

## CS124(2021)·课程资料包 @ShowMeAI



视频

中英双语字幕



课件

一键打包下载



笔记

官方笔记翻译



代码

作业项目解析



视频·B站 [ 扫码或点击链接 ]

<https://www.bilibili.com/video/BV1YA411w7ym>



课件 & 代码·博客 [ 扫码或点击链接 ]

<http://blog.showmeai.tech/cs124/>

文本处理

信息检索

编辑距离

推荐系统

神经嵌入

序列标注

社交网络分析

PageRank

倒排索引

对话系统

协同过滤

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP20+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets·持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2: 扫码点击**底部菜单栏**

称为 **AI 内容创作者**? 回复 [ 添砖加瓦 ]



# 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

# 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**

# Prereq: If you are on a Mac:

- Open the Terminal app

# Prereq: If you are on a Windows 10 machine and don't have Ubuntu on your machine:

- For today's class, it's easiest to work with someone who has a Mac or Linux machine, or has Ubuntu already.
- Or you can do the following so that you will have this ability:
  - Watch the first 9 minutes of Bryan's lovely pa0 video about how to download and install Ubuntu:  
<https://canvas.stanford.edu/courses/136258/modules/items/873662>
  - Watch Chris Gregg's excellent UNIX videos here: Logging in, the first 7 "File System" videos, and the first 8 "useful commands" videos.  
<https://web.stanford.edu/class/archive/cs/cs107/cs107.1186/unixref/>
  - From there you can use the ssh command to connect to the myth machines. Just be sure to keep track in your own mind of whether you're on myth or your own laptop at any given moment! The ssh command you want to type is:  
ssh [sunset]@rice.stanford.edu where [sunset] is your SUNet ID. It will ask for your password, which is your usual Stanford password, and you will have to do two-step authentication.

# Prerequisites: get the text file we are using

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

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

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

[http://cs124.stanford.edu/nyt\\_200811.txt](http://cs124.stanford.edu/nyt_200811.txt)

Or:

```
scp cardinal:/afs/ir/class/cs124/WWW/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

**OK, now go to your group work rooms!**

# 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

(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
```

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.)

# 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

# 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`  
`70334 509851 3052306 nyt_200811.txt`
- `wc -l nyt.words`  
`70334 nyt_200811.txt`

**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 specify 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]*[aeiou]//' | sort | uniq -c | sort -rn  
| less
```

## CS124(2021)·课程资料包 @ShowMeAI



视频

中英双语字幕



课件

一键打包下载



笔记

官方笔记翻译



代码

作业项目解析



视频·B站 [ 扫码或点击链接 ]

<https://www.bilibili.com/video/BV1YA411w7ym>



课件 & 代码·博客 [ 扫码或点击链接 ]

<http://blog.showmeai.tech/cs124/>

文本处理

信息检索

编辑距离

推荐系统

神经嵌入

序列标注

社交网络分析

PageRank

倒排索引

对话系统

协同过滤

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP20+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

**点击** 课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2: 扫码点击**底部菜单栏**

称为 **AI 内容创作者**? 回复 [ 添砖加瓦 ]