516_Homework_1

July 4, 2021

1 SIADS 516: Homework 1

Version 1.0.20200221.1 ### Dr. Chris Teplovs, School of Information, University of Michigan This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

1.0.1 Our first mrjob script

Recall the following example from the lectures:

Note the use of the magic command `%%file`. You can use this to write the contents of a cell out to a file, which is what we need to do to use mrjob:

[1]:
```
%%file word_count.py
from mrjob.job import MRJob
import re

class MRWordFrequencyCount(MRJob):

  ### input: self, in_key, in_value
  def mapper(self, _, line):
    yield "chars", len(line)
    yield "words", len(line.split())
    yield "lines", 1

  ### input: self, in_key from mapper, in_value from mapper
  def reducer(self, key, values):
    yield key, sum(values)
if __name__ == "__main__":
    MRWordFrequencyCount.run()
```

```
Writing word_count.py
```

1.0.2 Q1: Explain what each of the yield statements in the above script do. Provide a list of what the first few iterations through the mapper() step would yield if the script was run against the `data/gutenberg/short.t1.txt` file.

Answer

`yield "chars", len(line)` means to caculate the total number of characters in each line (or sentence). The output of this yield statement is to give a tuple with the key of "chars" and the number of characters.

`yield "words", len(line.split())` means to split the sentence by word and caculate the number of words in each line (or sentence). The output of this yield statement is to give a tuple with the key of "words" and the number of words.

`yield "lines", 1` is to yield a tuple with the key of "lines" and "1". "1" means the number of lines in the text file.

Now let's look at the output of running the script against that same file:

[2]:
```
!python word_count.py data/gutenberg/short.t1.txt
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/word_count.jovyan.20210704.020331.569433
Running step 1 of 1...
job output is in /tmp/word_count.jovyan.20210704.020331.569433/output
Streaming final output from
/tmp/word_count.jovyan.20210704.020331.569433/output...
"lines" 200
"words" 1822
"chars" 10653
Removing temp directory /tmp/word_count.jovyan.20210704.020331.569433...
```

1.0.3 Q2. Repeat the above cell using the the works of William Shakespeare text file (data/gutenberg/t8.shakespeare.txt). Provide an interpretation of the output (don't overthink this – just demonstrate that you can find the relevant information in the output).

[3]:
```
# insert your code here
!python word_count.py data/gutenberg/t8.shakespeare.txt
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/word_count.jovyan.20210704.020701.106842
Running step 1 of 1...
job output is in /tmp/word_count.jovyan.20210704.020701.106842/output
Streaming final output from
/tmp/word_count.jovyan.20210704.020701.106842/output...
"chars" 5333743
"lines" 124456
"words" 901325
Removing temp directory /tmp/word_count.jovyan.20210704.020701.106842...
```

Interpretation * `!python word_count.py` means to run the python file named `word_count.py`. * This code creates a temporary directory firstly. * The output below shoes that there're 5333743 characters, 124456 lines, and 901325 words in this text file. * Finally, the temporary directory is deleted once the command get executed.

1.0.4 Now let's look at a slightly more complicated example:

[5]:
```
%%file most_used_word.py
from mrjob.job import MRJob
from mrjob.step import MRStep
import re

WORD_RE = re.compile(r"[\w']+") # any whitespace or apostrophe, used to split␣
 ↪lines below


class MRMostUsedWord(MRJob):
    STOPWORDS = {'i', 'we', 'ourselves', 'hers', 'between', 'yourself', 'but',␣
 ↪'again', 'there', 'about', 'once', 'during', 'out', 'very', 'having',␣
 ↪'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours',␣
 ↪'such', 'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am',␣
 ↪'or', 'who', 'as', 'from', 'him', 'each', 'the', 'themselves', 'until',␣
 ↪'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me',␣
 ↪'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their',␣
 ↪'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no',␣
 ↪'when', 'at', 'any', 'before', 'them', 'same', 'and', 'been', 'have', 'in',␣
 ↪'will', 'on', 'does', 'yourselves', 'then', 'that', 'because', 'what',␣
 ↪'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you',␣
 ↪'herself', 'has', 'just', 'where', 'too', 'only', 'myself', 'which',␣
 ↪'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my',␣
 ↪'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}
    def steps(self):
        return [
            MRStep(mapper=self.mapper_get_words,
                   combiner=self.combiner_count_words,
                   reducer=self.reducer_count_words),
            MRStep(reducer=self.reducer_find_max_word)
        ]

    def mapper_get_words(self, _, line):
        # yield each word in the line
        for word in WORD_RE.findall(line):
            if word.lower() not in self.STOPWORDS:
                yield (word.lower(), 1)

    def combiner_count_words(self, word, counts):
        # optimization: sum the words we've seen so far
        yield (word, sum(counts))

    def reducer_count_words(self, word, counts):
        # send all (num_occurrences, word) pairs to the same reducer.
        # num_occurrences is used so we can easily use Python's max() function.
```

```
        yield None, (sum(counts), word)

    # discard the key; it is just None
    def reducer_find_max_word(self, _, word_count_pairs):
        # each item of word_count_pairs is (count, word),
        # so yielding one results in key=counts, value=word
        yield max(word_count_pairs)



if __name__ == '__main__':
    import time
    start = time.time()
    MRMostUsedWord.run()
    end = time.time()
    print(end - start)
```

```
Overwriting most_used_word.py
```

1.0.5 Q3: Explain what the yield statements in the above script do. Provide a list of what the first few iterations through the steps would yield.

Answer

First of all, we need to identify the definition of stop word.

According to *Data Mining*, "Stop words are words which are filtered out before or after processing of natural language data (text)." Stop words usually the common words which have high frequency appearing in the articles but deliver low effort of the meaning, such as "the", "a", "an", etc.

- `mapper_get_words` statement
  - Take a line from the text file.
  - If the word from the line is not in the dictionary of `STOPWORDS`, execute the next command.
  - Yield a tuple for each word in lowercase from the line and follow by 1.
- `combiner_count_words` statement
  - Count the number of time of words appearing in the line.
  - The output contains the word and the count number.
- `reducer_count_words` statement
  - Sum up the number of words.
  - The output contains the word and the sum-up number.
- `reducer_find_max_word` statement

– The output only contains the word having the highest frequencies of appearance in the text file.

Now run the file against data/gutenberg/short.t1.txt.

[6]:
```
!python most_used_word.py data/gutenberg/short.t1.txt
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/most_used_word.jovyan.20210704.021518.994286
Running step 1 of 2...
Running step 2 of 2...
job output is in /tmp/most_used_word.jovyan.20210704.021518.994286/output
Streaming final output from
/tmp/most_used_word.jovyan.20210704.021518.994286/output...
11      "day"
Removing temp directory /tmp/most_used_word.jovyan.20210704.021518.994286...
0.8557231426239014
```

1.0.6 Q4: Run the above script on the Shakespeare text file. What answer do you get?

[7]:
```
!python most_used_word.py data/gutenberg/t8.shakespeare.txt
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/most_used_word.jovyan.20210704.023030.866045
Running step 1 of 2...
Running step 2 of 2...
job output is in /tmp/most_used_word.jovyan.20210704.023030.866045/output
Streaming final output from
/tmp/most_used_word.jovyan.20210704.023030.866045/output...
5479    "thou"
Removing temp directory /tmp/most_used_word.jovyan.20210704.023030.866045...
7.893137454986572
```

Answer

`5479  "thou"` means the most appearred word in the Shakespeare text file is "thou" and the frequency is 5479 times.

1.0.7 Q5: What is the impact of removing the combiner from the above code in terms of efficiency? What does that suggest?

Answer

By removing the combiner from the above code, the amount of running time is decreased dramatically (7.6s to 5.9s). It might because that the `combiner` is repetitive work of `reducer_count`.

1.0.8 Q6: Write an mrjob script that finds the 10 words that have the most syllables from the t5.churchill.txt file. Interpret your results.

[14]:
```
pip install syllapy
```

```
WARNING: The directory '/home/jovyan/.cache/pip/http' or its parent
directory is not owned by the current user and the cache has been disabled.
Please check the permissions and owner of that directory. If executing pip with
sudo, you may want sudo's -H flag.
WARNING: The directory '/home/jovyan/.cache/pip' or its parent directory is
not owned by the current user and caching wheels has been disabled. check the
permissions and owner of that directory. If executing pip with sudo, you may
want sudo's -H flag.
Collecting syllapy
  Downloading https://files.pythonhosted.org/packages/11/31/e13c6b0ed7a95f46c3af
20df2b995877ea179b88a90ec39122e0621dae08/syllapy-0.7.1-py3-none-any.whl
Collecting ujson<2.0,>=1.35 (from syllapy)
  Downloading https://files.pythonhosted.org/packages/16/c4/79f3409bc71055
9015464e5f49b9879430d8f87498ecdc335899732e5377/ujson-1.35.tar.gz (192kB)
    || 194kB 11.4MB/s eta 0:00:01
Building wheels for collected packages: ujson
  WARNING: Building wheel for ujson failed: [Errno 13] Permission denied:
'/home/jovyan/.cache/pip/wheels/28'
Failed to build ujson
Installing collected packages: ujson, syllapy
  Running setup.py install for ujson ... done
Successfully installed syllapy-0.7.1 ujson-1.35
Note: you may need to restart the kernel to use updated packages.
```

[18]:
```
%%file most_ten_syllables.py
from mrjob.job import MRJob
from mrjob.step import MRStep
import re
import syllapy

WORD_RE = re.compile(r"[\w']+") # any whitespace or apostrophe, used to split␣
 ↪lines below

class MRMostTenSyllables(MRJob):
```

```
    STOPWORDS = {'i', 'we', 'ourselves', 'hers', 'between', 'yourself', 'but',␣
↪'again', 'there', 'about', 'once', 'during', 'out', 'very', 'having',␣
↪'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours',␣
↪'such', 'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am',␣
↪'or', 'who', 'as', 'from', 'him', 'each', 'the', 'themselves', 'until',␣
↪'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me',␣
↪'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their',␣
↪'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no',␣
↪'when', 'at', 'any', 'before', 'them', 'same', 'and', 'been', 'have', 'in',␣
↪'will', 'on', 'does', 'yourselves', 'then', 'that', 'because', 'what',␣
↪'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you',␣
↪'herself', 'has', 'just', 'where', 'too', 'only', 'myself', 'which',␣
↪'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my',␣
↪'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}

    def steps(self):
        return [
            MRStep(mapper=self.mapper_get_words),
            MRStep(reducer=self.reducer_find_max_word)
        ]

    def mapper_get_words(self, _, line):
        # yield each word in the line
        for word in WORD_RE.findall(line):
            if word.lower() not in self.STOPWORDS:
                syllable_count = syllapy.count(word)
                yield None, (syllable_count, word.lower())

    def reducer_find_max_word(self, key, values):
        self.list = []
        for value in values:
            self.list.append(value)
            self.new = []
        for i in range(10):
            self.new.append(max(self.list))
            self.list.remove(max(self.list))
        for i in range(10):
            yield self.new[i]


if __name__ == '__main__':
    import time
    start = time.time()
    MRMostTenSyllables.run()
    end = time.time()
    print(end - start)
```

```
Overwriting most_ten_syllables.py
```

[19]:
```
!python most_ten_syllables.py data/gutenberg/t5.churchill.txt
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/most_ten_syllables.jovyan.20210704.030238.295794
Running step 1 of 2...
Running step 2 of 2...
job output is in /tmp/most_ten_syllables.jovyan.20210704.030238.295794/output
Streaming final output from
/tmp/most_ten_syllables.jovyan.20210704.030238.295794/output...
8       "overcapitalization"
8       "incommunicability"
7       "unenforceability"
7       "overcapitalized"
7       "materialistically"
7       "invulnerability"
7       "interrogatively"
7       "infinitesimally"
7       "indissolubility"
7       "indispensability"
Removing temp directory /tmp/most_ten_syllables.jovyan.20210704.030238.295794...
8.29448413848877
```

Interpretation

The output gives the top 10 high frequencies of words in the text file in tuple for each word and its number of times.

[ ]:
```
```