Table of Contents

1 MIDS - w261 Machine Learning At Scale
1.1 Assignment - HW1
2 Table of Contents
3
3.0.1 IMPORTANT
3.0.2 === INSTRUCTIONS for SUBMISSIONS ===
4
5
5.1 HW1.0
5.1.1 HW1.0.1. Self-Introduction
5.1.2 HW1.0.2. Big data
5.1.3 HW1.0.3. Bias Variance
5.2 3. HW1.1 WordCount using a single thread
5.2.1 Dictionaries are a good way to keep track of word counts
5.2.2 defaultdict are slightly more effectice way of doing word counting
5.2.3 Fill in the code block below, then execute the cell as well as the cell below it.
5.2.3.1 This will write a file named alice_words.py and run it.
5.2.3.2 the output per line should be a tab separated key-value pair with the following format WORD TAB count
5.2.4 Pretty print top 10 results from alice_words.txt
5.2.5 HW1.1.1 How many times does the word alice occur in the book?
5.2.5.1 HINT: read the file with python, or use subprocess to access the commandline from within the python function
5.2.5.2 As before, fill in the code block below, then execute the cell as well as the cell below it. This will write a file named hw111.py and run it.
5.3 3. HW1.2 Command Line Map Reduce Framework
5.3.1 Test the framework without parameters:
5.3.2 Run the following two cells to generate mapper and reducer files, then run the shell script again with arguments.¶
5.3.3 Make the files executable:
5.3.4 Test the framework with parameters:
5.4 3. HW1.3 WordCount via Command Line Map Reduce Framework
5.4.1 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).
5.4.2 Run the command below with 4 mappers. You should get the same result as in HW1.1.1.
5.5 3. HW1.4 - Count words staring with uppercase and words starting with lowercase (This is an OPTIONAL HW)
5.6 3. HW1.5 Bias-Variance (This is an OPTIONAL HW)
5.7 ------- END OF HOWEWORK --------

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-w
ith-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: Nilesh Bhoyar
Class: MIDS w261 (Summer 2017 Group 2)
Email: nilesh.bhoyar@iSchool.Berkeley.edu
StudentId 26302327 **End of StudentId**
Week: 1

NOTE please replace `1234567` with your student id above
Due Time: HW is due the Tuesday of the following week by 8AM (West coast time). I.e., Tuesday,Jan 17, 2017 in the case of this homework. Please note a late submission may result in a ZERO grade for this homework.

Table of Contents

1. HW Intructions
2. HW References
3. HW Problems
4. HW Introduction
5. HW References
6. HW Problems
   1.0. HW1.0
   1.0. HW1.1
   1.2. HW1.2
   1.3. HW1.3
   1.4. HW1.4
   1.5. HW1.5

1 Instructions

Back to Table of Contents

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

Version 2017-01-12

IMPORTANT

HW1 can be completed locally on your computer. You will not need any other python modules than the ones provided. For example, you should not have to import numpy.

Homeworks will be graded using a combination of manual review and unit tests. Where applicable, please make sure to enter your solution between the comments:
`# START STUDENT CODE`
`# END STUDENT CODE.`
For questions that contain the above comments, do not modify anything else in the code block.

Create a HW1 directory on your hard-drive, and place this notebook into it. Rename it to include your UCB student id. All files output by the code blocks should also reside at the root of your HW1 directory. It might look something like this:

```
localhost:HW1 $ pwd
/Users/johndoes/Documents/UCBerkeley/261/HW1

localhost:HW1 $ tree
    ├── MIDS-W261-HW-01-yourUCBid.ipynb
    ├── alice_words.py
    ├── alice_words.txt
    ├── alicesTExtFilename.txt
    ├── alicesTExtFilename.txt.output
    ├── hw11.py
    ├── hw111.py
    ├── hw111.pyc
    ├── mapper.py
    ├── pWordCount.sh
    ├── reducer.py

    0 directories, 11 files
```

=== INSTRUCTIONS for SUBMISSIONS ===

Follow the instructions for submissions carefully.

Click this link to enable you to create a github repo within the MIDS261 Classroom: https://classroom.github.com/assignment-invitations/51317dde89d412134a749f035a0d59d3 (https://classroom.github.com/assignment-invitations/51317dde89d412134a749f035a0d59d3) , and follow the instructions to create a HW1 repo.

Push the following to your HW1 github repo into the master branch:

- PDF export of IPython Notebook.
- The contents of your local HW1 directory.

2 Useful References

Back to Table of Contents

- See lecture 1

3 HW Problems

Back to Table of Contents

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?

```
Answer:
I am in Parsippany, New Jersey.  I started this program in Jun,2015. I wil
l be completeting this course by end of this year.
I learnt few concepts in machine learning via W207 and in depth NLP via W2
61 course. NLP problems are also mostly large scale machine learning probl
ems where you have huge data and complex data structures.  I realized the
 need to taking this course - I intend to spend tons of time on understand
ing math behind alorithms and by end of the course I want to my self as da
ta scientist with solid math and skills to executre model on large scale.
```

HW1.0.2. Big data

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

```
Answer :

Big data refers to data sets whose size is beyond the ability of typical d
atabase software tools to capture, store, manage and analyze. But big data
 does not mean just boost existing data from giga byte to tera byte by rep
licating it multiple times.  Big data is collection of datasets some of wh
ich can be structured and unstructured , some of it could  be online and s
ome of it could be really a paper. Datasets could be transactional in natu
re or master data as well. so in general,clunky,messy and hard to use in E
XISTING SYSTEMS. For same reason we also hear big data contains volume, ve
locity, and variety.

So big data definition will change over the time. As technology advances,
 what is big data today won't be tomorrow. I work in insurance practice mo
stly around premium collections and claim disbursement. We are data native
 company where our acturial calculations depend on predicting loss for per
 dollar premium earned.We have to frequently build models to predict losss
es by analyzing the claims and policy details for past more then 30 years.
  For claims many times we will have images of catostrophic losses or indi
vidual accidents, claim can have structured data about every loss occured
 or it can be a pdf document with more than 200 pages describing some even
t.  So all losses we have paid in last 30 years is really big data and if
 we are able to squeez all insights from it , we can really have big compa
titive advantage.
```

HW1.0.3. Bias Variance

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

```
Answer
Performance measure for regression problems is the Root Mean Square Error
 (RMSE). It measures the standard deviation of the errors the system makes
 in its predictions.
RMSE can also be called as generalization error and can be expressed as

RMSE = Reducible Error + Irreducible error
     = ( Bias Error + Variance) + Irreducible error
```

Bias Error

```
This occurs is generalization error due to wrong assumption. Assuming data
 as linear while in actual it is quadratic.High bias model will underfit t
he data. I have explained this in detail with example in last section of t
his notebook.
```

Variance

```
This part is due to the model's excessive sensitivity to small variations
 in the training data. A model with many degrees of freedom  is likely to
 have high variance, and thus to overfit the training data.
```

Irreducible Error

```
This is due to noise in data itself.

Increasing the model complexity will reduce the bias but will increase the
 variance. so there is trade off when we think of bias and variance.  Vali
dation error will go down as we increase the complexity of model and after
 some time it will increase due to variance. There are various techniques
 to analyze such overfitting.
```

Some math on bias/Variance/Irreducible erro

```
Generalization error is calculated as
```

$$Err(x_0) = E[(Y - \hat{f}(x_0))^2]$$

```
It is possible to decompose this error in its Bias and Variance component
s:
```

$$Err(x_0) = \left(E[\hat{f}(x_0)] - f(x_0)\right)^2 + E\left[\hat{f}(x_0) - E[\hat{f}(x_0)]\right]^2 + \sigma_\epsilon^2$$
$$Err(x_0) = Bias^2 + Variance + IrreducibleError$$

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 tab separated 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):

```
a          690
abide      2
able       1
about      102
above      3
absence    1
absurd     2
accept     1
acceptance 1
accepted   2
```

In [1]:

```
# Make sure you are in HW1 directory
!pwd
```

```
/media/notebooks/DataScience/W261/homework/HW1
```

In [2]:

```
# !curl 'http://www.gutenberg.org/cache/epub/11/pg11.txt' -o alicesTExtFilename.
txt
# sometimes the above link produces junk characters. However, the direct link wo
rks as expected:
!curl 'http://www.gutenberg.org/files/11/11-0.txt' -o alicesTExtFilename.txt
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Ti
me  Current
                                 Dload  Upload   Total   Spent    Le
ft  Speed
100  169k  100  169k    0     0   223k      0 --:--:-- --:--:-- --:-
-:--  232k
```

In [3]:

```
#display the first few lines
!head alicesTExtFilename.txt
```

In [4]:

```
#example of a regular expression to detect words in a string.
import re
line = """ 0017.2000-01-17.beck 0       global risk management operations
        " congratulations, sally!!!  kk  ----------------------forwarded by kath
y kokas/corp/enron on 01/17/2000  08:08 pm---------------------------  from: ric
k causey 01/17/2000 06:04 pm  sent by: enron announcements  to: all enron worldw
ide  cc:  subject: global risk management operations  recognizing enron , s incr
easing worldwide presence in the wholesale energy  business and the need to insu
re outstanding internal controls for all of our  risk management activities, reg
ardless of location, a global risk management  operations function has been crea
ted under the direction of sally w. beck,  vice president. in this role, sally w
ill report to rick causey, executive  vice president and chief accounting office
r.  sally , s responsibilities with regard to global risk management operations
 will mirror those of other recently created enron global functions. in this  ro
le, sally will work closely with all enron geographic regions and wholesale  com
panies to insure that each entity receives individualized regional support  whil
e also focusing on the following global responsibilities:  1. enhance communicat
ion among risk management operations professionals.  2. assure the proliferation
 of best operational practices around the globe.  3. facilitate the allocation o
f human resources.  4. provide training for risk management operations personne
l.  5. coordinate user requirements for shared operational systems.  6. oversee
 the creation of a global internal control audit plan for risk  management activ
ities.  7. establish procedures for opening new risk management operations offic
es  and create key benchmarks for measuring on-going risk controls.  each region
al operations team will continue its direct reporting relationship  within its b
usiness unit, and will collaborate with sally in the delivery of  these critical
 items. the houston-based risk management operations team under  sue frusco , s
 leadership, which currently supports risk management activities  for south amer
ica and australia, will also report directly to sally.  sally retains her role a
s vice president of energy operations for enron  north america, reporting to the
 ena office of the chairman. she has been in  her current role over energy opera
tions since 1997, where she manages risk  consolidation and reporting, risk mana
gement administration, physical product  delivery, confirmations and cash manage
ment for ena , s physical commodity  trading, energy derivatives trading and fin
ancial products trading.  sally has been with enron since 1992, when she joined
 the company as a  manager in global credit. prior to joining enron, sally had f
our years  experience as a commercial banker and spent seven years as a register
ed  securities principal with a regional investment banking firm. she also owned
  and managed a retail business for several years.  please join me in supporting
 sally in this additional coordination role for  global risk management operatio
ns."""
re.findall(r'[a-z]+', line.lower()) [0:10]
```

Out[4]:

```
['beck',
 'global',
 'risk',
 'management',
 'operations',
 'congratulations',
 'sally',
 'kk',
 'forwarded',
 'by']
```

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 wi
th a nice
# default behaviours when a key does not exist in the dictionary
import re
from collections import defaultdict

line = """ 0017.2000-01-17.beck 0       global risk management operations
        " congratulations, sally!!!  kk  ----------------------forwarded by kath
y kokas/corp/enron on 01/17/2000  08:08 pm---------------------------  from: ric
k causey 01/17/2000 06:04 pm  sent by: enron announcements  to: all enron worldw
ide  cc:  subject: global risk management operations  recognizing enron , s incr
easing worldwide presence in the wholesale energy  business and the need to insu
re outstanding internal controls for all of our  risk management activities, reg
ardless of location, a global risk management  operations function has been crea
ted under the direction of sally w. beck,  vice president. in this role, sally w
ill report to rick causey, executive  vice president and chief accounting office
r.  sally , s responsibilities with regard to global risk management operations
 will mirror those of other recently created enron global functions. in this  ro
le, sally will work closely with all enron geographic regions and wholesale  com
panies to insure that each entity receives individualized regional support  whil
e also focusing on the following global responsibilities:  1. enhance communicat
ion among risk management operations professionals.  2. assure the proliferation
 of best operational practices around the globe.  3. facilitate the allocation o
f human resources.  4. provide training for risk management operations personne
l.  5. coordinate user requirements for shared operational systems.  6. oversee
 the creation of a global internal control audit plan for risk  management activ
ities.  7. establish procedures for opening new risk management operations offic
es  and create key benchmarks for measuring on-going risk controls.  each region
al operations team will continue its direct reporting relationship  within its b
usiness unit, and will collaborate with sally in the delivery of  these critical
 items. the houston-based risk management operations team under  sue frusco , s
 leadership, which currently supports risk management activities  for south amer
ica and australia, will also report directly to sally.  sally retains her role a
s vice president of energy operations for enron  north america, reporting to the
 ena office of the chairman. she has been in  her current role over energy opera
tions since 1997, where she manages risk  consolidation and reporting, risk mana
gement administration, physical product  delivery, confirmations and cash manage
ment for ena  , s physical commodity  trading, energy derivatives trading and fin
ancial products trading.  sally has been with enron since 1992, when she joined
 the company as a  manager in global credit. prior to joining enron, sally had f
our years  experience as a commercial banker and spent seven years as a register
ed  securities principal with a regional investment banking firm. she also owned
  and managed a retail business for several years.  please join me in supporting
 sally in this additional coordination role for  global risk management operatio
ns."""
wordCounts=defaultdict(int)
for word in re.findall(r'[a-z]+', line.lower()):
    wordCounts[word] += 1
for key in sorted(wordCounts)[0:10]:
    print (key, wordCounts[key])
```

```
('a', 7)
('accounting', 1)
('activities', 3)
('additional', 1)
('administration', 1)
('all', 3)
('allocation', 1)
('also', 3)
('america', 2)
('among', 1)
```

Fill in the code block below, then execute the cell as well as the cell below it.

This will write a file named alice_words.py and run it.

the output per line should be a tab separated key-value pair with the following format WORD TAB count

```
a          2333333
abide      2
able       1
about      102
above      3
absence    1
absurd     2
accept     1
acceptance 1
accepted   2
```

In [6]:

```
%%writefile alice_words.py
import re
import sys
from collections import defaultdict

pathToFile = sys.argv[1]
wordCounts = defaultdict(int)



def hw11(pathToFile):
  # takes the path to the file as command line argument
  # prints sorted tab separated list of words and counts
  # ex) print word,'\t',count
  # returns sorted list of tuples of words and counts: wordList
  # ex) wordList = [('a', 690),('abide', 2),...]

  wordList = []

  # START STUDENT CODE HW11
  file=open(pathToFile,"r+")


  template = "{0:17}{1:3}"
  for word in re.findall(r'[a-z]+', file.read().lower()):
    wordCounts[word] += 1
  for key in sorted(wordCounts):
        #print template.format(key, wordCounts[key])
        print key ,'\t' ,wordCounts[key]

  # END STUDENT CODE HW11

  #print wordList

hw11(pathToFile)
```

```
Overwriting alice_words.py
```

In [7]:

```
!python alice_words.py 'alicesTExtFilename.txt' > alice_words.txt
```

Pretty print top 10 results from alice_words.txt

In [8]:

```
#str.format() is a handy funcion for human friendly pretty printing:
#Examples: https://docs.python.org/2/library/string.html#format-examples
print '{:15}{}'.format('Word', 'Count')
print '='*20

with open("alice_words.txt") as f:
  idx = 0
  for line in f.readlines():
    line = line.strip()
    word, count = line.split('\t')
    # print the top 10 lines
    if idx < 10:
        print '{:17}{:3d}'.format(word, int(count))
        #print line
    idx += 1
```

```
Word           Count
====================
a                690
abide              2
able               1
about            102
above              3
absence            1
absurd             2
accept             1
acceptance         1
accepted           2
```

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

HINT: read the file with python, or use subprocess to access the commandline from within the python function

As before, fill in the code block below, then execute the cell as well as the cell below it. This will write a file named hw111.py and run it.

In [9]:

```
%%writefile hw111.py
import re
import sys
from collections import defaultdict

word = sys.argv[1]
pathToFile = sys.argv[2]
adict = defaultdict(int)
def hw111(word,pathToFile):
  # takes a word and the path to the file as arguments
  # returns the line containing the word and count

  # START STUDENT CODE HW111
      with open(pathToFile) as f:

          for line in f.readlines():
            line = line.strip()
            key, count = line.split('\t')
            adict[key.rstrip()] = count
      return adict[word]

  # END STUDENT CODE HW111

print hw111(word,pathToFile)
```

```
Overwriting hw111.py
```

In [10]:

```
!python hw111.py 'alice' 'alice_words.txt'
```

```
403
```

3. HW1.2 Command Line Map Reduce Framework

Back to Table of Contents

For this HW question:

- 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 all the code blocks.
- No need to modify anything in this question.

In [11]:

```
%%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
##       word = a word in quotes, e.g., "alice"
##       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.p
y,
##               as this will determine how the python scripts take input.

###----------------------------------------------------------------------------
-----------
#
# For HW1.3:
# modify this script to include shuffle/sort/merge phase,
# which will collate wordCount records with the same key (i.e., same word)
# run: man sort to learn more about the linux sort command
#
###----------------------------------------------------------------------------
-----------


usage()
{
    echo ERROR: No arguments supplied
    echo
    echo To run use
    echo "pWordCount.sh m word inputFile"
    echo Input:
    echo "number of processes/maps, EG, 4"
    echo "word = a word in quotes, e.g., 'alice'"
    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

word=$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 $data.chunk.*; do
    ## feed word list to the python mapper here and redirect STDOUT to a tempora
ry file on disk
    ####
    ####
    ./mapper.py  "$word" < $datachunk > $datachunk.counts &
    ####
    ####
done
## wait for the mappers to finish their work
wait



## '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 disk
####
####
cat $countfiles |sort -k1,1| ./reducer.py  > $data.output
####
####

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

## display the content of the output file:
cat $data.output
```

```
Overwriting pWordCount.sh
```

In [12]:

```
!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
##       word = a word in quotes, e.g., "alice"
##       inputFile = a text input file
##
## Instructions: Read this script and its comments closely.
```

In [13]:

```
# Change the execution priviledges to make the shell script executable by all
!chmod a+x pWordCount.sh
```

Test the framework without parameters:

In [14]:

```
! ./pWordCount.sh
```

```
ERROR: No arguments supplied

To run use
pWordCount.sh m word inputFile
Input:
number of processes/maps, EG, 4
word = a word in quotes, e.g., 'alice'
inputFile = a text input file
```

Run the following two cells to generate mapper and reducer files, then run the shell script again with arguments.¶

In [15]:

```
%%writefile mapper.py
#!/usr/bin/python
import sys
import re
count = 0

findword = sys.argv[1]
for line in sys.stdin:
    # count all occurances of the word in each line:
    count = count + line.lower().count(findword)
print count
```

```
Overwriting mapper.py
```

In [16]:

```
%%writefile reducer.py
#!/usr/bin/python
## Description: reducer code for HW1.2
import sys
import re
sum = 0

for line in sys.stdin:
  sum += int(line)

print sum
```

```
Overwriting reducer.py
```

Make the files executable:

In [17]:

```
!chmod a+x mapper.py
!chmod a+x reducer.py
```

Test the framework with parameters:

In [18]:

```
!./pWordCount.sh 4 'alice' 'alicesTExtFilename.txt'
```

```
403
```

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

Back to Table of Contents

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).

Write the mapper.py/reducer.py combination to perform WordCount using the command line mapreduce 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 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

```
a           690
abide       2
able        1
about       102
above       3
absence     1
absurd      2
accept      1
acceptance  1
accepted    2
```

Here, mapper.py will read in a portion (i.e., a single record corresponding to a row) of the Alice in Wonderland data, count the number of occurences of the word in question and print/emit a count to the output stream. The reducer responsible for reading in counts of the word from the input stream, 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 [19]:
```

```
%%writefile mapper.py
#!/usr/bin/python
import sys
import re
from collections import defaultdict
 #for each document create dictionary of words
word_cnts = defaultdict(int)
# START STUDENT CODE HW13MAPPER
for line in sys.stdin:



    for word in re.findall(r'[a-z]+',line.lower()):

        word_cnts[word] += 1
    # emit key-value pairs only for distinct words per document
for w in word_cnts.keys():

        print '%s\t%s' % (w,word_cnts[w])
# END STUDENT CODE HW13MAPPER
```

```
Overwriting mapper.py
```

```
In [20]:
```

```
%%writefile reducer.py
#!/usr/bin/python

# START STUDENT CODE HW13REDUCER
import sys
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    # remove leading and trailing whitespaces
    line = line.strip()
    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
# END STUDENT CODE HW13REDUCER
```

```
Overwriting reducer.py
```

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

```
In [21]:
```

```
!chmod +x mapper.py;
!chmod +x reducer.py
```

Run the command below with 4 mappers. You should get the same result as in HW1.1.1.

In [22]:

```
!./pWordCount.sh 4 '*' 'alicesTExtFilename.txt'
```

```
a       690
abide   2
able    1
about   102
above   3
absence 1
absurd  2
accept  1
acceptance      1
accepted        2
accepting       1
access  10
accessed        1
accessible      1
accident        2
accidentally    1
accordance      2
account 1
accounting      1
accounts        1
accusation      1
accustomed      1
ache    1
across  5
act     1
active  2
actual  1
actually        1
ada     1
added   23
adding  1
addition        1
additional      4
additions       1
address 1
addressed       2
addresses       1
addressing      1
adjourn 1
adoption        1
advance 3
advantage       3
adventures      12
advice  2
advisable       2
advise  1
affair  1
affectionately  1
afford  1
afore   1
afraid  12
after   43
afterwards      2
again   83
against 10
age     4
aged    1
agent   1
ago     2
agony   1
agree   11
```

```
agreed	1
agreement	18
ah	5
ahem	1
air	15
airs	1
ak	1
alarm	2
alarmed	1
alas	4
alice	403
alive	3
all	200
allow	4
almost	8
alone	5
along	6
aloud	5
already	3
also	4
alteration	1
altered	1
alternate	1
alternately	1
altogether	5
always	13
am	16
ambition	1
among	12
an	61
ancient	1
and	940
anger	2
angrily	9
angry	5
animal	2
animals	4
ann	4
annoy	1
annoyed	1
another	22
answer	9
answered	4
answers	1
antipathies	1
anxious	3
anxiously	14
any	76
anyone	5
anything	22
anywhere	3
appealed	1
appear	3
appearance	1
appeared	8
appearing	2
appears	1
applause	1
apple	1
apples	2
applicable	3
```

```
apply	1
approach	1
arch	1
archbishop	2
arches	4
archive	13
are	73
argue	1
argued	1
argument	4
arguments	1
arise	1
arithmetic	1
arm	15
arms	6
around	3
arranged	1
array	1
arrived	1
arrow	1
arrum	1
as	274
ascii	2
ashamed	2
ask	11
askance	1
asked	17
asking	5
asleep	8
assembled	2
assistance	1
associated	8
at	227
ate	1
atheling	1
atom	2
attached	1
attempt	1
attempted	1
attempts	1
attended	1
attending	3
attends	1
audibly	1
australia	1
author	1
authority	2
available	2
avoid	1
away	28
awfully	1
axes	1
axis	1
b	3
baby	14
back	39
backs	1
bad	2
bag	1
baked	1
balanced	1
```

```
balls	1
bank	3
banks	1
banquet	1
bark	2
barking	1
barley	1
barrowful	2
based	2
bat	3
bathing	1
bats	4
bawled	1
be	167
beak	1
bear	2
beast	1
beasts	2
beat	4
beating	2
beau	4
beauti	1
beautiful	13
beautifully	2
beautify	1
became	2
because	16
become	5
becoming	1
bed	1
beds	2
bee	1
been	38
before	40
beg	8
began	58
begged	1
begin	13
beginning	15
begins	4
begun	7
behead	1
beheaded	3
beheading	1
behind	13
being	19
believe	9
believed	1
bells	1
belong	1
belongs	2
beloved	1
below	6
belt	1
bend	2
bent	1
besides	4
best	12
better	14
between	6
bill	17
```

```
binary  1
bird    2
birds   10
birthday        1
bit     16
bite    2
bitter  1
blacking        1
blades  1
blame   1
blasts  2
bleeds  1
blew    2
blow    2
blown   1
blows   1
body    2
boldly  1
bone    1
bones   1
book    11
books   2
boon    1
boots   4
bore    1
both    16
bother  1
bottle  10
bottom  4
bough   1
bound   3
bowed   4
bowing  1
box     10
boxed   1
boy     3
brain   1
branch  1
branches        2
brandy  1
brass   1
brave   1
breach  2
bread   7
break   2
breath  4
breathe 3
breeze  1
bright  8
brightened      2
bring   3
bringing        3
bristling       1
broke   2
broken  6
brother 1
brought 3
brown   2
brush   1
brushing        1
burn    2
```

```
burning 1
burnt   1
burst   1
bursting        1
busily  4
business        9
busy    2
but     175
butter  9
buttercup       1
buttered        1
butterfly       1
buttons 1
by      78
bye     2
c       6
cackled 1
cake    3
cakes   3
calculate       1
calculated      1
call    9
called  15
calling 1
calmly  1
came    40
camomile        1
can     73
canary  1
candle  3
cannot  5
canterbury      1
canvas  1
capering        1
capital 4
card    1
cardboard       1
cards   3
care    4
carefully       3
cares   2
carried 4
carrier 1
carroll 4
carry   2
carrying        2
cart    1
cartwheels      1
case    5
cat     37
catch   4
catching        2
caterpillar     28
cats    13
cattle  1
caucus  3
caught  3
cauldron        2
cause   5
caused  2
cautiously      3
```

```
cease	1
ceiling	1
centre	1
certain	5
certainly	14
chain	1
chains	1
chair	1
chance	4
chanced	1
change	15
changed	8
changes	2
changing	2
chapter	12
character	2
charge	6
charges	2
charitable	1
charities	1
chatte	1
cheap	1
cheated	1
check	2
checked	3
checks	1
cheeks	1
cheered	3
cheerfully	1
cherry	1
cheshire	7
chief	2
child	11
childhood	1
children	10
chimney	6
chimneys	1
chin	7
choice	2
choke	1
choked	3
choking	1
choose	1
choosing	1
chop	1
chorus	6
chose	2
christmas	1
chrysalis	1
chuckled	1
circle	1
circumstances	1
city	1
civil	3
claim	1
clamour	1
clapping	1
clasped	1
classics	1
claws	2
clean	1
```

```
clear	2
cleared	1
clearer	1
clearly	2
clever	2
climb	1
clinging	1
clock	5
close	13
closed	2
closely	1
closer	1
clubs	1
coast	1
coaxing	2
codes	1
coils	1
cold	1
collar	1
collected	2
collection	4
come	47
comes	2
comfits	2
comfort	1
comfortable	1
comfortably	1
coming	9
commercial	1
committed	1
common	1
commotion	1
company	1
compilation	1
complained	1
complaining	1
completely	1
compliance	5
comply	6
complying	3
compressed	1
computer	2
computers	2
concept	2
concerning	2
concert	2
concluded	2
conclusion	2
condemn	1
conduct	1
confirmation	1
confirmed	1
confused	4
confusing	3
confusion	5
conger	1
conqueror	2
conquest	1
consented	1
consequential	1
consider	4
```

```
considerable	2
considered	3
considering	3
constant	3
consultation	1
contact	4
contain	2
containing	1
contempt	1
contemptuous	1
contemptuously	2
content	1
continued	9
contract	1
contradicted	1
contributions	2
conversation	9
conversations	2
convert	1
cook	13
cool	2
copied	2
copies	7
copy	12
copying	4
copyright	14
corner	4
corners	1
corporation	1
corrupt	1
cost	4
costs	2
could	78
couldn	9
counting	1
countries	1
country	3
couple	1
couples	1
courage	3
course	27
court	18
courtiers	2
coward	1
crab	3
crash	3
crashed	1
crawled	1
crawling	1
crazy	1
created	2
creating	4
creation	1
creature	4
creatures	10
credit	1
creep	1
crept	1
cried	20
cries	1
crimson	2
```

```
critical	1
crocodile	1
croquet	9
croqueted	1
croqueting	1
cross	3
crossed	3
crossly	1
crouched	1
crowd	4
crowded	5
crown	3
crumbs	4
crust	1
cry	3
crying	2
cucumber	2
cunning	1
cup	2
cupboards	2
cur	1
curiosity	5
curious	19
curiouser	2
curled	2
curls	1
curly	1
currants	1
current	1
curtain	1
curtsey	1
curtseying	1
curving	1
cushion	2
custard	1
custody	2
cut	5
cutting	1
d	30
dainties	1
daisies	1
daisy	1
damage	2
damaged	1
damages	4
dance	13
dancing	2
dare	5
daresay	1
dark	3
darkness	1
data	1
date	4
dates	1
daughter	1
day	29
days	8
dead	4
deal	12
dear	29
dears	3
```

```
death	1
decided	3
decidedly	4
declare	2
declared	1
deductible	1
deep	7
deepest	1
deeply	4
defect	3
defective	3
defects	1
delay	1
deletions	1
delight	3
delighted	2
delightful	2
demand	1
denial	1
denied	2
denies	1
deny	2
denying	1
depends	2
derision	1
derivative	3
derive	1
derived	1
described	1
deserved	1
desk	1
desks	1
despair	1
desperate	1
desperately	1
despite	1
destroy	2
detach	1
determine	1
diamonds	1
did	63
didn	14
die	1
died	1
different	10
difficult	2
difficulties	1
difficulty	4
dig	1
digging	4
diligently	1
dinah	14
dinn	2
dinner	2
dipped	2
direct	1
directed	2
direction	5
directions	3
directly	3
director	1
```

```
disagree        1
disappeared     2
disappointment  1
disclaim        1
disclaimer      3
disclaimers     1
discontinue     1
discover        1
discovered      1
disgust 1
dish    4
dishes  2
disk    1
dismay  1
disobey 1
display 1
displayed       1
displaying      4
dispute 2
distance        8
distant 2
distraction     1
distribute      6
distributed     4
distributing    7
distribution    6
distributor     1
dive    1
do      98
dodged  1
dodo    13
does    11
doesn   16
dog     3
dogs    3
doing   5
domain  8
don     61
donate  4
donation        1
donations       15
done    15
donors  1
door    30
doors   2
doorway 1
dormouse        40
doth    3
double  1
doubled 1
doubling        1
doubt   4
doubtful        2
doubtfully      2
down    102
downloading     1
downward        1
downwards       1
doze    1
dozing  1
dr      2
```

```
draggled        1
draw    7
drawing 1
drawling        3
dreadful        2
dreadfully      6
dream   7
dreamed 1
dreaming        1
dreamy  1
dressed 1
drew    5
dried   1
driest  1
drink   7
drinking        1
dripping        1
drive   2
drop    1
dropped 5
dropping        1
drowned 1
drunk   2
dry     8
duchess 42
duck    4
dull    3
dunce   1
e       29
each    9
eager   3
eagerly 8
eaglet  3
ear     6
earls   2
earnestly       2
ears    5
earth   4
easily  4
easy    3
eat     18
eaten   1
eating  1
eats    1
ebook   9
ebooks  7
edgar   1
edge    3
edition 2
editions        6
educational     1
educations      1
edwin   2
eel     2
eels    1
effect  3
effort  2
efforts 3
egg     1
eggs    5
eh      1
```

```
ein	1
either	11
elbow	3
elbows	1
elect	1
electronic	27
electronically	2
elegant	1
eleventh	1
else	12
elsie	1
em	3
email	3
emphasis	1
employee	1
employees	2
empty	1
encoding	1
encourage	1
encouraged	1
encouraging	2
end	20
ending	2
energetic	1
engaged	1
engine	1
england	1
english	7
engraved	1
enjoy	1
enormous	1
enough	18
ensuring	1
entangled	2
entirely	2
entity	3
entrance	1
equipment	3
errors	1
escape	4
especially	1
esq	1
est	1
even	21
evening	5
ever	21
every	12
everybody	8
everything	14
evidence	7
evidently	1
exact	1
exactly	8
examine	2
examining	1
excellent	2
except	7
exclaimed	6
exclamation	1
exclusion	1
execute	1
```

```
executed	6
executes	1
execution	3
executioner	6
executions	2
executive	1
exempt	2
existence	1
exists	1
expected	1
expecting	3
expend	1
expense	1
expenses	2
experiment	2
explain	10
explained	1
explanation	4
explanations	1
exporting	1
express	1
expressing	1
expression	1
extent	1
extra	1
extraordinary	2
extras	1
extremely	2
eye	7
eyed	1
eyelids	1
eyes	29
f	11
face	15
faces	5
facility	1
fact	8
fading	1
failure	1
faint	1
fainting	1
faintly	1
fair	1
fairbanks	1
fairly	1
fairy	1
fall	7
fallen	4
falling	2
familiarly	1
family	1
fan	10
fancied	2
fancy	7
fancying	1
fanned	1
fanning	1
far	13
farm	1
farmer	1
farther	1
```

```
fashion 2
fast    4
faster  3
fat     1
father  6
favoured        1
favourite       1
fear    4
feared  1
feather 1
feathers        1
federal 2
fee     8
feeble  2
feebly  1
feel    8
feeling 7
feelings        2
fees    4
feet    19
fell    6
fellow  4
fellows 1
felt    23
fender  1
ferrets 2
fetch   7
few     10
fidgeted        1
field   1
fifteen 1
fifteenth       1
fifth   1
fig     1
fight   2
fighting        1
figure  3
figures 1
file    2
files   2
filled  3
fills   1
financial       1
find    21
finding 3
finds   1
fine    2
finger  5
finish  5
finished        12
finishing       1
fire    4
fireplace       1
first   51
fish    8
fishes  1
fit     3
fitness 1
fits    1
fitted  1
five    8
```

```
fix     2
fixed   1
flame   1
flamingo        5
flamingoes      2
flapper 1
flappers        1
flashed 1
flat    2
flavour 1
flew    1
flinging        1
flock   1
floor   3
flower  2
flowers 2
flown   1
flung   1
flurry  1
flustered       1
fluttered       1
fly     3
flying  1
folded  3
folding 1
follow  3
followed        8
following       3
follows 3
fond    4
foolish 1
foot    10
footman 14
footmen 1
footsteps       2
for     179
forehead        2
forepaws        1
forget  2
forgetting      3
forgot  2
forgotten       6
fork    1
form    5
format  4
formats 2
forth   8
fortunately     1
forty   1
forwards        1
found   35
foundation      25
fountains       2
four    8
fourteenth      1
fourth  1
frame   1
frames  1
france  1
free    8
freely  4
```

```
french  4
friend  3
friends 2
fright  2
frighten        1
frightened      7
frog    3
from    51
front   2
frontispiece    1
frowning        4
frying  1
ful     1
fulcrum 1
full    19
fumbled 1
fun     3
fundraising     1
funny   3
fur     3
furious 1
furiously       1
furrow  1
furrows 1
further 4
fury    3
future  3
gained  1
gallons 1
game    13
games   1
garden  16
gardeners       8
gather  1
gave    15
gay     1
gazing  1
gbnewby 1
general 6
generally       7
generations     2
gently  3
geography       1
get     46
getting 22
giddy   2
girl    4
girls   3
give    16
given   2
giving  2
glad    11
glanced 1
glaring 1
glass   10
globe   1
gloomily        1
gloves  11
go      50
goals   1
goes    7
```

```
going	27
golden	7
goldfish	2
gone	13
good	27
goose	2
got	45
govern	1
graceful	1
grammar	1
grand	3
grant	1
granted	1
grass	4
gratefully	1
grave	3
gravely	3
gravy	1
grazed	1
great	39
green	4
gregory	1
grew	1
grey	1
grief	1
grin	6
grinned	3
grinning	1
grins	1
gross	1
ground	8
group	1
grow	13
growing	11
growl	3
growled	1
growling	1
growls	1
grown	7
grumbled	1
grunt	1
grunted	4
gryphon	55
guard	1
guess	3
guessed	3
guests	3
guilt	1
guinea	6
gutenberg	93
had	178
hadn	8
hair	7
half	23
hall	9
hand	21
handed	3
hands	12
handsome	1
handwriting	1
hanging	3
```

```
happen	8
happened	7
happening	1
happens	5
happy	1
hard	8
hardly	12
hare	31
harm	1
harmless	1
hart	2
has	9
hasn	2
haste	1
hastily	16
hat	1
hatching	1
hate	2
hated	1
hatter	56
hatters	1
have	85
haven	8
having	10
he	128
head	50
heads	10
heap	1
hear	15
heard	30
hearing	4
heart	2
hearth	1
hearthrug	1
hearts	8
heavy	2
hedge	2
hedgehog	7
hedgehogs	3
hedges	1
heels	1
height	5
held	4
help	12
helped	1
helpless	1
her	248
herald	1
here	51
hers	4
herself	83
hid	1
hide	1
high	16
highest	1
him	43
himself	6
hint	2
hippopotamus	1
his	96
hiss	1
```

```
histories       1
history 7
hit     2
hjckrrh 1
hm      1
hoarse  3
hoarsely        1
hold    11
holder  4
holding 3
hole    5
holiday 1
hollow  1
home    5
honest  1
honour  4
hookah  5
hope    4
hoped   1
hopeful 1
hopeless        1
hoping  3
horse   1
hot     7
hour    2
hours   4
house   18
housemaid       1
houses  1
how     72
however 21
howled  1
howling 3
http    8
humble  1
humbly  2
hundred 1
hundreds        1
hung    1
hungry  3
hunting 3
hurried 11
hurriedly       2
hurry   11
hurrying        1
hurt    3
hush    3
hypertext       1
i       545
idea    15
identification  1
identify        1
idiot   1
idiotic 1
if      116
ignorant        1
ii      1
iii     1
ill     2
imagine 2
imitated        1
```

```
immediate       3
immediately     3
immense 1
impatient       1
impatiently     5
impertinent     1
implied 2
important       8
imposed 1
impossible      3
improve 1
in      431
inaccurate      1
incessantly     1
inches  6
incidental      1
inclined        1
include 1
included        3
includes        1
including       8
incomplete      1
increasing      1
indeed  16
indemnify       1
indemnity       1
indicate        1
indicating      1
indignant       1
indignantly     4
indirect        1
indirectly      1
individual      4
information     8
infringement    1
injure  1
ink     1
inkstand        1
inquired        1
inquisitively   1
inside  2
insolence       1
instance        3
instantly       5
instead 3
insult  1
intellectual    2
interest        1
interesting     5
internal        1
international   1
interpreted     1
interrupt       1
interrupted     9
interrupting    2
into    67
introduce       2
introduced      1
invalidity      1
invent  1
invented        1
```

```
invitation	2
invited	2
involved	1
inwards	1
irons	1
irritated	1
irs	1
is	135
isn	7
it	610
its	63
itself	14
iv	1
ix	1
jack	1
jar	2
jaw	1
jaws	2
jelly	1
jogged	1
join	9
joined	3
journey	1
joys	1
judge	4
judging	1
jug	1
jumped	6
jumping	4
june	1
juror	1
jurors	4
jury	22
jurymen	4
just	52
justice	1
keep	13
keeping	4
kept	13
kettle	1
key	9
kick	3
kid	5
kill	1
killing	1
kills	1
kind	8
kindly	2
king	63
kings	1
kiss	1
kissed	1
kitchen	4
knave	9
knee	5
kneel	1
knelt	1
knew	15
knife	3
knock	1
knocked	1
```

```
knocking	3
knot	2
know	88
knowing	2
knowledge	3
known	1
knows	2
knuckles	1
label	2
labelled	1
lacie	1
lad	1
ladder	1
lady	3
laid	2
lake	1
lamps	1
land	1
language	1
languid	1
lap	2
large	33
larger	7
largest	1
lark	1
last	34
lasted	2
lastly	1
late	6
lately	1
later	3
latin	1
latitude	2
laugh	1
laughed	2
laughing	2
laughter	1
law	4
laws	8
lay	4
lazily	1
lazy	1
leaders	1
leading	1
leaning	2
leant	1
leap	1
learn	8
learned	1
learning	2
learnt	2
least	9
leave	9
leaves	6
leaving	1
led	4
ledge	1
left	14
lefthand	2
legal	2
legally	1
```

```
legged  2
legs    3
length  1
less    4
lessen  1
lesson  3
lessons 10
lest    1
let     22
letter  4
letters 1
lewis   4
liability       3
liable  1
library 1
license 16
licensed        1
licking 1
lie     2
lieu    2
life    13
lifted  1
like    85
liked   6
likely  5
likes   1
limbs   1
limitation      3
limited 5
line    2
lines   1
linked  2
links   3
lips    1
list    3
listen  7
listened        1
listeners       1
listening       3
lit     1
literary        13
little  128
live    8
lived   3
livery  3
lives   4
living  2
lizard  6
ll      57
lobster 7
lobsters        7
located 4
locations       2
lock    1
locked  1
locks   2
lodging 1
london  1
lonely  2
long    33
longed  2
```

```
longer  3
longitude       2
look    29
looked  45
looking 32
loose   2
lory    7
lose    1
losing  1
lost    3
lot     1
loud    6
louder  1
loudly  3
love    3
loveliest       1
lovely  2
loving  1
low     15
lower   1
lowing  1
luckily 2
lullaby 1
lying   8
m       63
ma      3
mabel   4
machine 1
machines        1
mad     15
made    30
magic   1
magpie  1
mail    1
main    1
maintaining     1
majesty 12
make    30
makes   12
making  8
mallets 1
man     5
manage  7
managed 4
managing        1
manner  2
manners 1
many    14
maps    1
march   35
marched 1
mark    3
marked  8
marmalade       1
mary    4
master  4
matter  9
matters 2
maximum 1
may     28
maybe   2
```

```
mayn    1
me      68
meal    1
mean    10
meaning 8
means   8
meant   5
meanwhile       1
measure 1
meat    1
medium  5
meekly  2
meet    3
meeting 1
melan   1
melancholy      6
memorandum      1
memory  1
men     1
mentioned       3
merchantibility 1
mercia  2
merely  2
merrily 1
messages        2
met     4
method  1
methods 1
mice    4
michael 2
middle  8
might   28
mile    2
miles   3
milk    2
millennium      1
mind    11
minded  1
minding 1
mine    10
mineral 1
minute  21
minutes 11
mischief        1
miserable       2
miss    4
missed  2
mission 4
mississippi     1
mistake 3
mixed   2
mock    56
moderate        1
modern  1
modification    1
modified        1
moment  31
money   3
month   2
moon    1
moral   8
```

```
morals  1
morcar  2
more    50
morning 5
morsel  1
most    11
mostly  2
mournful        1
mournfully      1
mouse   44
mouth   10
mouths  4
move    3
moved   5
moving  3
much    52
muchness        3
muddle  1
multiplication  1
murder  1
murdering       1
muscular        1
mushroom        8
music   3
must    54
mustard 3
muttered        2
muttering       3
my      58
myself  7
mystery 2
name    11
named   1
names   2
narrow  2
nasty   1
natural 4
natured 1
naturedly       1
nay     1
near    15
nearer  5
nearly  13
neat    1
neatly  2
necessarily     1
neck    7
need    1
needn   3
needs   1
negligence      1
neighbour       1
neighbouring    1
neither 2
nervous 5
nest    1
network 1
never   48
nevertheless    1
new     8
newby   1
```

```
newsletter      1
newspapers      1
next    30
nibbled 2
nibbling        3
nice    6
nicely  2
night   5
nile    1
nine    5
no      100
nobody  8
nodded  1
noise   3
noises  1
non     1
none    4
nonproprietary  1
nonsense        7
nor     3
normans 1
north   1
northumbria     2
nose    8
not     166
note    2
nothing 34
notice  8
noticed 8
noticing        1
notifies        1
notion  3
now     60
nowhere 2
number  8
numerous        1
nurse   3
nursing 3
o       6
obliged 3
oblong  1
obsolete        1
obstacle        1
obtain  3
obtaining       2
occasional      1
occasionally    1
occur   1
occurred        2
october 1
odd     1
of      631
off     73
offend  1
offended        10
offer   2
offers  1
office  2
officer 1
officers        4
official        3
```

```
often	6
oh	45
ointment	1
old	21
older	2
oldest	1
on	204
once	34
one	106
ones	1
oneself	1
onions	1
online	4
only	52
oop	7
ootiful	4
open	7
opened	10
opening	3
opinion	1
opportunities	1
opportunity	9
opposite	1
or	155
orange	1
order	3
ordered	4
ordering	2
org	13
organized	1
original	1
originator	1
ornamented	2
other	54
others	8
otherwise	4
ou	1
ought	14
our	12
ours	1
ourselves	1
out	118
outdated	1
outside	7
over	40
overcome	1
overhead	1
owed	1
owl	3
own	10
owner	5
owns	2
oyster	1
pace	1
pack	5
page	2
pages	1
paid	6
paint	1
painting	2
pair	5
```

```
pairs	1
pale	4
pan	1
panted	1
panther	3
panting	2
paper	5
paperwork	1
paragraph	11
paragraphs	3
parchment	2
pardon	6
pardoned	1
paris	2
part	6
particular	6
particularly	1
partner	1
partners	1
parts	1
party	11
pass	1
passage	4
passed	5
passing	1
passion	3
passionate	1
past	3
pat	3
patience	1
patiently	2
patriotic	1
patted	1
pattering	3
pattern	1
pause	2
paused	1
paw	3
paws	4
pay	1
paying	2
payments	3
pebbles	2
peeped	3
peeping	1
peering	1
pegs	1
pence	1
pencil	2
pencils	1
pennyworth	2
people	16
pepper	8
perfectly	4
perform	1
performances	1
performed	1
performing	3
perhaps	17
periodic	1
permanent	1
```

```
permission	7
permitted	3
persisted	2
person	8
personal	2
persons	1
pet	1
pg	1
pglaf	8
phrase	4
physical	2
picked	3
picking	2
picture	1
pictured	1
pictures	4
pie	3
piece	6
pieces	3
pig	11
pigeon	12
pigs	6
pinch	2
pinched	2
pine	1
pink	1
piteous	1
pitied	1
pity	3
place	9
placed	1
places	2
plain	2
plainly	1
plan	4
planning	1
plate	3
plates	2
play	8
played	1
players	4
playing	2
pleaded	3
pleasant	1
pleasanter	1
please	22
pleased	7
pleases	1
pleasing	1
pleasure	2
plenty	2
pocket	7
pointed	1
pointing	4
poison	3
poker	1
poky	1
politely	6
pool	11
poor	27
pop	1
```

```
pope    1
porpoise        4
position        2
positively      1
possessed       1
possession      1
possibility     1
possible        1
possibly        3
posted  5
posting 1
pot     1
pounds  1
pour    1
poured  1
powdered        1
practically     1
practice        1
pray    3
precious        1
prepare 2
present 3
presented       1
presently       2
presents        2
preserve        1
pressed 3
pressing        1
pretend 1
pretending      1
pretexts        1
prettier        1
pretty  1
prevent 2
previous        1
principal       1
print   1
printed 3
prison  1
prisoner        2
prize   1
prizes  5
problem 1
proceed 2
processing      1
procession      5
processions     1
produce 1
produced        2
producing       1
production      1
professor       1
profit  1
profits 1
prohibition     1
project 87
prominently     2
promise 1
promised        1
promising       1
promoting       2
```

```
promotion       1
pronounced      1
proofread       1
proper  3
property        2
proposal        1
proprietary     1
prosecute       1
protect 2
protection      1
proud   2
prove   1
proved  2
proves  2
provide 7
provided        4
providing       4
provision       1
provisions      1
provoking       1
public  9
puffed  1
pulled  1
pulling 1
pun     1
punching        1
punished        1
punitive        1
puppy   7
purple  1
purpose 3
purring 2
push    1
puss    1
put     31
putting 3
puzzle  1
puzzled 9
puzzling        4
quadrille       4
quarrel 1
quarrelled      1
quarrelling     2
queen   75
queens  1
queer   12
queerest        1
question        17
questions       4
quick   2
quicker 1
quickly 2
quiet   2
quietly 5
quite   55
quiver  1
rabbit  51
rabbits 1
race    6
railway 2
raised  2
```

```
raising 1
ran     16
rapidly 2
rapped  1
rat     1
rate    9
rather  25
rats    1
rattle  1
rattling        2
raven   1
ravens  1
raving  2
raw     1
re      40
reach   4
reaching        2
read    14
readable        2
readily 1
reading 4
ready   8
real    3
reality 1
really  13
rearing 1
reason  9
reasonable      2
reasons 1
receipt 2
receive 3
received        6
receiving       1
recognised      1
recovered       2
red     3
redistribute    1
redistributing  2
redistribution  2
reduced 1
reeds   1
reeling 1
references      2
refreshments    1
refund  10
refused 1
registered      2
regular 2
regulating      1
release 1
relief  2
relieved        1
remain  2
remained        3
remaining       2
remark  10
remarkable      2
remarked        10
remarking       3
remarks 3
remedies        2
```

```
remember        14
remembered      5
remembering     1
reminding       1
remove  1
removed 4
renamed 1
repeat  7
repeated        10
repeating       3
replace 1
replacement     5
replied 29
reply   5
reported        1
reports 1
representations 1
request 1
require 1
required        1
requirements    4
research        2
resource        1
respect 1
respectable     1
respectful      1
rest    10
resting 2
restrictions    2
result  1
retire  1
return  3
returned        2
returning       1
returns 1
revenue 1
rich    1
riddle  1
riddles 2
ridge   1
ridges  1
ridiculous      1
right   36
righthand       1
rightly 1
ring    2
ringlets        2
riper   1
rippling        1
rise    1
rises   1
rising  1
roared  1
roast   1
rock    1
rocket  1
rome    2
roof    6
room    13
roots   2
rope    1
```

```
rose	4
roses	3
rosetree	1
roughly 1
round	41
row	2
royal	2
royalties	2
royalty 3
rubbed	1
rubbing 2
rude	2
rudeness	1
rule	5
rules	5
rumbling	1
run	4
running 8
rush	2
rushed	1
rustled 1
rustling	1
s	219
sad	3
sadly	5
safe	2
sage	1
said	462
salmon	1
salt	3
same	25
sand	1
sands	1
sang	2
sat	17
saucepan	1
saucepans	1
saucer	1
savage	4
save	1
saves	1
saw	14
say	51
saying	15
says	4
scale	1
scaly	1
scattered	1
school	6
schoolroom	1
scolded 1
scrambling	1
scratching	1
scream	2
screamed	4
screaming	1
scroll	2
sea	14
seals	1
seaography	1
search	2
```

```
seaside 1
seated  1
second  6
secondly        2
secret  1
section 7
sections        1
secure  1
see     70
seeing  1
seem    8
seemed  27
seems   5
seen    15
seldom  1
sell    2
send    2
sending 3
sends   1
sensation       2
sense   3
sent    3
sentence        8
sentenced       1
series  1
seriously       1
serpent 9
serpents        3
service 1
set     23
setting 1
settle  1
settled 3
settling        1
seven   6
several 5
severely        4
severity        1
sh      2
shade   1
shake   1
shakespeare     1
shaking 3
shall   27
shan    6
shape   1
shaped  3
share   2
shared  2
sharing 2
shark   1
sharks  1
sharp   6
sharply 4
she     553
shedding        1
sheep   1
shelves 2
shepherd        1
shifting        1
shilling        1
```

```
shillings	1
shingle	1
shining	1
shiny	1
shiver	1
shock	1
shoes	7
shook	9
shore	4
short	4
shorter	2
should	29
shoulder	4
shoulders	4
shouldn	5
shouted	9
shouting	2
show	3
shower	2
showing	2
shriek	5
shrieked	1
shrieks	1
shrill	5
shrimp	1
shrink	1
shrinking	4
shut	5
shutting	2
shy	1
shyly	1
side	17
sides	4
sigh	4
sighed	5
sighing	3
sight	10
sign	1
signed	2
signifies	1
signify	1
silence	14
silent	7
simple	5
simpleton	1
simply	3
since	4
sing	6
singers	2
singing	2
sink	1
sir	7
sister	9
sisters	2
sit	8
site	4
sits	1
sitting	10
six	2
sixpence	1
sixteenth	1
```

```
size	13
sizes	1
skimming	1
skirt	1
skurried	1
sky	5
slate	4
slates	8
sleep	6
sleepy	5
slightest	1
slipped	3
slippery	1
slowly	8
sluggard	1
small	12
smaller	3
smallest	2
smile	2
smiled	2
smiling	2
smoke	1
smoking	2
snail	3
snappishly	1
snatch	2
sneeze	2
sneezed	1
sneezes	2
sneezing	6
snorting	1
snout	1
so	152
sob	1
sobbed	1
sobbing	3
sobs	4
soft	1
softly	1
soldier	1
soldiers	10
solemn	3
solemnly	4
soles	1
solicit	2
solicitation	1
solid	1
some	52
somebody	7
somehow	1
someone	1
somersault	2
something	18
sometimes	5
somewhere	3
son	1
song	7
soo	7
soon	25
sooner	2
soothing	1
```

```
sorrow	2
sorrowful	2
sorrows	1
sorry	1
sort	20
sorts	3
sound	4
sounded	5
sounds	4
soup	18
sour	1
spades	1
speak	15
speaker	1
speaking	5
special	1
specific	1
specified	2
spectacles	3
speech	3
speed	1
spell	1
spirited	1
spite	1
splash	1
splashed	1
splashing	2
splendidly	1
spoke	17
spoken	1
spoon	2
spot	1
sprawling	1
spread	4
spreading	1
squeaked	1
squeaking	2
squeeze	1
squeezed	1
staff	1
stairs	3
stalk	1
stamping	2
stand	6
standing	1
star	1
staring	3
start	3
started	2
startled	2
state	7
statements	1
states	14
station	1
status	4
stay	5
stays	1
steady	1
steam	1
sternly	1
stick	4
```

```
sticks  1
stiff   1
stigand 1
still   13
stingy  1
stirring        2
stockings       1
stole   2
stolen  1
stood   7
stool   1
stoop   2
stop    6
stopped 3
stopping        1
stored  1
story   8
straight        2
straightened    1
straightening   1
strange 5
strength        1
stretched       2
stretching      2
strict  1
strings 1
struck  2
stuff   4
stupid  6
stupidest       1
stupidly        1
subdued 1
subject 7
subjects        1
submitted       1
subscribe       1
succeeded       3
such    47
sudden  5
suddenly        13
suet    1
sugar   2
suit    3
sulkily 2
sulky   3
summer  2
sun     2
supple  1
support 4
suppose 14
suppress        1
suppressed      4
sure    24
surprise        5
surprised       7
survive 1
swallow 1
swallowed       1
swallowing      1
swam    5
swamp   1
```

```
sweet   1
swim    5
swimming        2
synonymous      1
t       218
table   18
tail    9
tails   3
take    22
taken   4
takes   3
taking  5
tale    4
tales   1
talk    14
talking 17
taller  2
tart    1
tarts   7
taste   2
tasted  3
tastes  1
taught  4
tax     6
taxes   1
tea     19
teaching        1
teacup  3
teacups 2
teapot  1
tears   11
teases  1
teeth   1
telescope       3
telescopes      1
tell    32
telling 2
tells   2
temper  5
tempered        2
ten     6
terms   22
terribly        1
terrier 1
terror  1
than    26
thank   4
thanked 1
that    330
thatched        1
the     1818
their   52
theirs  1
them    88
themselves      3
then    94
there   101
therefore       1
these   17
they    155
thick   1
```

```
thimble 4
thin    1
thing   49
things  33
think   53
thinking        11
thirteen        1
thirty  1
this    181
thistle 2
thoroughly      2
those   11
though  11
thought 74
thoughtfully    4
thoughts        2
thousand        2
three   28
threw   2
throat  2
throne  1
through 16
throughout      1
throw   3
throwing        2
thrown  1
thump   2
thunder 1
thunderstorm    1
thus    1
tide    1
tidy    1
tie     1
tied    1
tight   1
till    21
tillie  1
time    71
times   6
timid   3
timidly 9
tinkling        1
tiny    4
tipped  1
tiptoe  2
tired   7
tis     5
title   1
tittered        1
tm      57
to      809
toast   1
today   1
toes    3
toffee  1
together        9
told    6
tomorrow        1
tone    40
tones   2
tongue  4
```

```
too     26
took    24
top     8
tops    1
tortoise        3
toss    1
tossing 3
touch   1
tougher 1
towards 1
toys    1
trademark       11
trampled        1
transcribe      1
transcription   1
traps   1
tray    1
treacle 7
treading        2
treat   1
treated 1
treatment       1
tree    8
trees   7
tremble 1
trembled        2
trembling       6
tremulous       1
trial   10
trials  1
trickling       1
tricks  1
tried   19
trims   1
triumphantly    2
trot    1
trotting        2
trouble 6
true    4
trumpet 3
trusts  1
truth   1
truthful        1
try     12
trying  14
tucked  3
tulip   1
tumbled 1
tumbling        2
tunnel  1
tureen  1
turkey  1
turn    11
turned  16
turning 12
turns   3
turtle  59
turtles 2
tut     2
twelfth 1
twelve  4
```

```
twentieth       1
twenty  3
twice   5
twinkle 8
twinkled        1
twinkling       4
twist   2
two     40
txt     1
types   1
u       3
ugh     2
uglification    2
uglify  1
uglifying       1
ugly    2
unable  1
uncivil 1
uncomfortable   4
uncomfortably   1
uncommon        1
uncommonly      1
uncorked        1
under   22
underneath      1
understand      7
understood      1
undertone       2
undo    1
undoing 1
uneasily        2
uneasy  1
unenforceability        1
unfolded        2
unfortunate     3
unhappy 2
uniform 1
unimportant     5
united  10
unjust  1
unless  6
unlink  1
unlocking       1
unpleasant      2
unrolled        2
unsolicited     1
until   5
untwist 1
unusually       1
unwillingly     1
up      103
updated 2
upon    28
upright 1
upset   3
upsetting       1
upstairs        1
us      15
use     31
used    16
useful  2
```

```
user	3
using	6
usual	5
usually	2
usurpation	1
ut	1
utf	1
v	1
vague	1
vanilla	2
vanished	4
vanishing	1
variations	1
variety	1
various	2
ve	44
vegetable	1
velvet	1
venture	3
ventured	4
verdict	4
verse	4
verses	4
version	1
very	145
vi	1
viewed	1
viewing	1
vii	1
viii	1
vinegar	1
violates	1
violence	1
violent	2
violently	4
virus	1
visit	3
voice	48
voices	3
void	1
volunteer	1
volunteers	6
vote	1
vulgar	1
w	1
wag	1
wags	1
waist	1
waistcoat	2
wait	1
waited	11
waiting	9
wake	2
walk	5
walked	10
walking	5
walks	1
walrus	1
wander	1
wandered	2
wandering	2
```

```
want    9
wanted  4
wants   2
warning 1
warranties      3
warranty        2
was     358
wash    2
washing 3
wasn    11
waste   1
wasting 2
watch   8
watched 2
watching        3
water   5
waters  1
waving  5
way     58
ways    2
we      43
weak    2
wearily 1
web     6
week    3
weeks   1
welcome 1
well    63
went    83
wept    1
were    85
weren   1
west    1
wet     2
what    142
whatever        3
whatsoever      2
when    80
whenever        2
where   18
whereupon       1
wherever        2
whether 11
which   56
while   26
whiles  1
whiskers        3
whisper 3
whispered       5
whispers        1
whistle 1
whistling       1
white   30
whiting 8
who     66
whoever 1
whole   13
whom    2
whose   2
why     40
wide    3
```

```
wider   1
widest  2
wife    1
wig     2
wild    2
wildly  2
will    40
william 8
win     1
wind    2
window  8
wine    2
wings   1
wink    2
winter  1
wise    2
wish    22
with    228
within  6
without 34
witness 10
wits    1
woke    1
woman   2
won     26
wonder  18
wondered        1
wonderful       2
wondering       7
wonderland      8
wood    8
wooden  1
word    11
words   21
wore    1
work    53
works   33
world   7
worm    1
worried 1
worry   1
worse   3
worth   4
would   83
wouldn  13
wow     6
wrapping        1
wretched        2
wriggling       1
write   6
writhing        1
writing 9
written 9
wrong   5
wrote   3
www     6
x       1
xi      1
xii     1
yard    1
yards   1
```

```
yawned  2
yawning 2
ye      1
year    2
years   2
yelled  1
yelp    1
yer     4
yes     13
yesterday       3
yet     25
you     481
young   5
your    71
yours   3
yourself        10
youth   6
zealand 1
zigzag  1
zip     1
```

3. HW1.4 - Count words staring with uppercase and words starting with lowercase (This is an OPTIONAL HW)

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 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 [23]:

```
%%writefile mapper.py
#!/usr/bin/python
import sys
import re
from collections import defaultdict
 #for each document create dictionary of words
word_cnts = defaultdict(int)
# START STUDENT CODE HW13MAPPER
for line in sys.stdin:


    #Upper case letters
    for word in re.findall(r'\b[A-Z][a-z]*\b',line):

        word_cnts['upper'] += 1
    #lower case letters
    for word in re.findall(r'\b[a-z][a-z]*\b',line):

        word_cnts['lower'] += 1
    # emit key-value pairs only for distinct words per document
for w in word_cnts.keys():

        print '%s\t%s' % (w,word_cnts[w])
# END STUDENT CODE HW13MAPPER
```

```
Overwriting mapper.py
```

In [24]:

```
!./pWordCount.sh 4 '*' 'alicesTExtFilename.txt'
```

```
lower   26181
upper   3760
```

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 compare to the choice you made using the true function.

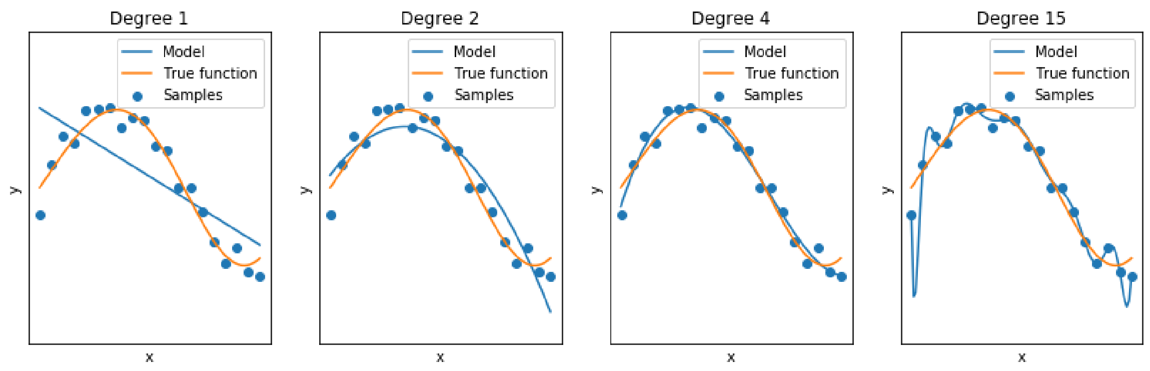
In [25]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline

np.random.seed(100)
# How many samples to generate. Try adjusting this value.
n_samples = 20

X = np.linspace(0, 1, n_samples)


# Set the true function as a piece of a cosine curve.
true_function = lambda x: np.sin(x + x*x + np.pi *x)


# Generate true y values.
y = true_function(X)

# Print the values of y to the nearest hundredth.
print ['%.2f' %i for i in y]

# Add random noise to y.
# The randn function samples random numbers from the standard Normal distributio
n.
# Multiplying adjusts the standard deviation of the distribution.
y += np.random.randn(n_samples) * 0.2

# Print the noise-added values of y for comparison.
print ['%.2f' %i for i in y]
X = X[:, np.newaxis]
print X.shape
```

```
['0.00', '0.22', '0.43', '0.63', '0.79', '0.92', '0.99', '1.00', '0.
94', '0.82', '0.63', '0.40', '0.13', '-0.16', '-0.44', '-0.68', '-0.
87', '-0.98', '-0.99', '-0.91']
['-0.35', '0.29', '0.66', '0.58', '0.99', '1.02', '1.03', '0.78',
 '0.90', '0.87', '0.54', '0.48', '0.01', '0.00', '-0.30', '-0.70',
 '-0.98', '-0.77', '-1.08', '-1.13']
(20, 1)
```

In [26]:

```
# Below, we'll fit polynomials to the noisy data with these degrees.
from sklearn.metrics import mean_squared_error
degrees = [1,2, 4, 15]

# Initialize a new plot.
plt.figure(figsize=(14, 4))

# We'll create a subplot for each value of the degrees list.
for i in range(len(degrees)):
    # The subplots are all on the same row.
    ax = plt.subplot(1, len(degrees), i+1)

    # Turn off tick marks to keep things clean.
    plt.setp(ax, xticks=(), yticks=())

    # Set up the polynomial features preprocessor.
    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)

    # Use the sklearn's Pipeline to string together 2 operations.
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])

    pipeline.fit(X, y)

    # Show samples from the fitted function.
    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")

    # Show the true function.
    plt.plot(X_test, true_function(X_test), label="True function")

    # Show the original noisy samples.
    plt.scatter(X, y, label="Samples")

    # Add a few more labels to the plot.
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((-.05, 1.05))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    plt.title("Degree %d" % degrees[i])

# Render the plots.
plt.show()
```



Understanding overfitting and underfitting

```
Model performs much better with 4 degree but after that it overfits.

While with just degree 1 model has wrong assumptions and it under fits i.
e. high bias errors
```