CS 124/LINGUIST 180 From Languages to Information

Unix for Poets

Dan Jurafsky

(original by Ken Church, modifications by me and Chris Manning)

Stanford University

Unix for Poets

- Text is everywhere
  - The Web
  - Dictionaries, corpora, email, etc.
  - Billions and billions of words
- What can we do with it all?
- It is better to do something simple, than nothing at all.
- You can do simple things from a Unix command-line
- Sometimes it's much faster even than writing a quick python tool
- DIY is very satisfying

Exercises we'll be doing today

1. Count words in a text
2. Sort a list of words in various ways
   - ascii order
   - ''rhyming'' order
3. Extract useful info from a dictionary
4. Compute ngram statistics
5. Work with parts of speech in tagged text

Tools

- **grep**: search for a pattern (regular expression)
- **sort**
- **uniq –c** (count duplicates)
- **tr** (translate characters)
- **wc** (word – or line – count)
- **sed** (edit string -- replacement)
- **cat** (send file(s) in stream)
- **echo** (send text in stream)
- **cut** (columns in tab-separated files)
- **paste** (paste columns)
- **head**
- **tail**
- **rev** (reverse lines)
- **comm**
- **join**
- **shuf** (shuffle lines of text)

Prerequisites: get the text file we are using

- rice: ssh into a rice and then do (don't forget the final ".")

```
cp afs/ir/class/cs124/nyt_200811.txt.gz .
```

- Or download to your own Mac or Unix laptop this file:

http://cs124.stanford.edu/nyt_200811.txt.gz

```
or scp cardinal:/afs/ir/class/cs124/nyt_200811.txt.gz  .
```

- Then:

```
gunzip nyt_200811.txt.gz
```

If you have a Windows machine

- You'll need to get UNIX tools running so you can get to rice or even run things locally
- Follow the instructions on this page: http://web.stanford.edu/class/cs124/unixfromWindows.html
- Tips:
  - If you've downloaded a file (for example from a browser) in Windows, you can find it through your WSL (Ubuntu) shell at /mnt/c/Users/[your username]/Downloads. You can then work with it there or make a copy to a location of your choice (i.e. your Ubuntu home directory) using cp.
  - "gzip -d filename" in terminal works to unzip if you don't have a

Prerequisites

- `The unix "man" command`
  - e.g., `man tr` (shows command options; not friendly)
- Input/output redirection:
  - `>` `"output to a file"`
  - `<` `"input from a file"`
  - `|` `"pipe"`
- `CTRL-C`

Exercise 1: Count words in a text

- Input: text file (nyt_200811.txt) (after it's gunzipped)
- Output: list of words in the file with freq counts
- Algorithm
  1. Tokenize (`tr`)
  2. Sort (`sort`)
  3. Count duplicates (`uniq –c`)
- Go read the man pages and figure out how to pipe these together

Solution to Exercise 1

- `tr -sc 'A-Za-z' '\n' < nyt_200811.txt | sort | uniq -c`

```
 1271 A
    3 AA
    3 AAA
    1 AARP
    1 ABANDONED
    2 ABBA
    1 ABBY
```

(Do you get a different sort order?
In some versions of UNIX, sort doesn't use ASCII order (uppercase before lowercase).)

Some of the output

- ```
  tr -sc 'A-Za-z' '\n'
  < nyt_200811.txt |
  sort | uniq -c |
  head –n 5
  ```

```
  1271 A
     3 AA
     3 AAA
     1 AARP
     1 ABANDONED
```

- ```
  tr -sc 'A-Za-z' '\n'
  < nyt_200811.txt |
  sort | uniq -c |
  head
  ```
- Gives you the first 10 lines
- tail does the same with the end of the input
- (You can omit the “-n” but it’s discouraged.)

Extended Counting Exercises

1. Merge upper and lower case by downcasing everything
   - Hint: Put in a second tr command

2. How common are different sequences of vowels (e.g., the sequences "ieu" or just "e" in "lieutenant")?
   - Hint: Put in a second tr command

Solutions

Merge upper and lower case by downcasing everything

```
tr -sc 'A-Za-z' '\n' < nyt_200811.txt | tr 'A-Z' 'a-z'
| sort | uniq -c
```

or

```
tr -sc 'A-Za-z' '\n' < nyt_200811.txt | tr '[:upper:]'
'[:lower:]' | sort | uniq -c
```

1. tokenize by replacing the complement of letters with newlines
2. replace all uppercase with lowercase
3. sort alphabetically
4. merge duplicates and show counts

Solutions

- How common are different sequences of vowels (e.g., ieu)

```
tr -sc 'A-Za-z' '\n' < nyt_200811.txt | tr
'A-Z' 'a-z' | tr -sc 'aeiou' '\n' | sort |
uniq -c
```

Sorting and reversing lines of text

- `sort`
- `sort –f` Ignore case
- `sort –n` Numeric order
- `sort –r` Reverse sort
- `sort –nr` Reverse numeric sort

- `echo "Hello" | rev`

Counting and sorting exercises

- Find the 50 most common words in the NYT
  - Hint: Use sort a second time, then head

- Find the words in the NYT that end in "zz"
  - Hint: Look at the end of a list of reversed words
  - tr 'A-Z' 'a-z' < filename | tr –sc 'a-z' '\n' | rev | sort | rev | uniq -c

Counting and sorting exercises

- Find the 50 most common words in the NYT

```
tr -sc 'A-Za-z' '\n' < nyt_200811.txt |
sort | uniq -c | sort -nr | head -n 50
```

- Find the words in the NYT that end in "zz"

```
tr -sc 'A-Za-z' '\n' < nyt_200811.txt | tr
'A-Z' 'a-z' | rev | sort | uniq -c | rev |
tail -n 10
```

Lesson

- Piping commands together can be simple yet powerful in Unix
- It gives flexibility.

- Traditional Unix philosophy: small tools that can be composed

Bigrams = word pairs and their counts

Algorithm:

1. Tokenize by word
2. Create two almost-duplicate files of words, off by one line, using **tail**
3. **paste** them together so as to get $word_i$ and $word_{i+1}$ on the same line
4. Count

Bigrams

- `tr -sc 'A-Za-z' '\n' < nyt_200811.txt > nyt.words`
- `tail -n +2 nyt.words > nyt.nextwords`
- `paste nyt.words nyt.nextwords > nyt.bigrams`
- `head –n 5 nyt.bigrams`

```
KBR       said
said      Friday
Friday    the
the       global
global    economic
```

Exercises

- Find the 10 most common bigrams
  - (For you to look at:) What part-of-speech pattern are most of them?

- Find the 10 most common trigrams

Solutions

- Find the 10 most common bigrams

```
tr 'A-Z' 'a-z' < nyt.bigrams | sort | uniq
-c | sort -nr | head -n 10
```

- Find the 10 most common trigrams

```
tail -n +3 nyt.words > nyt.thirdwords

paste nyt.words nyt.nextwords nyt.thirdwords >
nyt.trigrams

cat nyt.trigrams | tr "[:upper:]" "[:lower:]" | sort |
uniq -c | sort -rn | head -n 10
```

grep

- Grep finds patterns specified as regular expressions
- `grep rebuilt nyt_200811.txt`

Conn and Johnson, has been rebuilt, among the first of the 222 move into their rebuilt home, sleeping under the same roof for the the part of town that was wiped away and is being rebuilt. That is to laser trace what was there and rebuilt it with accuracy," she home - is expected to be rebuilt by spring. Braasch promises that a the anonymous places where the country will have to be rebuilt, "The party will not be rebuilt without moderates being a part of

grep

- Grep finds patterns specified as regular expressions
  - **g**lobally search for **r**egular **e**xpression and **p**rint

- Finding words ending in –ing:
- `grep 'ing$' nyt.words |sort | uniq –c`

grep

- grep is a filter – you keep only some lines of the input
- `grep gh` keep lines containing “gh”
- `grep '^con'` keep lines beginning with “con”
- `grep 'ing$'` keep lines ending with “ing”
- `grep –v gh` keep lines NOT containing “gh”

grep versus egrep (grep –E)

- `egrep or grep -E` [extended syntax]
- In egrep, +, ?, |, (, and) are automatically metacharacters
- In grep, you have to backslash them
- To find words ALL IN UPPERCASE:
- `egrep '^[A-Z]+$' nyt.words |sort|uniq -c`
- == `grep '^[A-Z]\+$' nyt.words |sort|uniq -c`

(confusingly on some systems grep acts like egrep)

Counting lines, words, characters

- `wc nyt_200811.txt`

```
140000 1007597 6070784 nyt_200811.txt
```

- `wc -l nyt.words`

```
1017618 nyt.words
```

Exercise: Why is the number of words different?

Exercises on grep & wc

- How many all uppercase words are there in this NYT file?
- How many 4-letter words?
- How many different words are there with no vowels
  - What subtypes do they belong to?
- How many “1 syllable” words are there
  - That is, ones with exactly one sequence of vowels

Type/token distinction: different words (types) vs. instances (tokens)

Solutions on grep & wc

- How many all uppercase words are there in this NYT file?

```
grep -E '^[A-Z]+$' nyt.words | wc
```

- How many 4-letter words?

```
grep -E '^[a-zA-Z]{4}$' nyt.words | wc
```

- How many different words are there with no vowels

```
grep -v '[AEIOUaeiou]' nyt.words | sort | uniq | wc
```

- How many “1 syllable” words are there
-
```
tr 'A-Z' 'a-z' < nyt.words | grep -P
'^[^aeiou]*[aeiou]+[^aeiou]*$' | uniq | wc
```

Type/token distinction: different words (types) vs. instances (tokens)

sed

- sed is used when you need to make systematic changes to strings in a file (larger changes than ‘tr’)
- It’s line based: you optionally specify a line (by regex or line numbers) and specific a regex substitution to make
- For example to change all cases of “George” to “Jane”:

- `sed 's/George/Jane/' nyt_200811.txt | less`

sed exercises

- Count frequency of word initial consonant sequences
  - Take tokenized words
  - Delete the first vowel through the end of the word
  - Sort and count

- Count word final consonant sequences

sed exercises

- Count frequency of word initial consonant sequences

```
tr "[:upper:]" "[:lower:]" < nyt.words | sed
's/[aeiou].*$//' | sort | uniq -c
```

- Count word final consonant sequences

```
tr "[:upper:]" "[:lower:]" < nyt.words | sed
's/^.*[aeiou]//g' | sort | uniq -c | sort -rn
| less
```

cut – tab separated files

```
scp <sunet>@myth.stanford.edu:/afs/ir/class/cs124/parses.conll.gz .
gunzip parses.conll.gz
head –n 5 parses.conll
```

```
1   Influential   _   JJ    JJ    _   2    amod   _   _
2   members       _   NNS   NNS   _   10   nsubj  _   _
3   of            _   IN    IN    _   2    prep   _   _
4   the           _   DT    DT    _   6    det    _   _
5   House         _   NNP   NNP   _   6    nn     _   _
```

The Penn TreeBank Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

cut – tab separated files

- Frequency of different parts of speech:

```
cut -f 4 parses.conll | sort | uniq -c | sort -nr
```

- Get just words and their parts of speech:

```
cut -f 2,4 parses.conll
```

- You can deal with comma separated files with: cut –d,

cut exercises

- How often is ‘that’ used as a determiner (DT) “that rabbit” versus a complementizer (IN) “I know that they are plastic” versus a relative (WDT) “The class that I love”
  - Hint: With grep , you can use '\t' for a tab character

- What determiners occur in the data? What are the 5 most common?

cut exercise solutions

- How often is ‘that’ used as a determiner (DT) “that rabbit” versus a complementizer (IN) “I know that they are plastic” versus a relative (WDT) “The class that I love”

```
cat parses.conll | grep -P
'(that\t_\tDT)|(that\t_\tIN)|(that\t_\tWDT)' | cut -f
2,4 | sort | uniq -c
```

- What determiners are in the data? What are the 5 most common?

```
cat parses.conll | tr 'A-Z' 'a-z'| grep -P
'\tdt\t' | cut -f 2,4 | sort | uniq -c | sort -
rn |head -n 5
```