**Max Awk Seminar**

**Tutorial online: https://www-users.york.ac.uk/~mijp1/teaching/2nd\_year\_Comp\_Lab/guides/grep\_awk\_sed.pdf**

**Wiki about awk, tutorials: https://en.wikibooks.org/wiki/Category:Book:An\_Awk\_Primer**

Using and awk, jq and GNU to process text. Quick, easy to debug, free.

In bash Terminal

AWK:

line-delimited files only (new line for each new entry, e.g. tweet, comment), each new line indicates a new field

what is the new line character? \n on Linux, on windows probably

preprocessing in another tool: remove new lines or escape them in another tool

field separator: awk -F “ ” for the delimiter “\t”, then the command {}, then the text

to show something: {print $1}

index fields : $1, $2, put $0 for all fields, $NF last field, $NF-1 before last

awk has an implicit loop over all lines running

show first and second field of each entry: print $1, $2

pype into awk: (pype = straight line)

cat tokenized\_text.tmp I “awk command”

Get to delimited file:

cat text.tmp I awk -f

-vOFS” “ output file separator

split function: Split string up at this character, split into array, split($0,m), use to tokenize

fields are tokens

gsub to replace characters with other stuff

cat text.tmp I wc - l - see number of lines

array is like a python dictionary

add zero to strings to make sure they are numeric

array: “I” “live” “in” “Vienna” - index is array 1 = “I”

If you populate array yourself you can’t be sure it’s ordered.

row subsets: conditionals

print row if column 1 i>0

awj -F “ “ ’$1>1{print $1}’ anger\_wordfreq implicit if

`print length($0)}’

regular expressions:

$2~/zer regular expression “zer”

$2~/^a

$==“n” every line that is exactly n, exact matching

regular matching: ~ word stem plus everything that comes after

in tutorial, look at:

inbuilt variables (like NF for last field)

Awk scripts:

call them with -f for file

awk - f filename.awk

Dictionary matching:

Kinds of dictionaries:

1. list of words

Prepare text, clean, put into awk, tokenize, and put it out again

implicit loop over each line, explicit loop over fields, and look if terms are in each field

dictanalysis.v1.5\_easier.awk:

BEGIN {} before the first line. Print out head, set variables here.

END clause after the last line. if you want to print only in the end.

Start program with {

construct table with words that should be matched

if NR==FNR, field number of records, n in the file. to make sure you stay in the first file. awk counts lines of all processed files (also the word list), therefore set this so it always starts at 1 again. (in the command in the terminal: awk - F wordlist textfile)

could be: NR>1 to skip the first line

vocabulary: wildcard

if wildcard loop:

substr( to remove last letter with -1)

everything is a string for awk, otherwise: “220”+0 to indicate it is a number

control statements: next refers to next record, continue to the next iteration in the loop, break breaks the loop, exit exits the program

if not wildcard, simply take the first field

if (FNR==1){ … the header in the next file (next file is our tokenized text)

print “name”, “number\_tokens”

if(NF==0){ …to check for empty lines, put NAs

do the matching:

1. set the hit count to 0: sum\_hits=0
2. loop over all fields, c-style loop: for (i=1; i<=NF; i++) this means: start condition; end condition; increase i by 1, i - - would be decrease by one, by 2: i+=2
3. for non exact words (wildcarwords), using regular expressions: if ($i~wildcard… is a bit slow. Quicker: shorten word by one character at a time until you get a match. Decreasing loop from 23 (longest word in dictionary):
   1. for(s=23; s<01; s- -){
   2. if(substr($i, 1,s) in prosocial\_words\_wildcard){
   3. sum\_hits++
   4. break because not more words }

substr above takes a string, $i, takes it from position 1 to position s, subpart of a string

gsub - replaces every match (or regular expression)

sub replaces the first match

Second type of dictionary: words that can be in more than one category

much quicker than running the script for each category, because you only have to write to the disk once. words can be in multiple categories now: unfreundlich 1 5 means once in anger, 5 times in sad for example.

Max script: liwc\_multi.vq.q.awk script

prepare dictionaries in wide format to read them into R, in R data table fread

use -vORS vector separator to add OR between all words

to delete the last OR/AND combine it with the function sed. pype sed AND /$/$ take the AND and replace it with the end

awk ’NR<{print ?} file 1 file2 file3 > combined

write them all into one file

Use Gnu parallel to run multiple awk tasks in parallel on different cores

function: xargs - j8 number of cores -I{} awk command

JQ: processing .json files

use it to get out the text into a csv, then load in into awk, in between maybe parallelise with GNU or xargs, then load into R

jq ‘.cts’ file.json I head - 10

jq -r raw text

‘.p[].tx’ @csv