SubString v0.9.4

The SubString package is a set of Unix Shell scripts used to consolidate frequencies of word n-grams of various different n (i.e. word n-grams of different lengths). In the process, the frequencies of substrings are reduced by the frequencies of their superstrings and a consolidated list with n-grams of different length is produced without an inflation of the overall count. The functions performed by this package will primarily be of interest to linguists and computational linguists working on formulaic language, multi-word sequences and other phraseological phenomena.

A. Frequency Consolidation

To illustrate how frequency consolidation among n-grams of various lengths proceeds, let us assume we have as input the n-grams in (1)a. These will have been extracted from a corpus and their frequency of occurrence in the corpus is indicated by the number to the right of each n-gram.

The 4-gram 'have a lovely time' occurs with a frequency of 15. The 3-grams 'have a lovely' and 'a lovely time' occur 58 and 44 times respectively. 15 of those occurrences are, however, occurrences as part of the superstring 'have a lovely time' (since they are substrings of 'have a lovely time'). To get the consolidated frequency of occurrence for 'have a lovely' and 'a lovely time' (i.e. the occurrences of these 3-grams on their own, NOT counting when they occur in a longer string), we therefore deduct the frequency of their superstring (15) from their own frequency. This results a consolidated frequency of 43 for 'have a lovely' (i.e. 58 minus 15) and 29 for 'a lovely time' (i.e. 44 minus 15), as shown in (1)b.

The remaining 2-grams ('have a', 'a lovely' and 'lovely time') are also substrings of 'have a lovely time' and therefore also need to have their frequency reduced by 15 (resulting in a frequency of 34692 for 'have a', 86 for 'a lovely' and 30 for 'lovely time'. In addition, 'have a' and 'a lovely' are substrings of 'have a lovely' and therefore the frequency of 'have a lovely' which is now 43, needs to be deducted from their frequencies. This results in a new

frequency of 34649 for 'have a' and 43 for 'a lovely'. 'a lovely' and 'lovely time' are furthermore substrings of 'a lovely time' and consequently need to have their frequencies reduced by that of 'a lovely time' (i.e. by 29): the consolidated frequency of 'a lovely' is now 14, that of 'lovely time' is 1. The output of the frequency consolidation is shown in (1)b.

```
(1)a                          (1)b

have a lovely time  15        have a lovely time   15
have a lovely       58        have a lovely        43
a lovely time       44        a lovely time        29
have a           34707        have a            34649
a lovely           101        a lovely             14
lovely time         45        lovely time           1
```

A more in-depth theoretical description and justification of the algorithm is currently in preparation. See also O'Donnell (2011) for a discussion of issues involved and alternative approaches.

B. Components

The current release of the SubString package contains the following components:

- `substring.sh` is the main script for user interaction
- `cutoff.sh` performs frequency cut-offs
- `length-adjust.sh` can be used to adjust the n of n-grams in a fully consolidated list
- `listconv.sh` can be used to convert input lists to required format
- `README.md / README.pdf` this document
- `test_data` a directory containing test data
- `EUPL.pdf` a copy of the European Union Public License under which SubString is licensed.

C. Installation

SubString was tested on MacOS X (v. 10.6 to 10.9) and Ubuntu Linux (versions Xubuntu 9.04 and 10.04), but should run on all platforms on which the bash shell (version 3.2 or higher) is installed. This includes Windows with the Cygwin package installed.

Generally, all scripts (i.e. the files ending in .sh) should be placed in a location that is in the user's $PATH variable (or the location should be added to the $PATH variable) so they can be called from the command line. A good place to put the scripts might be /usr/local/bin or $HOME/bin.

Detailed instructions of how to do this are given here for MacOS and Ubuntu:

1. open the Terminal application MacOS X: in Applications/Utilities Ubuntu Linux: via menu Applications>Accessories>Terminal
2. type: `mkdir /usr/local/bin` (it may say 'File exists', that's fine)
3. type: `echo $PATH` (if you can see /usr/local/bin somewhere in the output, move to step 8, if not carry on with the next step)
4. type: `cd $HOME` type: `cp .profile .profile.bkup` (if it says there no such file, that's fine)
5. type: `vi .profile`
6. move to an empty line and press the `i` key, then enter the following: `PATH=/usr/local/bin:$PATH`
7. press ESC, then type `:wq!`
8. move into the SubString directory. This can be done by typing `cd` (make sure there is a space after `cd` but don't press return yet) and then dragging the SubString folder onto the Terminal window and pressing return.
9. type: `sudo cp *.sh /usr/local/bin` (you will need to enter an admin password)

   Done!

The installation can be verified by calling each script's help function for the command line of a Terminal window:

1. open a new terminal window
2. Type `substring.sh -h` and hit enter. Try the same with `cutoff.sh -h`, `listconv.sh -h`, etc.
3. If the help texts appear, all is in order.

For further tests, you may wish to run SubString on the test data (see next section)

D. Operation

LISTCONV.SH

substring.sh (s. below) requires n-gram lists to be formatted in either the following ways: (the parts in square brackets are optional)

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. The listconv.sh script can be used to convert n-gram lists into this format. listconv.sh is able to convert output lists created with the N-Gram Processor, the Ngram Statistics Package, the NGramTools or n-grams lists provided by the Google Books corpus, although the latter will need previous selection of the relevant data as the same n-grams are listed for many different years.

To convert an n-gram list, simply supply the names of the files to be converted as arguments:

```
listconv.sh FILE+
```

Where FILE+ is the name of one or more input lists to be converted (if the list is not in the current working directory, the path needs to be indicated as well, i.e. /path/to/directory/list.lst). listconv.sh will convert the lists and create a backup of the original unconverted lists with the suffix .old in the same directory. More information on listconv.sh is available from the help function of listconv.sh which is called by typing the following:

```
listconv.sh -h
```

CUTOFF.SH

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

```
cutoff.sh -f '(CUTOFF REGEX)' FILE+
```

Where '(CUTOFF REGEX)' stands for a regular expression detailing the frequency cut-off to be applied. This needs to be quoted (i.e. have '( )' around it). Cutoff frequencies need to cover the frequencies that one wishes to filter out. For example, if all n-grams with a frequency of less then 10 should be filtered out, the appropriate regular expression would be '([0-9])'. Further examples are:

```
        <11     '([0-9]|[1][0])'
        <12     '([0-9]|[1][0-1])'
        <13     '([0-9]|[1][0-2])'
        <14     '([0-9]|[1][0-3])'
        <15     '([0-9]|[1][0-4])'
        <16     '([0-9]|[1][0-5])'
        <26     '([1-9]|1[0-9]|2[012345])'
```

FILE+ again stands for one or more input n-gram lists to be frequency-filtered. In normal operation the output lists will have the relevant regular expression attached to its name. This can sometimes be cumbersome and for this reason cutoff.sh provides an option `-i` which will merely attach any final plain number to the output lists, rather than the full expression. Consequently, the following command would result in an output file named `list.txt.cut.14`:

```
cutoff.sh -if '([0-9]|[1][0-4]|14)' list.txt
```

Whereas the following command would result in an otherwise identical output file named `list.txt.cut.([0-9]|[1][0-4])`:

```
cutoff.sh -f '([0-9]|[1][0-4])' list.txt
```

SUBSTRING.SH

substring.sh is the script that manages the frequency consolidation. Before detailing its operation, it is useful to highlight three areas of detail on how substring.sh handles frequency consolidation.

Firstly, n-gram lists extracted from a source document usually need to be filtered to be useful. Various filters are employed for this purpose including the use of minimal frequencies of occurrence or threshold values of statistical association measures. substring.sh accepts lists that have already been filtered in one way or another. For frequency filtering, the script cutoff.sh, which is included in the SubString package, can be used as explained above.

Secondly, regardless of the type of filter applied, frequency consolidation with substrd.sh can be performed more accurately if the script has access to the unfiltered (or less severely filtered) n-gram lists as well as the filtered lists given as input. This applies to cases where n-gram lists of length n=5 and above are involved. substring.sh therefore includes an optional preparatory stage which looks up certain n-grams in unfiltered lists prior to frequency consolidation in order to resolve overlaps between n-grams. The script can also be run without this preparatory stage.

Thirdly, n-gram lists to be consolidated by substring.sh 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.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.
6. input lists must be of the format n<>gram<> 1[1] or n·gram· 1[1], that is, the words of the n-grams are separated by '<>' or by '·', then a tab follows, then the FREQUENCY COUNT. Optionally, a tab and possibly some more information may follow. This additional information could be document counts (as provided by the N-Gram Processor software package) or measures of association strength. It is important to note that only the first number after the n-gram (which is assumed to be

the frequency) is consolidated and will appear in the consolidated output list. The only exception to this is if the -d option is active, in which case it is assumed that the number following the frequency is a document count (i.e. a count of the number of documents in which the n-gram appears) and this document count will appear in the output list. To convert lists into the required input format, use listconv.sh as described above.

To consolidate n-gram lists, substring.sh is called like this:

```
substring.sh [OPTIONS] FILE+
```

FILE+ again stands for the input lists, i.e. the lists that should be consolidated. The options will be discussed as we go along. A simple example, involving the data in example (1)a above, can be run as follows: (the necessary input lists are supplied in the directory SubString/test_data/example1)

1) move to the directory example1 in the test_data directory

2) type the following and press enter:

```
substring.sh 2-gram.lst 3-gram.lst 4-gram.lst
```

substring.sh now takes the input lists, consolidates them and displays the consolidated output which should correspond to the result in example (1)b (although listed in a different order). The output is sorted according to the n-grams themselves, rather than their frequency. To have the output sorted in decending order of frequency (n-gram with highest frequency first), use the -f option:

```
substring.sh -f 2-gram.lst 3-gram.lst 4-gram.lst
```

Now the n-grams appear in the order of frequency. To see processing information, the -v option can be invoked. While the -v option is active, the result will not be displayed on screen. Instead, the result is put in an output file in the current working directory: 2.lst-4.lst.substrd. The default name for the consolidated list ends in .substrd, but depending on the input lists, the preceding part of the name will be different. It is possible to specify where the output list should go and how it should be named by invoking the -o option followed by output file name (and a path if desired). This holds regardless of whether the -v option is active. Here's an example:

```
substring.sh -vo $HOME/Desktop/OUT.lst 2-gram.lst 3-gram.lst 4-gram.lst
```

This will name the consolidated list `OUT.lst` and place it in the user's desktop directory. The default .substrd - suffixed file in the working directory is not created.

A more complex example is the following. Input lists for this example are provided in SubString/test_data/example2. The lists with the extension .cut.9 have been frequency-filtered with cutoff.sh and contain only n-grams with a minimum frequency of 10. The lists without extension contain unfiltered lists of n-grams.

1. move to the directory example2 in the test_data directory
2. type the following (all on one line), then press enter:

```
 substring.sh -vf 2-grams.cut.9 3-grams.cut.9 4-grams.cut.9 5-
grams.cut.9 6-grams.cut.9 7-grams.cut.9
```

substring.sh takes longer to process the lists this time because the amount of data is larger (lists are based on about 120,000 words of text). The output (the .substrd file) is again in the current directory. The output list will not contain any 7-grams because there are none above the chosen cutoff of 9 (the list 7-grams.cut.9 is in fact empty) and there is only one 6-gram above the cutoff. The list produced should be identical to `2.lst-6.lst.substrd-GOLD1` found in the directory example2.[1] As mentioned earlier, frequency consolidation can be performed more accurately if the script has access to the unfiltered (or less severely filtered) n-gram lists as well as the filtered lists given as input. Unfiltered lists (or less severely filtered lists) are supplied using the -u option (u stands for 'unfiltered'). Each unfiltered list to be considered needs to be passed separately and the -u option will only accept lists of 4-grams and longer n-grams. Using the example data in example2, the unfiltered lists are passed like this:

```
substring.sh -vf -u 4-grams -u 5-grams -u 6-grams -u 7-grams 2-grams.cut.9
3-grams.cut.9 4-grams.cut.9 5-grams.cut.9 6-grams.cut.9 7-grams.cut.9
```

The list produced should be identical to `2.lst-7.lst.substrd-GOLD2` found in the directory 'example2'. This list will be slightly different from the previous one. It now contains a few 7-grams (as shown in the file name which is now 2.lst-7.lst.substrd) because some 7-grams were taken from the unfiltered lists and integrated into the results. Again, to make sure we only have n-grams with a frequency of at least 10, cutoff.sh can be used to remove n-grams. When unfiltered (or less severely filtered) lists are supplied, they should be passed so that there is an unfiltered list for each of the filtered input lists from 4-gram lists up: if we supply an unfiltered 4-gram list, we should also supply an unfiltered 5-gram list, 6-gram list, etc. up to the maximum length of filtered n-gram lists supplied to the script. It is recommended that the unfiltered lists supplied are minimally filtered to exclude n-grams with frequency 1. When processing very large amounts of data, it will be more convenient to limit unfiltered lists to n-grams of a minimum frequency of 3 or even more. This can be achieved using the cutoff.sh script to filter unfiltered lists prior to consolidation.

In the data of example2, the inclusion of unfiltered lists did not alter the result by much. In some cases, especially when processing large amounts of data that were heavily filtered, the provision of additional unfiltered lists can improve the accuracy of the consolidation more

notably. However, even with the use of additional unfiltered lists, it is sometimes unavoidable that the frequency consolidation of some n-grams is inaccurate, resulting in them receiving negative frequencies. This is caused by an insufficient resolution of overlapping n-grams and cannot be automatically resolved. In such cases the n-grams concerned are not listed in the output file but instead are written to a special output file named `neg_freq.lst` which is also placed in the current working directory.

So far, we have used 2-grams as the shortest n-grams. One may also wish to consolidate right down to 1-grams (i.e. lists of single words). This can be achieved by supplying a suitable 1-gram list as is included in the 'example3' directory. Running substring.sh on the data in the 'example3' directory of the test_data, as shown below, will consolidate down to wordlist level. The process also results in two words receiving a negative frequency. Those words are listed in the file named `neg_freq.lst` which will appear in the directory whenever negative frequencies are encountered. Make sure you are in the 'example3' directory before running the following command:

```
substring.sh -vf -u 4-grams -u 5-grams -u 6-grams -u 7-grams 1-grams.cut.7
2-grams.cut.7 3-grams.cut.7 4-grams.cut.7  5-grams.cut.7 6-grams.cut.7 7-
grams.cut.7
```

The output file created should be idential to `1.lst-5.lst.substrd-GOLD`. For further processing options, type in the following command:

```
substring.sh -h
```

LENGTH-ADJUST.SH

This programme takes as input fully consolidated lists (i.e. the output of substring.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.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, `1.lst-5.lst.substrd-GOLD` in example3 contains the 3-gram "der·Schlacht·bei·" (the battle of) with a frequency of 4. This is below the cutoff of 7 and would therefore be eliminated by cutoff.sh. However, there is a substring, the 2-gram "der·Schlacht·" ([of] the battle) in the list with a frequency of 13. Since we are now looking at consolidated n-grams, if we were to just eliminate "der·Schlacht·bei·", we'd lose those 4 instances of "der·Schlacht·" that occur as part of "der·Schlacht·bei·" which would result in a less than accurate frequency for "der·Schlacht·". length-adjust.sh therefore gets rid of "der·Schlacht·bei·", but also adds its frequency to the substring "der·Schlacht·" 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.[2] 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 '([REGEX])' [-OPTIONS] FILE+
```

Where FILE+ is one or more frequency-consolidated n-gram list and '(REGEX)' is the regular expression detailing the frequency cutoff (for details on this type of regular expression, 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. In the

'example3' directory, we can run the following command to length-adjust sequences below a frequency of 7:

```
length-adjust.sh -oc '([0-7])' 1.lst-5.lst.substrd-GOLD
```

You should now find a new file named `1.lst-5.lst.substrd-GOLD.adju` in the directory. It contains the length-adjusted list.[3] You will see another new file named `correction-log.txt` which lists all the length-adjustments made.

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 'example4', 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: (change to the example4 directory before typing this)

```
length-adjust.sh -b 2000 -c '([0-7])' example.lst
```

Again, the adjustments made are logged in a file named `correction-log.txt`. The output file 'example.txt.adju' 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 only lengths of 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 only lengths of 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 '([0-7])' example.lst
```

E. Known Issues

None reported at this time. Issues can be raised at http://github.com/buerki/SubString/issues. This is also the place to make feature requests.

F. Warning

SubString is still at beta stage. As article 7 of the EUPL states, the software is a work in progress, which is continuously improved. It is not a finished work and may therefore contain defects or “bugs” inherent to this type of software development. For the above reason, the software is provided under the Licence on an “as is” basis and without warranties of any kind concerning it, including without limitation merchantability, fitness for a particular purpose, absence of defects or errors, accuracy, non-infringement of intellectual property rights other than copyright as stated in Article 6 of this Licence. This disclaimer of warranty is an essential part of the Licence and a condition for the grant of any rights to the SubString.

G. Copyright, licensing, download

The project resides at http://buerki.github.com/SubString/ and new versions will be posted there. A mirror is kept at https://developer.berlios.de/projects/substrd. Suggestions and feedback are welcome. To be notified of new releases, go to https://github.com/buerki/SubString, click on the 'Watch' button and sign in.

---

1. Depending on the settings of locale on each machine, the sort order of n-grams might differ, but the same n-grams will be listed with the same frequencies. The consolidation will have produced some n-grams with frequencies below 10. To make sure the list only contains n-grams with a frequency of at least 10, cutoff.sh can be used to remove any n-grams below this frequency. An even better way to deal with n-grams below cutoff is to use the programme length-adjust.sh (see below). ↩
2. The specifics of these conditions are customisable using options -[2-9] -b and -s, see `length-adjust.sh -h` for details. ↩
3. Since even after length-adjustment there may be some n-grams below cutoff frequency (such as when a combination of superstring and substring occurred but their combined frequency is not above cutoff), it will be necessary to run cutoff.sh as well if all n-grams below cutoff are to be eliminated.↩