SubString v0.9.9

The SubString package is a set of Unix shell scripts used to consolidate frequencies of word n-grams of various different n (i.e. 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'. 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 found in Buerki (2017). See also O'Donnell (2011) for a discussion of issues involved and alternative approaches. The algorithm implemented in SubString works from n-gram lists (rather than source texts) and some assumptions are therefore made regarding what sequences are substrings of what other sequences without the possibility to check this on the basis of the source text. While consolidations are therefore coherent and accurate with regard to input lists, results will inherently differ from what a consolidation on the basis of full text might yield.

B. Components

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

- `substring.sh` is the main script for user interaction
- `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.

A number of further auxiliary scripts are also installed (see section E, below).

C. Compatible Systems

SubString was tested on OS X (v. 10.11 & 10.12), Ubuntu Linux (version Xubuntu 14.04) and on Windows 7 and 8 using Cygwin (v. 1.8). SubString should run on all platforms on which a bash shell is installed. For efficient processing of larger amounts of data, bash v. 4 is necessary (although the software will automatically substitute a slower algorithm if only bash v. 3 is available).[1] Especially under Windows systems, there can be problems if file and/or folder names contain spaces, so it is best to avoid locations on the system that have folder name with a space.

D. Installation

Using the supplied installers (recommended):

Double-clickable installers are provided for OS X, Xubuntu-Linux. On Windows, the Cygwin envirnonment has to be installed first. These installers replace previous versions of the installed files. It may be necessary to log out and log in again before the installation takes full effect. If an icon was generated on the desktop (or in the applications folder) during installation, the installation can be verified by double-clicking on it and SubString's main menu should appear.

OS X / Xubuntu

Inside the SubString directory, double-click on `Xubuntu_installer` (for Xubuntu), `OSX_installer` (OS X). The installer will ask the user to agree with the licence and whether an icon should be placed on the desktop.

Windows

The Cygwin environment needs to be installed first. During the installation procedure for Cygwin, the optional 'bc' package from the 'maths' category will need to be installed. A guide on how to install Cygwin is found here.

After Cygwin has been installed, double click on the `Cygwin_installer` or `Cygwin64_installer` (try both if one does not work) to start the installation process.

Manual installation / other flavours of Linux

1. open a Terminal window

   OS X: in Applications/Utilities

   Xbuntu Linux: via menu Applications>Accessories>Terminal

   Cygwin: via the link on the Windows desktop to Cygwin Terminal

2. drop the `install.sh` script (located inside the SubString-X.X.X directory) onto the terminal window and press ENTER.

This should start the installation process.

If an entirely manual installation is necessary, generally, all scripts (i.e. the files ending in .sh inside the folder named `src`) 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 `~/bin`. The installer places the files in `~/bin` (where `~` is a shortcut to `/Users/YOURNAME/bin` where YOURNAME is the user name.)

To uninstall, open a terminal window then drop the `install.sh` script (located inside the SubString-X.X.X directory) on to the terminal window, then type `-u` so that the line ends in `-u` and press ENTER. Alternatively, manually delete the relevant files from the directory `~/bin`.

D. Basic Operation

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

If you chose not to create a launch icon at installation, SubString can be started from a terminal by typing `substring.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 0.9.9



       Please choose a function

       (S) substring reduce and consoliate n-gram lists
       (A) assess accuracy of a consolidated list
       (X) exit

      >
```

N-gram lists processed by SubString 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 SubString 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 substring reduction and consolidation on test data, follow the steps below:

1. Choose `s` 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-X.X.X directory, you will find a folder called `test_data`. 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(s). The list with '1.5.1.enfltd' in its name is the filtered list (if the filter was chosen during processing). Those lists 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 choos `s` 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 list and a 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 `A` from the main menu. The user is prompted to drop a consolidated (and, if desired, filtered) 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 the extention .tpfltd added to 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.

E. Advanced Operation

If more flexibility is needed than the menu-based interface provides, users can interact directly (via command line) with auxiliary scripts that are included in the SubString package. Each of the scripts and its 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 `list.txt.cut.14`:

```
cutoff.sh -if 15 list.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 `list.txt.cut.([0-9]|[1][0-4])`:

```
cutoff.sh -f 15 list.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.[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 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: (change to the example3 directory before typing this)

```
length-adjust.sh -b 2000 -c 9 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 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 the extention .fltd attached 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.

F. Known Issues

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

G. Copyright, licensing, download

The project resides at http://buerki.github.com/SubString/ and new versions will be posted there. 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.

H. Warning

As article 7 of the EUPL states, the SubString 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 SubString.

---

1. Most recent operating system versions have bash v. 4 installed as standard, but MacOS X has bash v. 3.2 installed as standard. Bash v. 4 can be installed using MacPorts, Homebrew or similar and then the new version would either need to be put in the directory `/bin` (replacing the old version) or the first line of the `substring.sh` script would need adjusting to point to the new version of bash (if installed via MacPorts, the new line would read `#!/opt/local/bin/bash` instead of `#!/usr/bin/env bash`). ↩

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