CS 124 / LINGUIST 180

Stanford University

Text Processing with UNIX

Adapted from
Ken Church's "Unix for Poets"
by Dan Jurafsky and Chris Manning

Text processing with UNIX

Billions of words and text are everywhere

- The web, email, social media

What can we do with it all?

The UNIX intuition:

- Modularity and simplicity! Simple command-line tools combined together
- Often faster even than writing a quick python tool
- DIY is very satisfying

7 exercises we'll be doing today

1. Count words in a text with "tr", "sort", and "uniq"
2. Counting more complex text patterns with "tr"
3. Using "sort" in more powerful ways for counting
4. Using "grep" to find patterns
5. Data ethics: Where did this data come from?
6. Simple tricks to compute n-gram statistics
7. Unigram Language Modeling

Tools

sort
uniq –c (count duplicates)
tr (translate characters)
grep: search for a pattern (regular expression)
cat (send file(s) in stream)
head
tail
rev (reverse lines)
sed (edit string -- replacement)

Prereqs:

Mac:
Open the Terminal app

Windows:

Make sure you have Ubuntu (PA0)

Open PowerShell and type in the `wsl` command

Prerequisites: get the text file we are using

- Myth/Rice: ssh into a farmshare machine and then run the following command (don't forget the final "."):

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

- Local: Download this file on your own laptop using right click + “save as”:

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

Prerequisites

The UNIX "`man`" command

- e.g., `man tr`
- `man` shows you the command options
  - it's not particularly friendly

Prerequisites

How to chain shell commands and deal with input/output

Input/output redirection:

- `>` "output to a file"
- `<` "input from a file"
- `|` "pipe"

`CTRL-C`

The **`less`** command (quit by typing "`q`")

Exercise 1: Count words in a text

Input: text file (nyt_200811.txt)

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`

```
  633 A
    1 AA
    1 AARP
    1 ABBY
   41 ABC
    1 ABCNews
```

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

Note that you may get a slightly different sort because sort doesn't use ASCII order (uppercase before lowercase) in some versions of Unix.

Some of the output

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

```
633 A
   1 AA
   1 AARP
   1 ABBY
  41 ABC
```

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

Ex 2: Extended Counting Exercises

1. Merge upper and lower case by downcasing everything
   - 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

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`

Ex 3: Counting and sorting exercises

Find the 50 most common words in the NYT

- Hint: Use sort a second time, then head

Ex 3 Counting and sorting solutions

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
```

1. tokenize by replacing the complement of letters with newlines
2. sort alphabetically
3. merge duplicates and show counts
4. sort numerically (i.e. by the counts) and in descending/reverse order
5. show the top 50

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
```

First, let’s save our previous work (where each word in the article takes a new line) into a file called `nyt.words`:

- `tr -sc 'A-Za-z' '\n' < nyt_200811.txt > nyt.words`

grep

Grep finds patterns specified as regular expressions

- **g**lobally search for **r**egular **e**xpression and **p**rint

(A much easier way to find words ending in a pattern than the rev method we used before)

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)

Ex 4: Exercises on grep & wc

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

How many 4-letter words?

Ex 4 Solutions on grep & wc

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

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

Answer: 8604

1. look for words that start capitalized, are 1 capitalized letter or more, and end capitalized.
2. word count

How many 4-letter words?

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

Answer: 86943

Lesson

Piping commands together can be simple yet powerful in Unix

It gives flexibility.

Traditional Unix philosophy: small tools that can be composed

Ex 5: Data Ethics

The text for today's lab comes from the New York Times Annotated Corpus, released by the Linguistic Data Consortium.

Understanding a dataset's origins gives us insight into its scope and limitations, which can affect the conclusions we draw from it.

Take a few minutes with your group to explore the origins of this text.

What types of texts, from what time period, genre, and language, are included in this dataset?

What types are not? Then, answer questions on the following slide

Data Ethics Questions

1. Imagine you are a researcher creating an educational platform as part of a high-school history course, and you trained your model solely on text from this corpus. Though the NYT is a respected publication, its content reflects its very particular context. **What biases could arise from relying solely on this corpus for training your model?** (For example, whose perspectives are predominantly included, and whose might be underrepresented or excluded?)
2. With your group, brainstorm **ways to reduce the biases** identified in part (1) when building this tool. How would you evaluate the effectiveness of these methods in reducing biases?

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

- `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
```

Ex 6: Bigrams

Find the 10 most common bigrams

- (Just for you to notice:) What part-of-speech pattern are most of them?

Ex 6 Solutions

Find the 10 most common bigrams

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

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

1. lowercase bigrams
2. sort alphabetically
3. count
4. sort in descending order by number
5. look at top 10

Computing Perplexity

The details of computing the perplexity of a test set W depends on which language model we use. Here's the perplexity of W with a unigram language model (just the geometric mean of the inverse of the unigram probabilities):

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i)}} \tag{3.16}$$

Ex 7: Unigram Language Modeling

Training corpus: “I scream you scream we all scream for ice cream”

Test corpus 1: “I scream we scream”

Test corpus 2: “you scream ice cream”

1. Build unigram LM from training corpus (compute by hand)
2. Compute perplexity for each test corpus (ok to use a spreadsheet)
3. Determine which test corpus has lower perplexity

Ex 7 Solutions

Training corpus: “I scream you scream we all scream for ice cream”

Test corpus 1: “I scream we scream”

Test corpus 2: “you scream ice cream”

Word	Probability from training corpus
scream	0.3
I	0.1
you	0.1
we	0.1
all	0.1
for	0.1
ice	0.1
cream	0.1

Perplexities:

Test corpus 1: $\sqrt[4]{\frac{10}{1} \cdot \frac{10}{3} \cdot \frac{10}{1} \cdot \frac{10}{3}} \approx 5.77.$

Test corpus 2: $\sqrt[4]{\frac{10}{1} \cdot \frac{10}{3} \cdot \frac{10}{1} \cdot \frac{10}{1}} \approx 7.60.$

Ex 7: Unigram Language Modeling

Training corpus: “I scream you scream we all scream for ice cream”

Test corpus 1: “I scream we scream”

Test corpus 2: “you scream ice cream”

Reflection:

Why does test corpus 1 have a **lower** perplexity than test corpus 2?

- ➔ Hint: Does lower perplexity mean lower probability or higher probability?

Extra Credit – Secret Message

- Now, let's get some extra credit practice with Unix!
- The answers to the extra credit exercises will reveal a secret message.
- You can do the extra credit at home! Just submit before midnight!
- We will be working with the following text file for these exercises: https://web.stanford.edu/class/cs124/lec/secret_ec.txt
- To receive credit, enter the secret message here:
- https://forms.gle/fGNKgdhbfskXjchT6

Extra Credit Exercise 1

Find the 2 most common words in secret_ec.txt containing the letter e.

Your answer will correspond to the first two words of the secret message.

Extra Credit Exercise 2

Find the 2 most common bigrams in secret_ec.txt where the second word in the bigram ends with a consonant.

Your answer will correspond to the next four words of the secret message.

Extra Credit Exercise 3

Find all 5-letter-long words that only appear once in secret_ec.txt.

Concatenate (by hand) your result. This will be the final word of the secret message.