Tools for Text

Unix Pipe Fitting for Data Analysis

David A. Smith
Tools
Data
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="Fiction">
    <title lang="en">Harry Potter and the Goblet of Fire</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="Science Fiction">
    <title lang="en">The Three-Body Problem</title>
    <author>Erik Tuguo</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
Data Structures

Sequence of lines:
nobody:*:-2:
nogroup:*:-1:
wheel:*:0:root
daemon:*:1:root
kmem:*:2:root
sys:*:3:root
tty:*:4:root
operator:*:5:root
mail:*:6:teamsserver

Delimiters separate fields.
Data Structures

Sequence of lines

Whitespace separates fields

1330 was
1177 had
1159 her
949 be
926 not
855 it
853 that
819 she
787 as
Sir Walter Elliot, of Kellynch Hall, in Somersetshire, was a man who, for his own amusement, never took up any book but the Baronetage; there he found occupation for an idle hour, and consolation in a distressed one; there his faculties were roused into admiration and respect, by contemplating the limited remnant of the earliest patents; there any unwelcome sensations, arising from domestic affairs.
Data Structures

- “Plain” text
- “Flat file” databases
- Linear not random access
- Scales to arbitrary sizes
Tools for Research
Tools for Research

• If we knew what we wanted to do...
Tools for Research

• If we knew what we wanted to do...
• ...and what our data should look like, ...
Tools for Research

• If we knew what we wanted to do...
• ... and what our data should look like, ...
• ... we could build special-purpose programs and data structures.
Tools for Research

• If we knew what we wanted to do...
• ... and what our data should look like, ...
• ... we could build special-purpose programs and data structures.
• But we don’t know what we want!
Tools for Research

• If we knew what we wanted to do...
• ... and what our data should look like, ...
• ... we could build special-purpose programs and data structures.
• But we don’t know what we want!
  ✤ (It’s research)
Tools for Research

• If we knew what we wanted to do...
• ... and what our data should look like, ...
• ... we could build special-purpose programs and data structures.
• But we don’t know what we want!
  ✤ (It’s research)
• Some tools already exist; some don’t.
Tools for Research

• If we knew what we wanted to do...
• ... and what our data should look like, ...
• ... we could build special-purpose programs and data structures.
• But we don’t know what we want!
  ✤ (It’s research)
• Some tools already exist; some don’t.
• Tools communicate with simple text data.
Big Tools

Composability

Discoverability
Today’s Tasks

- Sorting and counting words
- Regular expressions
  - Matching and substitution
- Scripting and automation
Getting Some Data

- Unzip (e.g., `unzip -a gutenberg.zip`)
- Open the Terminal app, or similar
- Go to the new directory
  - `cd ~/Downloads/gutenberg`
- `ls`
Looking at Files

• **head**
  ✶ −100 : first 100 lines

• **tail**
  ✶ −20 : last 20 lines
  ✶ −n  +20 : tail starting at 20th line

• **less**

• Your favorite text editor
Counting Words

• `wc`
  ✤ Discoverability: `man wc`

• But I really meant, counting *different* words
  ✤ Really simple tokenization: `tr`
  ✤ Sorting: `sort`
  ✤ Aggregation (counting): `uniq`
Directing Data

• < Input from a file
• > Output to a file
• | Pipe
Counting Words

• `tr -sc 'A-Za-z' '\n'`
  • Turns one set of characters into another
  • What about: `tr 'A-Z' 'a-z'`

• `sort`
  • `sort -n` numerical order
  • `sort -r` reverse order
  • `sort -k 2` sort by 2nd (whitespace) field

• `uniq -c` count contiguous repeated lines
Counting Words

$ tr -sc 'A-Za-z' '
' < austen-emma.txt
| sort | uniq -c
  1
125  A
  31 Abbey
  1 Abbots
  1 Abdy
...

Try piping to less, head, or out to a file.
More Counting

- Sort by word frequency
- Merge counts for upper and lower case
- Sort by word endings (rev)
Sorting by Frequency

$ tr -sc 'A-Za-z' '\n' < austen-emma.txt | sort | uniq -c | sort -rn
5186 to
4846 the
4673 and
4281 of
3192 I
...

Merging Case

$ tr -sc 'A-Za-z' '"n" < austen-emma.txt
  | tr 'A-Z' 'a-z' | sort | uniq -c
  1
1595 a
  1 abbreviation
  1 abdication
  1 abide
  3 abilities
30 able
...

...
Parallel Streams

- `paste file1 file2`
  - `paste <(pipe1) <(pipe2)`
- `cut -f 1`
- **Counting bigrams**
- `join -1 2 -2 2 : join on second field`
  - Inputs must be sorted on the joined field!
Parallel Streams

$ tr -sc 'A-Za-z' '\n' < austen-emma.txt | tr 'A-Z' 'a-z' > austen-emma.words
$ paste austen-emma.words <(tail -n +2 austen-emma.words)

emma   by
by     jane
jane   austen
austen volume
volume i
i      chapter
chapter i
i      emma
emma   woodhouse
woodhouse    handsome
handsome    clever
clever    and
and      rich
rich      with
...

Counting Bigrams

$ paste austen-emma.words <(tail -n +2 austen-emma.words) | sort | uniq -c | sort
  608 to be
  566 of the
  449 it was
  446 in the
  395 i am
  334 she had
  331 she was
  308 had been
  301 it is
  299 mr knightley
  283 i have
  278 could not
  265 of her
  256 mrs weston
...
Comparing Counts

$ tr -sc 'A-Za-z' '\n' < austen-sense.txt | grep -v '^$' | tr 'A-Z' 'a-z' | sort | uniq -c > austen-sense.counts
$ tr -sc 'A-Za-z' '\n' < austen-persuasion.txt | grep -v '^$' | tr 'A-Z' 'a-z' | sort | uniq -c > austen-persuasion.counts
$ join -1 2 -2 2 austen-sense.counts austen-persuasion.counts
a 2092 1595
abilities 9 3
able 46 30
abode 5 1
about 144 97
above 20 6
abroad 9 5
absence 11 9
...
Comparing Counts

$ tr -sc 'A-Za-z' '
' < austen-sense.txt | grep -v '^$' | wc -l
120734
$ tr -sc 'A-Za-z' '
' < austen-persuasion.txt | grep -v '^$' | wc -l
84123
$ join -l 2 -2 2 austen-sense.counts austen-persuasion.counts | awk
'{ print $1, $2 * 10000 / 120734, $3 * 10000 / 84123 }' > sense-persuasion.counts
$ less sense-persuasion.counts
a 173.273 189.603
abilities 0.74544 0.356621
able 3.81003 3.56621
abode 0.414134 0.118874
about 11.927 11.5307
above 1.65653 0.713241
abroad 0.74544 0.594368
absence 0.911094 1.06986
absent 0.911094 1.06986
absent 0.24848 0.475494
...
Comparing Counts

Filtered out words w/similar frequencies
Digression: R graphs

```r
> counts <- read.table("sense-persuasion.counts")
> names(counts) <- alist(word, sense, persuasion)
> pcount <- subset(counts, (sense > 2 | persuasion > 2) &
abs(log(sense/persuasion)) > 1.1)
> plot(pcounts$sense, pcounts$persuasion, type="n", log="xy")
> text(pcount$sense, pcount$persuasion, label=pcount$word)
```
Filters

• `grep '^\[A-Z\]'`
  * “generalized regular expression parser” ???!
  * Matches words beginning with a capital

• Count results?

• What are regular expressions?
Regular Expressions
Regular Expressions

• Two types of characters
  ✤ Literal
    • Every “normal” alphanumeric character is an RE and matches itself
  ✤ Meta-characters
    • Special characters that allow you to combine REs in various ways

• Example
  ✤ \texttt{a} matches \texttt{a}
  ✤ \texttt{a*} matches \( \varepsilon \) or \texttt{a} or \texttt{aa} or \texttt{aaa} or \ldots
<table>
<thead>
<tr>
<th>Regex Basics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Character Concat</strong></td>
</tr>
<tr>
<td>went</td>
</tr>
<tr>
<td>Alternatives</td>
</tr>
<tr>
<td>[aeiou]</td>
</tr>
<tr>
<td>[^aeiou]</td>
</tr>
<tr>
<td>.</td>
</tr>
<tr>
<td><strong>Loops &amp; skips</strong></td>
</tr>
<tr>
<td>one or more</td>
</tr>
<tr>
<td>zero or one</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
More Regexes

• Special characters
  – \t  tab          \v  vertical tab
  – \n  newline        \r  carriage return

• Aliases (shorthand)
  – \d  digits       [0-9]
  – \D  non-digits   [^0-9]
  – \w  alphabetic   [a-zA-Z]
  – \W  non-alphabetic [^a-zA-Z]
  – \s  whitespace   [\t\n\r\f\v]
  – \W  alphabetic   [a-zA-Z]

• Examples
  – \d+ dollars  3 dollars, 50 dollars, 982 dollars
  – \w*oo\w*   food, boo, oodles

• Escape character
  – \  is the general escape character; e.g. \ is not a wildcard, but matches a period .
  – if you want to use \ in a string it has to be escaped \\
More Regexes

• **Anchors.** AKA, “zero width characters”.
• They match positions in the text.
  – ^ beginning of line
  – $ end of line
  – \b word boundary, i.e. location with \w on one side but not on the other.
  – \B negated word boundary, i.e. any location that would not match \b

• Examples:
  – \bthe\b the together

• **Counters** \{1\}, \{1,2\}, {3,}
More Regexes

• Grouping
  – a (good|bad) movie
  – He said it (again and and )*again.

• Parens also indicate Registers (saved contents)
  – b(\w+) h\1
    matches boohoo and baha, but not boohaa
    The digit after the \ indicates which of multiple paren
    groups, as ordered by when then were opened.

• Grouping without the cost of register saving
  – He went (?:this|that) way.
Exercising grep

• How many upper case words?
  ✤ How many distinct upper case words?

• Are there any words with no vowels?

• Any one-syllable (i.e. one vowel) words?
  ✤ Omit words with silent e

• Gotchas: grep doesn’t support all modern “perl-style” regex extensions, try egrep
Regex Substitutions

- Lots of languages support them: perl, python, ruby, Java (and the JVM), ...

- perl -pe 's/regex/subst/g'
  - easiest on the command line

- perl -000 -pe 's/regex/sub/g'
  - Match paragraphs, not lines

- perl -ne 'print $1 while /foo (bar)/g'
  - Print only the “captured” part
Munging with Regexes

• Remove common morphological suffixes from word list and get counts
• Patterns that indicate proper names
• Extract direct speech from novels
• Put each paragraph on a single line
  ✤ For, e.g., Mallet input
Replicable Research
Shell Scripts

• If you’ve figured out a good pipeline, write it down!
  ✤ Better yet, make it executable.
  ✤ Make a file histogram containing

```
#!/bin/sh
tr -sc 'A-Za-z' '
' | \ 
  sort | uniq -c | sort -rn
```

• Then run chmod +x histogram
Makefiles

- Output from one process is input for another
  - Capture dependencies in a Makefile
  - Run `make target` to generate output

```
comparison:  emma.txt  sense.txt
  join -1 2 -2 2 $^  |  \
  hist-stats > $@

%.hist: %.txt histogram
  ./histogram < $< > $@
```
Where Next?

• Transforming data for more specialized analysis tools
  ✤ E.g., Gephi, R, Voyant

• Programming to supply missing tools
  ✤ E.g., R, python (“Programming Historian”)

• Tools for more structured (XML) text
  ✤ Though regexes can work pretty well!
Thanks

• Inspired by Ken Church’s “Unix for Poets”

• My colleagues at the NULab for Texts, Maps & Networks