assignment1

November 28, 2024

1 SIADS 516: Homework 1

- **Dr. Chris Teplovs**, School of Information, University of Michigan
- **Kris Steinhoff**, School of Information, University of Michigan

[1]:
```
# The AutograderHelper class provides methods used by the autograder.
from autograder_helper import AutograderHelper
```

[2]:
```
# Autograder cell. This cell is worth 0 points.
# This cell has hidden code used to configure the autograder.
```

1.1 Question 0 – Word Count

1.1.1 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:

[3]:
```
# Clean up previous versions of word_count.py and its output files
!rm -f -v word_count*
```

[4]:
```
%%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
```

Now let's look at the output of running the script with that file. Note that we are using the `tee` command here to make things easier for the autograder. The `|` character sends the output of our script to the `tee` command which prints it to the display, and also writes the script output to the given file name ("file_stats_output.tsv" in this case).

[5]:
```
!python word_count.py ../../assets/data/gutenberg/short.t1.txt | tee␣
 ↪word_count_output_short.tsv
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/word_count.jovyan.20241125.023615.059714
Running step 1 of 1…
job output is in /tmp/word_count.jovyan.20241125.023615.059714/output
Streaming final output from
/tmp/word_count.jovyan.20241125.023615.059714/output…
"words" 1822
"lines" 200
"chars" 10653
Removing temp directory /tmp/word_count.jovyan.20241125.023615.059714…
```

A note on interpreting MRJob output...

Since we're using the `tee` command above, the standard output (`stdout`) from the MRJob command (`python word_count.py`) is redirected to the file `word_count_output_short.tsv`, and that is what the grader cell below is reading. This is why the grader cells that show answers are only showing the data output. The data lines for the output above should be:

```
"chars" 10653
"lines" 200
"words" 1822
```

The other output before and after the data lines are logging messages from MRJob. These are written to standard error (`stderr`) and are visible in the notebook, but not in the data output saved to the file.

[6]:
```
# This cell deliberately includes answers to provide guidance on how questions␣
 ↪in this assignment are graded.

correct = AutograderHelper.parse_mrjob_output(
    """
"chars"         10653
"lines"         200
"words"         1822
```

```
""".strip().split("\n")
)

submitted = AutograderHelper.parse_mrjob_output_file("word_count_output_short.
 ↪tsv")

AutograderHelper.assert_same_shape(correct, submitted)
AutograderHelper.assert_same_rows(correct, submitted)
```

1.2 Question 1 – Most-used Word

A note about word splitting:

The `split()` method used in the example above, breaks on white space and there are cases where this isn't ideal:

```
"My dog--Luna--is barking".split()          # --> ['My', 'dog--Luna--is', 'barking']
```

So in the following exercises, we will use a regular expression to split words instead. Note that the starter code below provides a function `splitter()`. We can use this to get better word splitting:

```
splitter("My dog--Luna--is barking")        # --> ['My', 'dog', 'Luna', 'is', 'barking']
```

1.2.1 Task: Complete the most-used word count implementation

Your task in this exercise is to complete the implementation of the `mapper_get_words()` method below. It should: - Use `splitter()` function to split words - Yield 2-tuples that are key-value pairs, where the key (first item in the tuple) is the word to be counted. - Only yield for words that are NOT in the STOPWORDS set - This should be case insensitive, meaning that, for example, "Dog" and "dog" will be processed together.

Note that you should only need to modify the `mapper_get_words` method.

A note about debugging in this context...

- **Use `debug()` instead of `print()`** Since we are running these MRJob scripts and capturing their output for grading, if you use the standard print() function for debugging, your message will end up in the data output which will corrupt the results. So instead use the debug() function that is defined at the top of the starter code for each of the scripts below. It works mostly like print() does, but its output goes to the stderr interface instead of stdout. This means you'll see it like a normal print() in the output of a notebook cell, but those message won't end up in the data output.
- **Double check the data output** If you are getting unexpected results from the grader, one thing to double check is the output date file directly. For example, the first output file by this notebook is "word_count_output_short.tsv" (above). You can open that file and make sure its contents are what you would expect (in that case three lines with the keys "chars", "lines", and "words").

[7]:
```
# Clean up previous versions of most_used_word.py and its output files
!rm -f -v most_used_word*
```

[10]:
```
%%file most_used_word.py
from mrjob.job import MRJob
from mrjob.step import MRStep
import re

from sys import stderr

# See the note above about debugging
def debug(*msg, **kwargs):
    """Print debugging message to standard error."""
    print(*msg, file=stderr, **kwargs)


def splitter(text):
    WORD_RE = re.compile(r"[\w']+")
    return WORD_RE.findall(text)


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'
}


class MRMostUsedWord(MRJob):
    def steps(self):
```

```
        return [
            MRStep(mapper=self.mapper_get_words,
                   reducer=self.reducer_count_words),
            MRStep(reducer=self.reducer_find_max_word)
        ]

    def mapper_get_words(self, _, line):
        # Use splitter to split the line into words
        words = splitter(line)
        #raise NotImplementedError()

        # Yield words that are not in the STOPWORDS set
        for word in words:
            word = word.lower()  # Convert to lowercase for case insensitivity
            if word not in STOPWORDS:
                yield (word, 1)

    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()
    debug("Run time:", end - start, "seconds")
```

```
Overwriting most_used_word.py
```

Now let's run this script on a small file…

[11]:
```
!python most_used_word.py ../../assets/data/gutenberg/t3.lewis.txt | tee␣
↪most_used_word_output_lewis.tsv
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/most_used_word.jovyan.20241125.032018.945038
Running step 1 of 2…
```

```
Running step 2 of 2…
job output is in /tmp/most_used_word.jovyan.20241125.032018.945038/output
Streaming final output from
/tmp/most_used_word.jovyan.20241125.032018.945038/output…
1334    "river"
Removing temp directory /tmp/most_used_word.jovyan.20241125.032018.945038…
Run time: 1.8390789031982422 seconds
```

[12]:
```
# Autograder cell. This cell is worth 1 point (out of 20). This cell does not␣
 ↪contain hidden tests.
# This cell deliberately includes answers to provide guidance on how this␣
 ↪question is graded.

correct = AutograderHelper.parse_mrjob_output(
    """
1334        "river"
""".strip().split("\n")
)

submitted = AutograderHelper.
 ↪parse_mrjob_output_file("most_used_word_output_lewis.tsv")

AutograderHelper.assert_same_shape(correct, submitted)
AutograderHelper.assert_same_rows(correct, submitted)
```

Now let's run this script on a larger file…

[13]:
```
!python most_used_word.py ../../assets/data/gutenberg/t8.shakespeare.txt | tee␣
 ↪most_used_word_output_shakespeare.tsv
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/most_used_word.jovyan.20241125.032032.418489
Running step 1 of 2…
Running step 2 of 2…
job output is in /tmp/most_used_word.jovyan.20241125.032032.418489/output
Streaming final output from
/tmp/most_used_word.jovyan.20241125.032032.418489/output…
5479    "thou"
Removing temp directory /tmp/most_used_word.jovyan.20241125.032032.418489…
Run time: 4.337578058242798 seconds
```

[14]:
```
submitted = AutograderHelper.parse_mrjob_output_file(
    "most_used_word_output_shakespeare.tsv"
)

assert len(submitted) == 1, "The submission is not the correct length."
```

[15]:
```
# Autograder cell. This cell is worth 5 points (out of 20). This cell contains␣
↪hidden tests.
```

1.3 Question 2 – Syllable Count

The `syllables` package (which is pre-installed for you) has an `estimate()` method you can use to get an estimated count of syllables for a given word.

A couple of examples:

[32]:
```
import syllables

syllables.estimate("funny")
```

[32]: 2

[33]:
```
syllables.estimate("strengths")
```

[33]: 1

These are just estimates, so you'll see inaccurate counts from this package, for example, with "temperature". Don't worry about this. For this exercise, we just care about the result from `syllables.estimate()`, not how accurate it is.

[34]:
```
syllables.estimate("temperature")
```

[34]: 5

Also note that this function is sensitive to the case of the input, and can return different counts based on capitalization.

[35]:
```
syllables.estimate("Unfortunately")  # Note the upper-case "U"
```

[35]: 5

[36]:
```
syllables.estimate("unfortunately")  # Note the lower-case "u"
```

[36]: 6

1.3.1 Task: Write an MRJob script

Your task is to write a MRJob script that finds the 10 words that have the most syllables from the input file. This top-ten list should be sorted first by the syllable count, then by the word in alphabetic order. It should:

- Use the `splitter()` function to split words
- Only process words that are NOT in the STOPWORDS set

- This should be case insensitive, meaning that, for example, “Dog” and “dog” should be processed together.
- Use the `sort_results()` function (see below) to sort the final results.

Sorting the Results In order to simplify interpretation of the results, use the provided `sort_results()` function. For example: if our input is:

```
The dog sleeps by the fireplace.
```

and our mapper gives us a result like this:

```
[
    (1, "dog"),
    (3, "fireplace"),
    (1, "the"),
    (1, "sleeps"),
    (1, "by"),
]
```

The `sort_results()` function will sort that like this:

```
[
    (3, "fireplace"),
    (1, "by"),
    (1, "dog"),
    (1, "sleeps"),
    (1, "the"),
]
```

[37]:
```
# Clean up previous versions of top_10_syllable_count.py and its output files
!rm -f -v top_10_syllable_count*
```

```
removed 'top_10_syllable_count_output_short.tsv'
removed 'top_10_syllable_count.py'
```

[82]:
```
%%file top_10_syllable_count.py

import re
from sys import stderr

from mrjob.job import MRJob
from mrjob.step import MRStep

import syllables

def debug(*msg, **kwargs):
    """Print debugging message to standard error."""
    print(*msg, file=stderr, **kwargs)

def splitter(text):
    WORD_RE = re.compile(r"[\w']+")
```

```
    return WORD_RE.findall(text)

def sort_results(results):
    """
    Sorts a list of 2-tuples descending by the first value in the
    tuple, ascending by the second value in the tuple.
    """
    return sorted(results, key=lambda k: (-k[0], k[1]))

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'
}

class MRMostSyllables(MRJob):
    def steps(self):
        return [
            MRStep(mapper=self.mapper_get_words,
                   reducer=self.reducer_find_top_ten)
        ]

    def mapper_get_words(self, _, line):
        words = splitter(line)
        for word in words:
            word = word.lower()
            if word.isalpha() and word not in STOPWORDS:
                syllable_count = syllables.estimate(word)
                yield None, (syllable_count, word)
```

```
    def reducer_find_top_ten(self, _, syllable_word_pairs):
        unique_results = {tuple(pair) for pair in syllable_word_pairs}  # Use a␣
 ↪set of tuples to remove duplicates
        sorted_results = sort_results(unique_results)
        for count, word in sorted_results[:10]:
            yield count, word

if __name__ == '__main__':
    MRMostSyllables.run()

# raise NotImplementedError()
```

```
Overwriting top_10_syllable_count.py
```

[83]:
```
!python top_10_syllable_count.py test.txt | tee␣
 ↪top_10_syllable_count_output_test.tsv
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/top_10_syllable_count.jovyan.20241125.052240.574336
Running step 1 of 1…
job output is in /tmp/top_10_syllable_count.jovyan.20241125.052240.574336/output
Streaming final output from
/tmp/top_10_syllable_count.jovyan.20241125.052240.574336/output…
6       "phonotelephote"
6       "plenipotentiaries"
6       "revolutionized"
6       "unfortunately"
6       "unimaginable"
5       "accumulator"
5       "accumulators"
5       "agriculture"
5       "civilization"
5       "communicate"
Removing temp directory
/tmp/top_10_syllable_count.jovyan.20241125.052240.574336…
```

Now let's run this script on a small file…

[84]:
```
!python top_10_syllable_count.py ../../assets/data/gutenberg/short.t1.txt | tee␣
 ↪top_10_syllable_count_output_short.tsv
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/top_10_syllable_count.jovyan.20241125.052242.770588
Running step 1 of 1…
job output is in /tmp/top_10_syllable_count.jovyan.20241125.052242.770588/output
Streaming final output from
```

```
/tmp/top_10_syllable_count.jovyan.20241125.052242.770588/output…
6       "phonotelephote"
6       "plenipotentiaries"
6       "revolutionized"
6       "unfortunately"
6       "unimaginable"
5       "accumulator"
5       "accumulators"
5       "agriculture"
5       "civilization"
5       "communicate"
Removing temp directory
/tmp/top_10_syllable_count.jovyan.20241125.052242.770588…
```

[85]:
```
# Autograder cell. This cell is worth 2 points (out of 20). This cell does not␣
 ↪contain hidden tests.
# This cell deliberately includes answers to provide guidance on how this␣
 ↪question is graded.

correct = AutograderHelper.parse_mrjob_output(
    """
6       "phonotelephote"
6       "plenipotentiaries"
6       "revolutionized"
6       "unfortunately"
6       "unimaginable"
5       "accumulator"
5       "accumulators"
5       "agriculture"
5       "civilization"
5       "communicate"
""".strip().split("\n")
)

submitted = AutograderHelper.parse_mrjob_output_file(
    "top_10_syllable_count_output_short.tsv"
)

AutograderHelper.assert_same_shape(correct, submitted)
AutograderHelper.assert_same_rows(correct, submitted)
```

Now let's run this script on a larger file…

[86]:
```
!python top_10_syllable_count.py ../../assets/data/gutenberg/t5.churchill.txt |␣
 ↪tee top_10_syllable_count_output_churchill.tsv
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
```

```
Creating temp directory /tmp/top_10_syllable_count.jovyan.20241125.052246.357588
Running step 1 of 1…
job output is in /tmp/top_10_syllable_count.jovyan.20241125.052246.357588/output
Streaming final output from
/tmp/top_10_syllable_count.jovyan.20241125.052246.357588/output…
8       "incommunicability"
8       "overcapitalization"
7       "apologetically"
7       "authoritatively"
7       "characteristically"
7       "communicativeness"
7       "corroboratively"
7       "disproportionately"
7       "imaginatively"
7       "impenetrability"
Removing temp directory
/tmp/top_10_syllable_count.jovyan.20241125.052246.357588…
```

[80]:
```
submitted = AutograderHelper.parse_mrjob_output_file(
    "top_10_syllable_count_output_churchill.tsv"
)

assert len(submitted) == 10, "The submission is not the correct length."
```

[81]:
```
# Autograder cell. This cell is worth 12 points (out of 20). This cell contains␣
↪hidden tests.
```

[ ]:
```

```