Substring Tutorial

Substring-B is part of the SubString package which consists of two separate modules (substring-A and substring-B). Substring-B is a set of Unix shell scripts designed to work on a range of simpler n-gram list formats (where n-grams are not indexed to their occurrences in a specific corpus). Such lists are produced by the NGramProcessor, the Ngram Statistics Package and various other tools and sources such as Google N-Grams. The algorithm implemented by substring-B is described in detail in Buerki (2017).

A. Substring-A module

Typically, substring-A would be used as a further filtering stage in a MWE extraction procedure using the mwetoolkit. An example procedure, using the data supplied in the test_data-A directory is shown below. It is assumed that mwetoolkit 3 is installed. Substring-A enters at step 6 below (but if mwetookit is installed, steps 1 to 5 can be followed as well):

1. Open a terminal and change the present working directory to test_data-A: `cd test_data-A`
2. Index corpus using mwetoolkit: `index.py -v --from PlainCorpus -i index-enwiki/enwiki enwiki.txt`
3. Extract MWE candidates using mwetoolkit:
   `candidates.py -n 2:9 -v -S --corpus-from=BinaryIndex index-enwiki/enwiki.info > wikicand.xml`
4. Add frequencies of a corpus (needed for next step) using mwetoolkit:
   `counter.py -v -sg -i index-enwiki/enwiki.info wikicand.xml > wikicand-count.xml`
5. Apply frequency filter using mwetoolkit:
   `filter.py -v -t enwiki:3 wikicand-count.xml > wikicand-count-f2.xml`
6. Frequency-consolidate list using this script:
   `substring-A.py wikicand-count-f2.xml > wikicand-consolidated.xml`

Note that in order to use the auxiliary scripts on the output of the substring-A module, the output should be specified to be in NGP-format, which can be specified as follows:

```
substring-A.py --to NGP wikicand-count-f2.xml > wikicand-consolidated.txt
```

`substring-A.py` offers a number of further processing options to modify its behaviour. These can be displayed by running `substring-A.py -h` and are shown below:

```
   --to <output-filetype-ext>
Convert input to given filetype extension.
(By default, keeps input in original format):
* "ARFF": ARFF file type, as defined by the WEKA toolkit
* "CONLL": CONLL tab-separated 10-entries-per-word
* "CoreNLP": An XML in CoreNLP format
* "CSV": Tab-separated CSV filetype format, one field per column
* "cupt": Cupt tab-separated 11-entries-per-word
* "DiMSUM": DiMSUM CONLL-like format (with supersenses)
* "GloVe": GloVe format (headerless word2vec)
* "HTML": Pretty HTML for in-browser visualization
* "MinimanticsProfile": Minimantics profile format
* "Moses": Moses factored format (word=f1|f2|f3|f4)
* "NGP": Format of the N-Gram Processor, n-gram + frequency (n·gram·  78)
* "NSP": Ngram Statistics Package-like format (n<>gram<>78)
* "PALAVRAS": PALAVRAS parser format
* "PlainCandidates": One multi_word_candidate per line
* "PlainCorpus": One sentence per line, with multi_word_expressions
* "pWaC": WaC parsed format
* "RASP": RASP parser format with grammatical relations
* "TaggedPlainCorpus": One sentence per line with <mwepart>tags</mwepart>
* "TextualPattern": Human-readable MWE-pattern format
* "TreeTagger": 3-field tab-separated format output by TreeTagger
* "UCS": UCS 2-gram filetype format
* "word2vec": word2vec textual format
* "XML": An XML in mwetoolkit format (dtd/mwetoolkit-*.dtd)

-z OR --retainzero
Retain candidates with zero frequency after frequency consolidation

-l OR --logfreqconsol
Log the entities whose frequencies were adjusted during consolidation
The log will be sent to standard out and also to a file named 'candidates_updated.xml'
in the current directory.

-V OR --version
display the version number of the script

-v OR --verbose
Print messages that explain what is happening.

-D OR --debug
Print debug information when an error occurs.

-h OR --help
Print usage information about parameters and options
```

B. Substring-B module

If an icon was generated on the desktop (or in the applications folder) during installation, double-click on it and Substring-B's main menu should appear.

If you chose not to create a launch icon at installation, Substring-B can be started from a terminal by typing `substring-B.sh` and pressing ENTER. This should also bring up the main menu.

The main menu, as shown below, presents three options:

```
      SUBSTRING
      substring reduction and frequency consolidation tool
      version 1.0



      Please choose a function

      (A) consolidate n-gram list using alogrithm A
      (B) consolidate n-gram list using algorithm B
      (T) assess accuracy of a consolidated list
      (X) exit

     >
```

N-gram lists processed by Substring-B must be formatted in the correct format (see below, or the `test_data` folder for examples). N-gram lists produced by the N-Gram Processor are suitable for direct input; if n-gram lists were produced by a different programme they might need reformatting. N-gram lists to be consolidated by algorithm B need to conform to the following conditions:

1. only one length of n-gram per input list (i.e. 2-grams must be in a separate input file from 3-grams from 4-grams, etc.).
2. minimally 2 input lists must be provided
3. it is strongly recommended that n-gram lists do not contain n-grams across sentence boundaries
4. a 2-gram list must always be provided, regardless of what other n sizes are provided
5. n-gram lists must be consecutive in size, i.e. substring-processor.sh does not consolidate 2-grams with 4-grams directly, but would in this case require a list of 3-grams to be supplied as well.

Additionally, input lists must be of the following format

n<>gram<> 1[1]

or

n·gram· 1[1]

That is, an n-gram (with constituents either delimited by diamonds (<>) or the unicode character interpunct (middle dot)), followed by a tab and the frequency count, optionally followed by another tab and a document count.

To run a frequency consolidation on test data, follow the steps below:

1. Choose `B` and press ENTER to start the substring reduction and frequency consolidation procedure. The programme will ask for a folder with n-gram lists to be dropped onto the window. This folder must contain at least two correctly formatted lists of n-grams. It must not contain any other lists or other files.
2. Inside the Substring-B directory, you will find a folder called `test_data-B`. Drag the `example1` folder into the window and press ENTER. The `example1` folder contains lists of n-grams corresponding to those of example (1)a, above.
3. Confirm that the presented list of files contains all the files and only the files that need to consolidated.
4. The programme asks for a minimum frequency to be entered. For this example, we choose the lowest possible value, 3.

5. After some processing, the programme asks the user to specify whether length-adjustment should be applied. This is fully explained in the next section under 'LENGTH-ADJUST.SH', but is generally recommended, so enter `y` and press ENTER.
6. The programme then asks if the standard lexico-structural filter for English should be applied. Again, this is generally recommended as it filters out a number of n-grams that are less likely to be of interest to linguists. A pure substring reduction and frequency consolidation, however, does not require filtering.
7. When the task completes, the programme asks if the user would like to open the output directory. Answer with `y`.

The output directory (which will be the same folder as originally supplied, now containing the input as well as output lists) now holds the consolidated list. The output list should show the n-grams as in example (1)b, above, except that the the n-gram lovely·time is missing since it has a frequency below the minimum frequency of 3 that was entered in step 2.

A larger amount of data are contained in the folder `example2` within the test data folder. To substring reduce and consolidate these data, again choose `B` from the main menu and follow the same steps as above. As a suggestion, a minimum frequency of 9 could be entered and again a consolidated and filtered list (if the filter was chosen) will appear in the folder after the process has concluded. There is also log (named `correction_log.txt`), detailing the adjustments made by the length adjustment algorithm (if chosen), and a list of called `neg-freq` which lists items for which the consolidation procedure failed to yield accurate frequencies (due to not having access to the underlying text, as discussed above).

To assess the accuracy of an extracted and consolidated list of n-grams, choose `T` from the main menu. The user is prompted to drop a consolidated n-gram list into the window. Next, the user is prompted for a sample size. A random sample of the size indicated will then be taken of the list provided. If the list contains fewer entries than the indicated sample size, the entire list will be processed. The user is then asked to rate each n-gram in the sample for whether it is a true positive (i.e. correctly extracted n-gram) or a false positive (an incorrectly extracted sequence). At the end of the rating the percentages for true positives (TPs) vs. false positives (FPs) will be displayed. The rated sample, together with the ratings is saved in the same directory as the consolidated list with `tpfltd` added into its name. This list can be provided again as file to be assessed if one wants to check the accuracy figures again or has not finished assessing all cases in the sample.

C. Auxiliary Scripts

Users can interact directly (via command line) with auxiliary scripts that are included in the SubString package. Each of the scripts and their function is illustrated in the following. Commands shown need to be typed into a Terminal window.

CUTOFF.SH

cutoff.sh can be used to enforce frequency-cut-offs on n-gram lists. This is how cutoff.sh is used:

```
cutoff.sh -f N FILE+
```

Where 'N' stands for a number representing the minimum frequency to be applied. This value should be between 3 and 101. FILE+ again stands for one or more input n-gram lists to be frequency-filtered. cutoff.sh provides an option `-i` which will attach the cutoff frequency (minimum frequency minus 1). Consequently, the following command would result in an output file named `4-gram.cut.14.txt`:

```
cutoff.sh -if 15 test_data-B/example1/4-gram.txt
```

Whereas the following command would result in an identical output file but with the regular expression representing the cutoff frequency attached to its name, resulting in the file name `4-gram.cut.([0-9]|[0-0][0-9]|1[0-4]|14).txt`:

```
cutoff.sh -f 15 test_data-B/example1/4-gram.txt
```

LENGTH-ADJUST.SH

This programme is run as part of the nenu-based interface if the lengh-adjustment step is selected. When run as an auxiliary script, it takes as input fully consolidated lists (i.e. the output of substring-processor.sh) and does two things:

1. It looks at all the n-grams in the list that feature a frequency below a given cutoff (sensibly this would be the same cutoff as used when producing the input lists to substring-processor.sh) and rather than just getting rid of them (as cutoff.sh would do), it checks if there are substrings in the list with which the below-cutoff n-gram could be combined to reach above-cutoff frequency. For example, consider the 3-gram "the·battle·of" with a frequency of 4. At a chosen minimum frequency of 7, this sequence would be eliminated. However, let us say there is a substring, the 2-gram "the·battle" in the list with a frequency of 13. Since we are looking at consolidated n-grams, if we were to just eliminate "the·battle·of", we'd lose those 4 instances of "the·battle" that occur as part of "the·battle·of" which would result in a less than accurate frequency for "the·battle". length-adjust.sh therefore gets rid of "the·battle·of", and adds its frequency to the substring "the·battle" which then displays a new frequency of 17 (13 + 4).
2. There are also cases where we would want such a combination of superstring and substring to occur even if the superstring is not below the cutoff frequency. For example, the sequence "for example" is a very frequent one in English. Typically, therefore, an n-gram extraction would yield a number of n-grams containing "for example" in a consolidated list such as the one in (2) below. This is fine for some applications, but in others we would prefer the various extensions of the 2-gram "for·example·" NOT to appear and their frequencies instead be listed under "for·example·" itself. length-adjust.sh therefore reduces extensions off a frequent base if certain conditions are met.[1] These conditions are the following:

    1. number of extensions: adjustment can only take place if there are at least 6 extensions off a base. In (2), there are 13 extensions off the base.
    2. frequency ratio base - extension: the frequency of the base must be at least a third of the frequency of base and all extensions together. In (2), the base with a frequency of 1745 accounts for 53% of the total frequency of base and extensions ((1745 + 1547) / 1745), so this hurdle is passed.
    3. base frequency: the base frequency must not be over 1,000. In (2), it is 1745 so with the conservative default settings, (2) would not actually be adjusted (but see below for how to change this).

.

```
(2)
for·example·1745
for·example·the·    605
for·example·a·  220
for·example·in· 169
for·example·in·the· 104
for·example·if· 100
for·example·NE· 73
for·example·an· 47
for·example·when·   46
for·example·it· 45
for·example·NUM·    35
for·example·some·   35
for·example·by· 34
for·example·to· 34
```

length-adjust.sh is called like this:

```
length-adjust.sh -c N [-OPTIONS] FILE+
```

Where FILE+ is one or more frequency-consolidated n-gram list and 'N' is the minimum frequency (see discussion of cutoff.sh, above). There are various options available, the most important of which are discussed in the following.

The -o option only adjusts n-grams that are below the frequency cutoff provided.

Using options -[2-9], -b and -s, we can vary the conditions applied to any length adjustment of n-grams that are above cutoff. In directory 'example3', there is a file named `example.lst` which contains the list given in (2) above. As we saw above, the default settings specify that a base must not have a frequency over 1,000 to be corrected. This is a conservative value set to avoid too sweeping an approach to length adjustment. In the case of our data in (2), we want to override these settings by specifying a new value of 2,000 using option -b as follows:

```
length-adjust.sh -b 2000 -c 9 test_data-B/example3/example.txt
```

The adjustments made are logged in a file named `correction-log.txt`. The output file `example.adju.txt` will now contain the adjusted n-grams in (3). Note that 'for·example·in·the·' was NOT adjusted because it is an extension of more than one element off the base 'for·example·'. This is again to prevent adjustments that are too sweeping.

```
(3)
for·example·    3188
for·example·in·the·     104
```

We can adjust the other parameters as well. -[2-9] allows adjusting the threshold of the number of extensions necessary to trigger adjustment. If we wanted to correct n-grams that had at least 9 extensions, we would pass the option -9. -s allows the adjustment of the ratio of base frequency vs. total frequency. If we wanted to correct n-grams that had a ratio of base frequency vs. base + extension frequency of 70% we would pass the option -s 0.7. If the following command is run, no adjustments will be made because the base frequency vs. base + extensions frequency ratio is below 0.7:

```
length-adjust.sh -b 2000 -9 -s 0.7 -c 9 test_data-B/example3/example.txt
```

EN-FILTER.SH

This programme takes as input n-gram lists that need to be filtered in order to eliminate n-grams that are unlikely to form a unit of meaning. It is intended for English language data only and employs a lexico-structural filter (a filter which uses specific words and aspects of grammatical structure) with around 50 entries to delete certain n-grams. en-filter.sh is called like this:

```
en-filter.sh [-OPTIONS] FILE+
```

where the most frequent option is -d to preserve document frequencies in the input file. The output file will have `1.5.3.enfltd` added to its name.

RANDOM_LINES.sh

This script takes a sample of random lines from input files and puts them into a document in the present working directory with the extension .random0000, where 0000 represents the sample size. The script is called as follows:

```
random_lines.sh [-OPTIONS] FILE
```

The sample size can be specified using the -n option, so for a sample of 100 lines, `-n 100` would be specified.

TP-FILTER.sh

This programme is used to assess the accuracy of a (consolidated) list of n-grams. Typically a random sample produced with random_lines.sh is used as an input file. TP-filter.sh is called like this:

```
TP-filter.sh [-OPTIONS] FILE
```

The programme then presents a screen where assessments can be entered. Once all the n-grams in the supplied file have been assessed, the programme shows the percentages of true positives vs. false positives. The following options are available: -t (to display the percentages based on tokens rather than types), -o (use a rating scheme other than true positives/false positives). Output files have the extention .tpfltd added to their name.

---

1. The specifics of these conditions are customisable using options -[2-9] -b and -s, see `length-adjust.sh -h` for details. ↩