Text Processing in Linux
A Tutorial for NLP (CSE 562/662)

Kristy Hollingshead
Winter 2006

www.cslu.ogi.edu/~hollingk/NLP_tutorial.html

Overview

- The goal here is to make your lives easier!
- NLP is very text-intensive
- Simple tools for text-manipulation
  - `sed`
  - `awk`
  - `bash/tcsh`
  - `grep`
  - `head, tail`
  - `wc`
- When & how to use each of these tools

Regular expressions crash course

- [a-z] exactly one lowercase letter
- [a-z]* zero or more lowercase letters
- [a-z]+ one or more lowercase letters
- [a-zA-Z0-9] one lowercase or uppercase letter, or a digit
- [^(] match anything that is *not* '('

sed: overview

- a stream editor
- WHEN
  - "search-and-replace"
  - great for using regular expressions to change something in the text
- HOW
  - `sed 's/regexp/replacement/g'`
    - 's/... = substitute
    - .../g' = global replace
      (otherwise will only replace first occurrence on a line!)

sed: special characters

- ^ the start of a line...
  except at the beginning of a character set (e.g., [^a-z]), where it complements the set
- $ the end of a line
- & the text that matched the regexp
- We'll see all of these in examples...

sed: (simple) examples

- eg.txt =
  The cops saw the robber with the binoculars
- `sed 's/robber/thief/g' eg.txt`
  The cops saw the thief with the binoculars
- `sed 's/^/She said, "/g' eg.txt`
  She said, "The cops saw the robber with the binoculars
- `sed 's/^/She said, "/g' eg.txt | sed 's/$/"/g'`
  She said, "The cops saw the robber with the binoculars"

sed: examples from the homework!

- eg2.txt =
  (TOP (NP (DT The) (NNS cops)) (VP (VBD saw) (NP (DT the) (NN robber)) (PP (IN with) (NP (DT the) (NNS binoculars)))))
- "remove the syntactic labels"
  hint!: all of (and only) the syntactic labels start with '('

```
sed 's/([^ ]* //g' eg2.txt | sed 's/)//g'
```

  The cops saw the robber with the binoculars
- "now add explicit start & stop sentence symbols (<s> and </s>, respectively)"

```
sed 's/([^ ]* //g' eg2.txt  | sed 's/)//g' |
sed 's/^/<s> /g' | sed 's/$/ <\/s>/g'
```

  <s> The cops saw the robber with the binoculars </s>

Unix pipes & redirection crash course

- `>` write out to file
  (creates file if doesn't exist, re-writes file if already exists)
- `>>` append to file
- `<` read in from file
- `|` "pipe through"

- `sed 's/AAA/BBB/g' file.txt`
  is equivalent to
- `cat file.txt | sed 's/AAA/BBB/g'`
  is equivalent to
- `sed 's/AAA/BBB/g' < file.txt`

awk: overview

- a simple programming language specifically designed for text processing
  - somewhat similar in nature to Tcl
- WHEN
  - using simple variables (counters, arrays, etc.)
  - treating each word in a line individually
- HOW
  -
```
awk 'BEGIN      {initializations}
     /regexp1/  {actions1}
     /regexp2/  {actions2}
     END        {final actions}' file.txt
```
    (blue text indicates optional components)

awk: useful constructions & examples

- `$1, $2, …, $NF`
  imagine every line of text as a row in a table; one word per column. `$1` will be the word in the first column, `$2` the next column, and so on up through `$NF` (the last word on the line)
- `$0` – the entire row
- eg3.txt =
  The cow jumped over the moon
- `awk '{print $2}' eg3.txt`
  cow
- `cat eg3.txt | awk '{$NF=42; print $0; \ $1="cool NLP!"; print $0;}' -`

  The cow jumped over the 42
  cool NLP! cow jumped over the 42

awk: useful constructions & examples

- eg3.txt =
  The cow jumped over the moon
- if statements
  - `awk '{if ($1 == "he") { print $0; }}' eg3.txt`
    (empty)
  - `awk '{if ($1 ~ "he") { print $0; } else { … }}' eg3.txt`
    The cow jumped over the moon
- for loops
  - `awk '{for (j=1; j <= NF; j++) { print $j }}' eg3.txt`
  - what if I only wanted to print every other word (each on a new line), in reverse order?

```
awk '{for (j=NF; j > 0; j--=2) { print $j }}' eg3.txt
```

The cow jumped over the moon

awk: useful constructions & examples

- eg4.txt =
  The cow jumped over the moon
  And the dish ran away with the spoon
- printf statements
  - `awk '{for (j=1; j <= NF; j++) { \ printf("%d\t%s\n", j, $j);}}' eg4.txt`
  - what if I want continuous numbering?
  - `awk 'BEGIN {idx=0;} {for (j=1; j <= NF; j++) { \ printf("%d\t%s\n", idx, $j); idx++;}}' eg4.txt`
- substrings
  - substr(<string>, <start>, <end>)
  - `awk '{for (j=1; j <= NF; j++) { \ for (k=1; k < length($j), k++) { \ printf("%s ",substr($j,k,1));} print "";}}' eg4.txt`

    T h e c o w j u m p e d o v e r t h e m o o n
    A n d t h e d i s h r a n a w a y w i t h t h e s p o o n

1 The
2 cow
3 jumped
4 over
5 the
6 moon
1 And
2 the

bash: overview

- shell script
- WHEN
  - repetitively applying the same commands to many different files
  - automate common tasks
- HOW
  - on the command line
  - in a file (type `` `which bash` '' to find your location):

```
#!/usr/bin/bash
<commands…>
```

13

bash: examples

- `for f in *.txt; do echo $f;`
  `tail -1 $f >> txt.tails; done`
- `for (( j=0; j < 4; j++ )); do`
  `cat part$j.txt >> parts0-3.txt; done`
- `for f in hw1.*; do mv $f ${f//hw1/hw2};`
  `done`

14

head & tail

- WHEN
  - dividing files into sub-parts by line number
  - viewing the *i*-th line in a file
- HOW
  - `head -42 file.txt`
    for the first 42 lines of file.txt
  - `tail -42 file.txt`
    for the last 42 lines of file.txt
  - `tail +42 file.txt`
    for everything *except the first 42* lines of file.txt
  - `head -42 file.txt | tail -1`
    to see the 42nd line of file.txt

15

miscellaneous

- wc
  - a counting utility
  - `wc -l file.txt`
    counts number of lines in a file
  - `wc -c` counts number of characters,
    `wc -w` counts number of words,
    `wc -L` counts max number of characters on a line
- sort
  - `sort -u file.txt`
    for a uniquely-sorted list of each line in the file
- tr
  - "translation" utility
  - `cat mixed.txt | tr [a-z] [A-Z] > upper.txt`

16

Putting it all together!

- Let's say I have a text file, and I'd like to break it up into 4 equally-sized (by number of lines) files.
- `wc -l orig.txt`
  8000
- `head -2000 orig.txt > part1.txt`
- `tail +2000 orig.txt | head -2000 > part2.txt`
- `tail +2000 orig.txt | tail +2000 | head -2000 > part3.txt`
- `tail -2000 orig.txt > part4.txt`

17

Putting it all together!

- Now for each of those files, I'd like to see a numbered list of all the capitalized words that occurred in each file… but I want the words all in lowercase.

```
for f in part*;
 do echo $f;
 cat $f | awk 'BEGIN {idx=0} {
  for (j=1; j <= NF; j++)
   if (substr($j,1,1) ~ "[A-Z]") {
    printf("%d\t%s\n", idx, $j);
    idx++;
   }
 }' - | tr [A-Z] [a-z] >
 ${f//part/out};
 echo ${f//part/out};
done
```

18

Putting it all together!

- Now I'd like to see that same list, but only see each word once (unique).
- hint: you can tell 'sort' which fields to sort on
- e.g., `sort +3 -4` will skip the first 3 fields and stop the sort at the end of field 4; this will then sort on the 4th field.
- ```
  for f in out*; do cat $f |
  sort +1 -2 -u > ${f//out/unique};
  done
  ```
- and if I wanted to re-number the unique lists:
- ```
  for f in out*; do cat $f |
  sort +1 -2 -u | awk 'BEGIN {idx=0} {$1=idx;
  print $0; idx++}' - > ${f//out/unique};
  done
  ```

19

Putting it all together!

- And finally, I'd like to see the first 5 & the last 5 words in each list, but I already have a list of these first-and-lasts started, so I just want to add onto it instead of creating a new one.
- ```
  for f in unique*; do
  head -5 $f >> top-and-bottom-5;
  tail -5 $f >> top-and-bottom-5; done
  ```
- (and of course, I could then re-number top-and-bottom-5 if I were so inclined)

20

sed: (more complicated) example

- eg2.txt =
  (TOP (NP (DT The) (NNS cops)) (VP (VBD saw) (NP (DT the) (NN robber)) (PP (IN with) (NP (DT the) (NNS binoculars)))))
- "show just the POS-and-word pairs: e.g., (POS word)"

```
cat eg2.txt | sed 's/([^ ]* [^(]/~&/g' |
sed 's/[^)~]*~/ /g' |
sed 's/^ *//g' |
sed 's/))*/)/g'
```

(DT The) (NNS cops) (VBD saw) (DT the) (NN robber) (IN with) (DT the) (NNS binoculars)

21

Resources

- You can always look at the man page for help on any of these tools!
  - i.e.: `` `man sed' `` or `` `man tail' ``
- My favorite online resources:
  - sed: www.grymoire.com/Unix/Sed.html
  - awk: www.vectorsite.net/tsawk.html
  - bash: www.tldp.org/LDP/abs/html/
    (particularly section 9.2 on string manipulation)
- Google it. ☺

22

Warning!

- These tools are meant for very simple text-processing applications!
- Don't abuse them by trying to implement computationally-intensive programs with them
  - like Viterbi search and chart parsing
- Use a more suitable language like C, C++, or Java.

23
```