In [1]:
```
%%javascript
/*******************************************************************************
*****************
Known Mathjax Issue with Chrome - a rounding issue adds a border to the right
 of mathjax markup
https://github.com/mathjax/MathJax/issues/1300
A quick hack to fix this based on stackoverflow discussions:
http://stackoverflow.com/questions/34277967/chrome-rendering-mathjax-equations
-with-a-trailing-vertical-line
********************************************************************************
****************/

$('.math>span').css("border-left-color","transparent")
```

In [2]:
```
%reload_ext autoreload
%autoreload 2
```

MIDS - w261 Machine Learning At Scale

Course Lead: Dr James G. Shanahan (**email** Jimi via James.Shanahan *AT* gmail.com)

Assignment - HW1

Name: *Razib Shishir*
Class: MIDS w261 (Section *Fall 2016 Group 3*)
Email: *shishir@ischool.berkeley.edu*
Week: 1

Due Time: HW is due the Tuesday of the following week by 8AM (West coast time). I.e., Tuesday, Sept 6, 2016 in the case of this homework.

Table of Contents

1 Instructions

Back to Table of Contents

MIDS UC Berkeley, Machine Learning at Scale DATSCIW261 ASSIGNMENT #1

Version 2016-09-2

=== INSTRUCTIONS for SUBMISSIONS === Follow the instructions for submissions carefully.

https://docs.google.com/forms/d/1ZOr9RnIe_A06AcZDB6K1mJN4vrLeSmS2PD6Xm3eOiis/viewform?usp=send_form
(https://docs.google.com/forms/d/1ZOr9RnIe_A06AcZDB6K1mJN4vrLeSmS2PD6Xm3eOiis/viewform?usp=send_form)

IMPORTANT

HW1 can be completed locally on your computer

Documents:

- IPython Notebook, published and viewable online.
- PDF export of IPython Notebook.

2 Useful References

Back to Table of Contents

- See lecture 1

HW Problems

Back to Table of Contents

3. HW1.0

Back to Table of Contents

HW1.0.1. Self-Introduction

W1.0.0 Prepare your bio and include it in this HW submission. Please limit to 100 words. Count the words in your bio and print the length of your bio (in terms of words) in a separate cell.

Fill in the following information [Optional]

- Your Location
- When did you start MIDS and what is your target finish date
- What you want to get out of w261?

Razib Shishir completed his PhD in Electrical Engineering from Arizona State University in 2009. Then he joined as a Post Doc Fellow at University of South Carolina. In 2010, he joined at Intel Corp and currently working as a Device Engineer. Razib is responsible for process control based on electrical measurement data by identifying, investigating and reacting to process deviations and equipment mismatches in the FAB. He works with a lot of data on a daily basis and he developed many scripts to automate this process. He recently enrolled in UC Berkeley's MIDS program with expected graduation date December 2017.

HW1.0.2. Big data

Define big data. Provide an example of a big data problem in your domain of expertise.

Based on today's technology, a laptop can have 1TB storage. So data size on the order of 100 TB or more can be called Big Data. I work as a device Engineer where my responsibility is process control by identifying process variation and equipment mismatches in the FAB. when a wafer goes through different opearations in the FAB, tons of data are collected for that wafer, process and equipment log data for tens of thousands of parameters which make it a Big Data problem.

HW1.0.3. Bias Variance

What is bias-variance decomposition in the context machine learning? How is it used in machine learning?

```
In [ ]:
```

3. HW1.1 WordCount using a single thread

Back to Table of Contents

Write a program called alice_words.py that creates a text file named **alice_words.txt** containing an alphabetical listing of all the words, and the number of times each occurs, in the text version of Alice's Adventures in Wonderland. (You can obtain a free plain text version of the book, along with many others, from here (http://www.gutenberg.org/cache/epub/11/pg11.txt) The first 10 lines of your output file should look something like this (the counts are not totally precise):

Word Count ======================= a 631 a-piece 1 abide 1 able 1 about 94 above 3 absence 1 absurd 2

```
In [3]: # check where is the current directory and change if necessary using something
         like: %cd W261MasterDir
        %cd Documents\MIDS\W261\HW1

        C:\Users\rsshishi\Documents\MIDS\W261\HW1
```

```
In [ ]: # Here is an example of wordcounting with a defaultdict (dictionary structure
         with a nice
        # default behaviours when a key does not exist in the dictionary
        import re
        from collections import defaultdict
        wordCounts=defaultdict(int)
        %cd \C\Users\rsshishi\Documents\MIDS\W261\HW1
        # open the
        f = open('pg19033.txt', 'r')
        for line in f:
            linestr=f.readline()
            for word in re.findall(r'[a-z]+', linestr.lower()):
                wordCounts[word] += 1


        g=open('alice_words.txt','w')
        print("{:20s}{:>7}".format("Word","Count"))
        print("==================================")
        g.write("{:20s}{:>7}".format("Word","Count"))
        g.write("\n")
        g.writelines("==================================\n")
        for key in sorted(wordCounts):
            #print (key, "\t\t",wordCounts[key])
            print("{:20s}{:>7}".format(key,wordCounts[key]))
            g.writelines("{:20s}{:>7}".format(key,wordCounts[key]))
            g.write("\n")
```

In [36]:
```
#display the first few lines
!head alicesTExtFilename.txt
```

Dictionaries are a good way to keep track of word counts

wordCounts={}

defaultdict are slightly more effectice way of doing word counting

One way to do word counting but not best. A defaultdict is like a regular dictionary, except that when you try to look up a key it doesn’t contain, it first adds a value for it using a zero-argument function you provided when you created it. In order to use defaultdicts, you have to import them

In [5]:

```
# Here is an example of wordcounting with a defaultdict (dictionary structure
 with a nice
# default behaviours when a key does not exist in the dictionary
import re
from collections import defaultdict
wordCounts=defaultdict(int)
# open the text file of the book
f = open('pg19033.txt', 'r')
for line in f:
    linestr=f.readline()
    for word in re.findall(r'[a-z]+', linestr.lower()):
        wordCounts[word] += 1


g=open('alice_words.txt','w')
print("{:20s}{:>7}".format("Word","Count"))
print("=================================")
g.write("{:20s}{:>7}".format("Word","Count"))
g.write("\n")
g.writelines("==================================\n")
for key in sorted(wordCounts):
    #print (key, "\t\t",wordCounts[key])
    print("{:20s}{:>7}".format(key,wordCounts[key]))
    g.writelines("{:20s}{:>7}".format(key,wordCounts[key]))
    g.write("\n")
```

```
Word                  Count
==================================
a                       169
about                    19
accept                    1
accepted                  1
access                    6
accordance                2
accusation                1
accustomed                1
across                    3
active                    1
actual                    1
actually                  1
added                     4
addition                  1
additional                1
additions                 1
addressed                 1
adjourn                   1
adoption                  1
adventures                1
advisable                 2
advise                    1
affectionately            1
afore                     1
afraid                    3
after                    12
afterwards                1
again                    11
against                   4
aged                      1
agree                     5
agreement                 8
ah                        1
ahem                      1
air                       4
ak                        1
alarm                     1
alice                    85
all                      39
allow                     1
almost                    1
alone                     2
along                     1
also                      1
alteration                1
alternate                 1
altogether                1
always                    2
am                        2
among                     4
an                       11
and                     207
anger                     1
angry                     2
animal                    2
```

animals	3
another	6
answer	1
anxiously	2
any	27
anyone	2
anything	7
anywhere	1
appear	1
appearance	1
appearing	1
appears	1
approach	1
archive	9
are	20
arise	1
arm	3
arms	3
array	1
arrived	1
as	51
asked	2
asking	3
asleep	1
assembled	2
associated	3
at	45
ate	1
atheling	1
atom	1
attached	1
attending	1
away	11
awhile	2
b	2
baby	3
back	8
bank	1
banks	1
barrowful	2
based	2
be	23
beautiful	1
beautifully	1
because	2
bed	1
been	5
before	8
began	13
begged	1
begin	3
beginning	3
begun	1
beheading	1
behind	3
being	1
believe	3

below	2
bend	1
best	2
better	1
between	1
bill	3
birds	4
bit	4
blasts	1
blew	2
blow	1
blows	1
blue	1
book	3
both	3
bottle	6
bough	1
box	3
brandy	1
brass	1
bread	1
breath	1
breathe	1
bright	2
bring	1
bringing	1
broken	1
brought	1
brushing	1
burning	1
business	2
but	26
butter	2
buttered	1
by	23
c	2
cake	1
calculated	1
call	3
called	4
calling	1
came	12
can	13
cannot	2
card	1
cards	1
care	3
carefully	1
cares	1
carried	1
carroll	1
carrying	1
cat	8
caterpillar	7
cats	3
cause	3
caused	1

cease	1
certain	1
certainly	2
chain	1
chains	1
chanced	1
change	1
changed	1
charge	2
check	1
checks	1
cherry	1
cheshire	3
children	1
chimney	3
chin	2
choked	1
choose	1
chorus	2
circle	1
city	1
claim	1
close	3
closely	1
clubs	1
collection	4
come	8
comes	2
comfits	2
coming	2
committed	1
commotion	1
compilation	1
complaining	1
completely	1
compliance	3
comply	2
complying	2
compressed	1
computers	1
concept	1
concluded	1
condemn	1
confirmed	1
confusing	1
confusion	2
conqueror	2
conquest	1
consequential	1
consider	1
considering	1
constant	1
contact	1
contain	1
containing	1
continued	1
conversation	2

conversations	2
cook	3
cool	2
copied	1
copies	4
copy	6
copying	2
copyright	6
corner	1
cost	1
could	16
countries	1
country	2
course	3
court	6
created	1
creating	1
creation	1
creature	2
credit	1
cried	4
croquet	2
croqueting	1
cross	2
crossly	1
crouched	1
crowd	2
crowded	3
crown	1
cry	3
cur	1
curiosity	1
curious	3
curls	1
current	1
curtain	1
cushion	1
custard	1
d	3
daisy	1
damage	1
damaged	1
damages	1
dark	1
darkness	1
day	4
days	1
dead	2
deal	3
dear	9
dears	1
death	1
decided	1
deductible	1
defective	3
defects	1
deletions	1

delight	1
demand	1
depends	1
derivative	2
desk	1
despite	1
destroy	2
determine	1
diamonds	1
did	8
didn	2
digging	2
dinah	2
direct	1
direction	2
directions	2
directly	2
disappointment	1
disclaim	1
disclaimer	3
disclaimers	1
discover	1
discovered	1
dishes	1
disk	1
distance	2
distribute	2
distributed	3
distributing	4
distribution	4
do	25
dodo	3
does	3
doesn	1
dogs	1
doing	2
domain	5
don	7
donate	1
donation	1
donations	6
done	2
door	11
doors	1
dormouse	2
double	1
doubt	1
doubtfully	1
down	23
dr	1
draw	1
dreadfully	2
dream	1
drew	1
dried	1
drink	3
dripping	1

drop	1
drunk	1
dry	2
duchess	9
duck	3
e	6
each	3
eager	1
ear	1
earls	1
easily	1
easy	1
eat	5
ebook	6
ebooks	4
edgar	1
edition	1
editions	3
edwin	1
effort	1
efforts	1
eggs	2
either	3
elbow	1
elect	1
electronic	18
else	3
email	2
emphasis	1
employees	1
empty	1
end	4
energetic	1
engaged	1
english	1
enough	2
ensuring	1
entangled	1
entrance	1
equipment	3
escape	2
even	2
ever	4
every	3
everything	4
evidence	2
exactly	2
examine	1
except	2
executed	1
execution	1
exempt	1
exists	1
explain	1
exporting	1
extent	1
eyes	3

f	7
face	5
fact	1
fairbanks	1
fall	1
fallen	3
falling	1
familiarly	1
family	1
fan	6
fancied	1
fancy	1
far	1
farther	1
fast	1
favored	1
fear	1
fee	2
feelings	1
fees	2
feet	5
felt	2
ferrets	1
fetch	1
few	2
field	1
fighting	1
file	1
files	1
financial	1
find	4
finding	1
fine	1
finger	1
finished	1
first	8
fish	1
fitness	1
fitted	1
five	1
flamingos	1
flashed	1
flinging	1
floor	2
followed	2
following	1
fond	2
foot	4
footman	5
footsteps	1
for	34
forgetting	1
forgot	1
forgotten	2
form	2
format	3
forth	2

found	13
foundation	16
fountains	2
four	1
free	4
freely	1
french	1
fright	1
frightened	2
frog	2
from	14
front	2
full	13
fur	1
further	1
fury	1
future	1
gallons	1
game	1
garden	9
gardeners	1
gave	3
gbnewby	1
general	2
generally	2
gently	1
get	10
getting	3
girl	2
give	3
glad	2
glass	3
gloves	7
go	14
goals	1
goes	1
going	6
golden	3
gone	3
good	3
got	7
grand	1
granted	1
gratefully	1
gravely	1
great	8
green	1
gregory	1
grew	1
grinned	2
grins	1
ground	3
grow	4
growing	3
grown	1
guard	1
guess	1

guessed	1
gutenberg	54
had	37
half	6
hall	3
hand	6
handed	1
hands	3
hanging	1
happen	1
happened	1
happens	2
hard	2
hare	4
hart	1
has	2
hastily	6
hatching	1
hated	1
hatter	4
have	11
having	1
he	13
head	12
heads	1
heap	1
hear	3
heard	9
hearing	1
hearts	1
hedge	1
hedgehog	2
hedgehogs	2
height	2
held	1
help	2
helpless	1
her	59
herald	1
here	6
hers	1
herself	22
high	7
highest	1
him	2
his	11
hiss	1
history	2
hit	1
hold	3
hole	1
honor	1
hookah	2
hopeless	1
hoping	1
hot	2
hour	2

house	5
housemaid	1
how	9
however	4
howling	2
http	4
hung	1
hurried	3
hurriedly	1
hurry	2
hurt	2
i	84
idea	3
identify	1
if	30
ii	1
illustration	14
immediate	2
impatiently	1
implied	1
important	2
impossible	1
in	124
inaccurate	1
inches	3
include	1
included	1
including	5
incomplete	1
increasing	1
indeed	2
indemnify	1
indemnity	1
indicating	1
indignantly	2
indirect	1
indirectly	1
individual	1
information	3
infringement	1
inquisitively	1
inside	1
instantly	1
insult	1
intellectual	1
internal	1
international	1
interpreted	1
into	18
inwards	1
irma	1
irs	1
is	34
isbell	1
isn	1
it	97
its	15

itself	1
iv	1
ix	1
jar	2
jason	1
joined	1
judge	1
judging	1
jumping	2
jurors	1
jury	2
jurymen	1
just	12
keep	2
keeping	2
kept	2
key	6
kid	4
kill	1
king	13
kitchen	1
knave	1
knee	1
kneel	1
knew	2
knocked	1
knocking	2
know	9
knuckles	1
label	1
ladder	1
lake	1
lamps	1
lap	1
large	10
larger	3
last	6
late	1
later	1
laws	4
lay	2
leap	1
learn	1
least	4
leave	3
leaves	2
led	3
left	6
legally	1
legged	1
legs	1
less	1
lesson	1
lest	1
let	2
letter	2
lewis	1

liable	1
library	1
license	8
licensed	1
lieu	1
life	2
like	11
liked	1
limitation	2
limited	3
links	1
listen	1
literary	10
little	31
live	1
livery	2
lives	2
ll	14
located	1
locations	1
lock	1
locks	1
long	8
longed	1
look	1
looked	7
looking	5
loose	1
lost	2
lot	1
loudly	2
love	1
loveliest	1
lovely	1
low	3
luckily	1
lying	2
m	14
mad	3
made	7
majesty	2
make	6
makes	2
making	3
mallets	1
man	1
many	1
march	3
marked	3
master	1
matter	1
maximum	1
may	8
me	15
mean	3
meaning	2
means	3

meant	2
measure	1
medium	4
meet	2
meeting	1
melancholy	1
merchantibility	1
met	2
method	1
middle	5
might	2
mile	2
miles	1
milk	1
mind	6
mine	1
minute	3
minutes	3
mischief	1
miss	2
mission	2
mississippi	1
mistake	1
modification	1
moment	8
money	2
moral	1
morcar	1
more	16
morning	3
morsel	1
most	2
mostly	1
mouse	14
mouth	2
moved	1
much	4
murder	1
mushroom	4
must	15
my	10
myself	3
n	2
name	2
named	1
narrow	1
nay	1
near	5
nearer	1
nearly	1
neat	1
needs	1
negligence	1
neither	1
nervous	1
nest	1
net	1

network	1
never	8
new	2
newby	1
newsletter	1
next	6
nibbled	2
nibbling	2
nice	1
no	19
noise	2
non	1
nonproprietary	1
nonsense	2
nor	1
north	1
nose	1
not	33
note	1
nothing	6
notice	3
noticed	5
notifies	1
now	13
nowhere	1
number	3
nursing	1
o	1
obsolete	1
obtain	3
obtaining	2
occasionally	1
of	161
off	15
offended	3
offer	1
offers	1
official	2
oh	7
old	3
on	45
once	4
one	18
ones	1
online	3
only	11
opened	5
opportunity	2
or	54
org	6
other	15
others	2
ought	3
our	5
out	29
outdated	1
outside	4

over	3
overhead	1
owed	1
own	2
owner	3
owns	2
pack	1
page	1
pages	1
paid	3
painting	2
pair	3
pale	1
panting	1
paper	4
paragraph	4
paragraphs	1
parchment	1
pardoned	1
part	3
particular	2
party	5
passage	3
patted	1
pattering	2
paw	1
paying	2
payments	2
peeped	1
pegs	1
people	5
pepper	4
performances	1
performing	1
perhaps	1
periodic	1
permission	3
permitted	1
person	1
pgdp	1
pglaf	3
phrase	3
physical	2
picked	2
pictures	2
piece	2
pig	1
pigeon	3
pineapple	1
places	1
plainly	1
plates	1
play	1
players	1
playing	1
pleasant	1
please	3

plenty	2
pocket	1
poison	1
pool	4
poor	8
pope	1
possessed	1
possible	1
posted	4
pot	1
practically	1
prepare	2
preserve	1
pressed	2
pressing	1
prevent	1
previous	1
printed	1
prison	1
procession	1
produced	1
profit	1
prohibition	1
project	52
prominently	1
promised	1
promoting	1
promotion	1
proofread	1
proofreading	1
proper	1
property	2
proprietary	1
prosecute	1
prove	1
provide	5
provided	3
providing	2
provision	1
provisions	1
public	5
pulled	1
pulling	1
punitive	1
purpose	1
put	6
puzzled	1
quarrel	1
quarreling	1
queen	14
queens	1
queer	2
question	6
quick	1
quite	12
rabbit	14
race	1

raising	1
ran	5
rapidly	1
rate	4
rather	2
rats	1
raven	1
re	11
reach	2
read	4
reading	3
ready	2
really	2
receive	1
received	1
recognized	1
red	2
redistribution	2
references	1
refreshments	1
refund	6
registered	1
regulating	1
relieved	1
remained	1
remaining	1
remark	1
remarkable	2
remarked	1
remarking	1
remedies	1
remember	1
remembered	1
repeated	2
replace	1
replacement	2
replied	3
reported	1
reports	1
required	1
requirements	3
research	1
restrictions	1
return	2
riddle	1
riddles	2
right	8
roared	1
roast	1
roof	2
room	3
roots	1
rose	2
roses	2
round	9
royal	1
royalties	1

royalty	1
rules	1
running	4
rush	1
s	35
sad	2
sadly	1
safe	1
said	68
salt	2
same	8
sat	2
saucer	1
savage	1
save	1
saves	1
saw	2
say	4
saying	1
scattered	1
scolded	1
scrambling	1
scratching	1
scream	1
scroll	1
sea	2
search	1
second	1
secondly	1
section	2
secure	1
see	11
seem	1
seemed	4
seen	2
send	1
sending	1
sends	1
sensation	2
sense	1
sent	1
sentence	2
serpent	3
set	6
seven	1
severely	3
sha	2
shall	2
shared	1
sharp	1
sharply	1
she	135
shedding	1
shelves	1
shoes	1
shook	2
shore	2

short	1
should	4
shoulders	1
shouldn	1
shrieks	1
shrill	1
shrinking	3
shut	1
side	3
sides	2
sight	1
signed	2
silence	4
since	2
sir	2
sister	3
sit	1
site	2
sitting	3
size	6
skurried	1
sky	1
sleep	2
slipped	1
slowly	1
small	8
snatch	2
sneeze	1
sneezing	2
so	21
softly	1
soldier	1
soldiers	2
solemn	2
solicit	1
solicitation	1
some	14
somehow	1
something	7
sometimes	1
soon	9
soothing	1
sorrowful	1
sort	3
sorts	1
sounds	1
soup	2
speak	1
speaker	1
special	1
speed	1
spehar	1
spoke	5
staff	1
stairs	1
stalk	1
standing	1

start	2
started	1
state	2
states	7
status	4
sticks	1
still	1
stockings	1
stole	1
stood	3
stool	1
stop	1
stopping	1
straight	1
stretched	2
strict	1
struck	2
stuff	1
subject	1
submitted	1
subscribe	1
succeeded	2
such	12
sudden	2
suddenly	3
summer	1
support	2
suppose	1
sure	4
surprise	1
surprised	1
survive	1
swallow	1
swam	3
swamp	1
swim	1
synonymous	1
t	25
table	5
tail	2
take	6
taken	1
takes	1
tale	2
talk	2
talking	6
taller	1
tart	1
tarts	2
tax	3
tea	3
team	1
tears	3
telescope	1
telescopes	1
tell	3
temper	1

ten	4
terms	9
than	5
thank	1
that	60
the	417
their	5
theirs	1
them	17
then	21
there	25
these	5
they	20
thick	1
thing	7
things	3
think	8
thinking	1
this	54
those	3
though	1
thought	7
thousand	1
three	8
threw	1
through	6
throwing	1
thump	2
tidy	1
till	3
time	14
times	1
timid	2
timidly	2
tiny	2
tired	4
tis	2
tm	31
to	156
toffy	1
together	4
told	1
tone	8
too	5
took	9
top	2
trademark	3
transcribe	1
tree	5
trees	3
tremble	1
trial	5
tried	5
trotting	1
trouble	2
trumpet	1
trying	4

tucked	1
tumbling	1
tunnel	1
turkey	1
turned	4
turning	3
two	7
txt	1
u	2
ugh	1
uncomfortable	1
uncorked	1
under	3
underneath	1
understand	1
uneasily	1
unfolded	2
uniform	1
united	6
unpleasant	1
until	1
untwist	1
up	20
updated	1
upon	8
upright	1
upsetting	1
upstairs	1
us	2
use	5
used	3
user	1
using	4
usual	1
usurpation	1
ut	1
vanished	2
various	1
ve	8
venture	1
ventured	2
verdict	2
very	18
vi	1
violates	1
visit	1
voice	5
voices	1
void	1
volunteer	1
volunteers	3
waited	1
waiting	3
walked	3
walks	1
wandered	1
want	3

warranties	1
warranty	1
was	77
wasn	1
wasting	1
watch	1
water	2
waving	1
way	10
ways	1
we	10
web	4
well	9
went	16
were	16
west	1
wet	2
what	17
whatever	1
whatsoever	1
when	16
where	3
whether	2
which	15
while	2
whilst	1
whiskers	1
whispers	1
white	11
who	12
whoever	1
whole	3
whose	1
why	7
wide	1
will	8
william	2
wind	1
window	4
wings	1
wink	1
wish	3
with	55
within	3
without	7
witness	4
wits	1
won	4
wonder	2
wonderland	4
wood	4
words	5
wore	1
work	22
works	20
world	2
worm	1

```
worse                      1
worth                      1
would                      5
wouldn                     2
wriggling                  1
writing                    3
www                        5
x                          1
yards                      1
ye                         1
yer                        1
yes                        2
yet                        2
you                       83
your                      11
yourself                   1
zip                        1
```

HW1.1.1 How many times does the word alice occur in the book?

In [6]:
```
print(wordCounts["alice"])
```

```
85
```

3. HW1.2 Command Line Map Reduce Framework

Back to Table of Contents

Read through the provided mapreduce shell script (pWordCount.sh) provided below and all of its comments. When you are comfortable with their purpose and function, respond to the remaining homework questions below. Run the shell without any arguments.

In [10]:

```
%%writefile pWordCount.sh
#!/bin/bash
## pWordCount.sh
## Author: James G. Shanahan
## Usage: pWordCount.sh m wordlist testFile.txt
## Input:
##       m = number of processes (maps), e.g., 4
##       wordlist = a space-separated list of words in quotes, e.g., "the and
 of"
##       inputFile = a text input file
##
## Instructions: Read this script and its comments closely.
##               Do your best to understand the purpose of each command,
##               and focus on how arguments are supplied to mapper.py/reducer.
py,
##               as this will determine how the python scripts take input.
##               When you are comfortable with the unix code below,
##               answer the questions on the LMS for HW1 about the starter cod
e.


usage()
{
    echo ERROR: No arguments supplied
    echo
    echo To run use
    echo "     pWordCount.sh m wordlist inputFile"
    echo Input:
    echo "      number of processes/maps, EG, 4"
    echo "      wordlist = a space-separated list of words in quotes, e.g., 't
he and of'"
    echo "      inputFile = a text input file"
}

if [ $# -eq 0 ]
  then
    usage
    exit 1
fi

## collect user input
m=$1 ## the number of parallel processes (maps) to run

wordlist=$2 ## if set to "*", then all words are used

## a text file
data=$3

## 'wc' determines the number of lines in the data
## 'perl -pe' regex strips the piped wc output to a number
linesindata=`wc -l $data | perl -pe 's/^.*?(\d+).*?$/$1/'`

## determine the lines per chunk for the desired number of processes
linesinchunk=`echo "$linesindata/$m+1" | bc`

## split the original file into chunks by line
split -l $linesinchunk $data $data.chunk.
```

```
## assign python mappers (mapper.py) to the chunks of data
## and emit their output to temporary files
for datachunk in $data.chunk.*; do
    ## feed word list to the python mapper here and redirect STDOUT to a tempo
rary file on disk
    ####
    ####
    ./mapper.py  "$wordlist" <$datachunk > $datachunk.counts &
    ####
    ####
done
## wait for the mappers to finish their work
wait

###-----------------------------------------------------------------------------
-------------
#TODO
#Insert a sort -k1,1 here to collate wordCount records with the same key (i.
e., same word)
#
###-----------------------------------------------------------------------------
-------------



## 'ls' makes a list of the temporary count files
## 'perl -pe' regex replaces line breaks with spaces
countfiles=`\ls $data.chunk.*.counts | perl -pe 's/\n/ /'`

## feed the list of countfiles to the python reducer and redirect STDOUT to di
sk
####
####
./reducer.py <$countfiles > $data.output
####
####

## clean up the data chunks and temporary count files
\rm $data.chunk.*
```

```
Overwriting pWordCount.sh
```

In [78]:
```
!head pWordCount.sh
```

```
#!/bin/bash
## pWordCount.sh
## Author: James G. Shanahan
## Usage: pWordCount.sh m wordlist testFile.txt
## Input:
##       m = number of processes (maps), e.g., 4
##       wordlist = a space-separated list of words in quotes, e.g., "the and
 of"
##       inputFile = a text input file
##
## Instructions: Read this script and its comments closely.
```

In [ ]:
```
!chmod a+x pWordCount.sh
! ./pWordCount.sh
```

If pWordCount.sh is ran without any arguments, then it shows follwoing messages

ERROR: No arguments supplied

To run use

```
pWordCount.sh m wordlist inputFile
```

Input:

```
 number of processes/maps, EG, 4

 wordlist = a space-separated list of words in quotes, e.g., 'the and of'

 inputFile = a text input file
```

Please feel free to adopt and modify the following mapper for your purpose¶

In [75]:
```
%%writefile mapper.py
#!/usr/bin/python
import sys
import re
count = 0
#filename = sys.argv[2]
findword = sys.argv[1]
for line in sys.stdin:
    if findword.lower() in line.lower():
        count = count + 1
print count
```

```
Writing mapper.py
```

Please feel free to adopt and modify the following reducer for your purpose¶

(i.e., there will be no need for a sort in reducer.py code; leverage mapreduce framework).

In [ ]:
```
%%writefile reducer.py
#!/usr/bin/python
import sys
sum = 0
for countStr in sys.stdin:
    sum = sum + int(countStr)
print sum
```

Dont forget to add a sort component to your MapReduce framework and leverage the sort order in your reduceer (i.e., there will be no need for a sort in reducer.py).

I.e., insert code

In [ ]:
```
!chmod a+x mapper.py
!chmod a+x reducer.py
!chmod a+x pWordCount.sh
! ./pWordCount.sh
```

3. HW1.3 WordCount via Command Line Map Reduce Framework

Back to Table of Contents

Write the mapper.py/reducer.py combination to perform WordCount using the command line mapreeduce framework containing an alphabetical listing of all the words, and the number of times each occurs, in the text version of Alice's Adventures in Wonderland. (You can obtain a free plain text version of the book, along with many others, from here (http://www.gutenberg.org/cache/epub/11/pg11.txt) The first 10 lines of your output file should look something like this (the counts are not totally precise):

To do so, make sure of the following:

- That the mapper.py counts all occurrences of a single word
- In the pWordCount.sh, please insert a sort command between the mappers (after the for loop) and the reducer calls to collate the output key-value pair records by key from the mappers. E.g., sort -k1,1. Use "man sort" to learn more about Unix sorts.
- reducer.py sums the count value from the collated records for each word. There should be no sort in the reducer.py

Word Count ====================== a 631 a-piece 1 abide 1 able 1 about 94 above 3 absence 1 absurd 2

Here, mapper.py will read in a portion (i.e., a single record corresponding to a row) of the email data, count the number of occurences of the word in questions and print/emit a count to the output stream. While the utility of the reducer responsible for reading in counts of the word and summarizing them before printing that summary to the output stream. See example the notebook (http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/5zq0faibmvtjlbr/DivideAndConquer2-python-Plus-CmdLine.ipynb) See video section 1.12.1 1.12.1 Poor Man's MapReduce Using Command Line (Part 2) located at: https://learn.datascience.berkeley.edu/mod/page/view.php?id=10961 (https://learn.datascience.berkeley.edu/mod/page/view.php?id=10961)

NOTE in your python notebook create a cell to save your mapper/reducer to disk using magic commands (see example here)

In [ ]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: XYZ
## Description: mapper code for HW1.2-1.5

import sys
import re
count = 0

## collect user input
filename = sys.argv[1]
findwords = re.split(" ",sys.argv[2].lower())
..........
```

In the next cell use the Unix chmod command to change the permissions of the mapper/reducer using the following commands:

In [ ]:
```
!chmod +x mapper.py;
!chmod +x reducer.py
```

In [ ]:
```
```

3. HW1.4

Back to Table of Contents

Change the mapper.py/reducer.py combination so that you get only the number of words starting with an uppercase letter, and the number of words starting with a lowercase letter for Alice in Wonderland available here (http://www.gutenberg.org/cache/epub/11/pg11.txt). In other words, you need an output file with only 2 lines, one giving you the number of words staring with a lowercase ('a' to 'z'), and the other line indicating the number of words starting with an uppercase letter ('A' to 'Z'). In the pWordCount.sh, please insert a sort command between the mappers (after the for loop) and the reducer calls to collate the output key-value pair records by key from the mappers. E.g., sort -k1,1. Use "man sort" to learn more about Unix sorts.

```
In [ ]:
```

3. HW1.5 Bias-Variance (This is an OPTIONAL HW)

Back to Table of Contents

Provide and example of bias variance in action for a similated function y = f(x). E.g., y = sin(x+x^2). Provide code, data, and graphs.

Using a bias-variance decomposition analsysis on your choosen problem, describe how you would decide which model to choose when you dont know the true function and how does this choice compares to the choice you made using the true function.

```
In [ ]:
```

Back to Table of Contents

------- END OF HOWEWORK --------