered by each configuration. For further information on what this (most specific term) means, see the dictionary lookup help documentation.

## Span Match

Only exact span matches were considered, not overlapping span matches.

## Term Match

Because of the lack of Word Sense Disambiguation, multiple Terms could exist at a single span. For instance, the text “blister” exists for three different Terms, each with a unique CUI, two being findings and one being a disorder. For such spans, if there is an exact match between the gold annotation span and the system span and one of the system CUIs is equal to the gold annotation CUI then there is a Term Match at that span. Unmatched CUIs are then ignored for that span.

## Discoveries, Matches

## Accuracy

## Summary

All dictionary and annotator combinations discover more spans, CUIs, and Terms than existed in the gold annotations. The smaller, cleaner dictionaries actually enabled both annotators to find more spans than the larger unfiltered dictionaries, most likely because the unfiltered dictionary term texts contained tokens that did not appear in the note text.

The number of matched spans and terms also improved when using the smaller dictionaries, as it improved when using the new dictionary lookup annotator. In all, the filtered dictionary with the new dictionary lookup annotator had the greatest number of spans and terms that matched the gold annotations.

Obviously, both Span and Term Recall followed the trend set by the number of matches achieved per combination. The Precision and overall F1 scores were more greatly affected by the sheer number of discoveries (or lack thereof) by each dictionary and annotator combination, with the combination of the unfiltered large dictionary and the First Word annotator (FN) getting the best overall scores.

The baseline tests being complete, more tests were run using different configurations of the new dictionary lookup module.

# Accuracy Tests, New Dictionary Lookup Module

## Dictionaries

For the following tests, only the small filtered dictionary was used.

## Annotator Configurations

In addition to the baseline “RY” configuration, configurations were run with the overlap annotator to find discontiguous spans, Sentence lookup windows, and minimum lookup token span lengths of 3 characters. It should be noted that the speed differences between the fastest and slowest of these configurations was roughly 0.05 seconds per note.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Index Type** | **Match Allowance** | **Lookup Window** |
| FY | First Word | -Overlap- | Noun Phrase |
| RY | Rare Word | Exact | Noun Phrase |
| RY\_O | Rare Word | Overlap | Noun Phrase |
| RY\_S | Rare Word | Exact | Sentence |
| RY\_SO | Rare Word | Overlap | Sentence |

## Discoveries, Matches

Below are the statistics for matches with no minimum span length. In other words, single-letter tokens are candidates for Terms (such as abbreviations).

**Accuracy**

## Discoveries, Matches

Below are the statistics for matches with a minimum span length of three characters. In other words, single-letter and double-letter tokens are not candidates for Terms. For proper comparison, all single and double letter annotations made by the previous dictionary lookup configuration (FY) were culled from its discoveries. The same was not done to the Sharp gold annotations; all gold annotations were kept.

**Accuracy**

## Summary

Using Sentence as the lookup window appears to increase the F1 score by roughly 1 point. Use of overlap span matching does not increase the score as much when compared to exact span matching, but it does seem to make a difference. The combination of Overlap matching with Sentence lookup windows slightly outperforms the rest of the combinations.

In all configurations using a minimum span length of 3 characters improved F1 scores by 2 to 3 points. It should be noted that there were over 200 gold annotations with span lengths less than three characters, so using a minimum span length is not a loss-less operation.

Improvement of the best configuration over the worst for the new dictionary lookup is roughly 4 points. Improvement over the previous dictionary configuration (FY) F1 score is over 7 points.

# Speed Test, Memory Resident Dictionary

## Hsqldb

Because the filtered dictionary is under 65MB in size, it is a valid candidate for use as an in-memory database table with Hsqldb. In addition to the data being in RAM instead of cached on disk, Hsql in-memory tables have the additional speed advantage that they are stored and referenced as POJOs, and no data translation is necessary during lookup.

Loading the database table into memory on pipeline initialization is almost instantaneous.

The test was run on the 284 note Sharp corpus using the simple RY configuration. Once again, the Assertion module was not run in this pipeline.

As can be seen in the table below, when running with this configuration the time spent in dictionary lookup becomes almost negligible.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Total Corpus Seconds** | **Seconds per Note** | **Notes per Second** | **Notes per Hour** |
| Annotator | 2.27 | 0.0080 | 125.28 | 450992.5 |
| Full Pipeline | 94.84 | 0.3340 | 2.99 | 10780.5 |

# Rare Word Part of Speech

## Verbs

When the Rare Word index is created the candidacy of a word is limited by its part of speech. Unfortunately, verb parts of speech are not accounted for. This means that the word for a term in the dictionary index may be a verb. If verbs are excluded parts of speech for lookup tokens (true by default) then the term will not be identified by the annotator.

One possible solution to this problem is to allow verbs to be used as lookup tokens. Doing so with the previous dictionary lookup module would have made a drastic speed difference, but the same is not true for the new module.

A test was run on the Sharp corpus and gold annotations allowing verbs to be lookup tokens, Sentence as a lookup window, no span lengths smaller than 3 characters, and exact span matching (RY\_S\_V). Though the number of spans and terms matching the gold annotations was greater with verbs than without, there were overall more spans, CUIs, and terms discovered which drove the F1 scores down over 1 point.

## Discoveries, Matches

**Accuracy**

# Further Consideration of Precision

## Exact Span Matching

One way to look at exact span matching is that if the span in the note (with variants) matches text in the dictionary, then the match must be correct. In other words, when using exact span match every discovery is a true positive. Pinning the true positive to the number of finds, the precision becomes 1. The number of false negatives can be used to calculate recall, and some very impressive scores arise.

## Accuracy

The charts below compare scores for exact span longer than 3 character discoveries when including verbs as lookup tokens with no assumption of exact precision (RY\_S\_V) and with assumption of exact precision (RY\_S\_V\_1).