Machine Problem 0: Java/Python Programming Essentials

Overview

Welcome to the Java/Python Programming Assignment. You may choose to complete this machine problem in **either** Java or Python. Please choose the language that you prefer. This assignment evaluates the Java/Python programming skills you will need to work on the upcoming assignment.

Java submission

If you choose to do this assignment in Python, skip this part and go to **"Python submission" part below

1 Requirements

This assignment will be graded based on **JDK 8**

If you are new to Java programming language, take a look at:

https://docs.oracle.com/javase/tutorial/

2 Procedures

Step 1: Download the project files from

https://github.com/UIUC-public/MP0

Step 2: Edit the java template file: **MP0.java**. All you need to edit is the part marked with **TODO**. You will find more information on what to do in the next section "Exercise: Select Word Count - Java".

Step 3: Compile the file and run it on the provided input. Your program should use a number as an argument. The detailed requirements for this assignment are explained in "Exercise: Select Word Count - Java" below. The following is an example of a command to run the application, using '1' as the argument. The command line might need some modifications for your platform:

```
$ javac MP0.java
$ cat input.txt | java MP0 1
```

Don't change the filename, class name, or the main function.

Step 4: After you are done with the assignment, compress your **MP0.java** to a .zip file named as "**MP0.zip**", submit your "**MP0.zip**".

3 Exercise: Select Word Count - Java

In this exercise, you are to implement an application to find the top **20** words used in Wikipedia titles (provided). To make the implementation easier, we have provided a boilerplate for this exercise in **MP0.java** file.

All you need to do is to make necessary changes in the file.

Your application takes a huge list of Wikipedia titles (one in each line) as an input. You need to make some preprocessing on the input, and then return the top **20** words that appear the most in a selection of titles. One possible procedure is the following:

1. Divide each sentence into a list of words using delimiters provided in the "**delimiters**" variable.

One possible approach is to use StringTokenizer. For more information see:

https://docs.oracle.com/javase/7/docs/api/java/util/StringTokenizer.html

2. Make all the tokens lowercase and remove any tailing and leading spaces.

More information about string operations can be found in:

https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

3. Ignore all common words provided in the "**stopWordsArray**" variable.

One possible approach is to use lists. For more information see:

https://docs.oracle.com/javase/7/docs/api/java/util/List.html#contains(java.lang.Object)

4. Keep track of word frequencies. To make the application more interesting, you have to process **only the titles with certain indexes**. These indexes are accessible using the "**getIndexes**" method, which returns an Integer Array with 0-based indexes to the input file. It is possible to have an index appear several times. In this case, just process the index multiple times.

One possible approach is to use Maps in java. For more information see:

https://docs.oracle.com/javase/7/docs/api/java/util/Map.html

5. Sort the words by frequency in a descending order. If two words have the same number count, use the **lexigraphy**. For example, the following is sorted:

```
{(Orange, 3), (Apple, 2), (Banana, 2)}
```

6. Print out the top 20 items from the sorted list as a String Array.

Here is the output of this application if **"0"** is used for the argument:

**note that the output may be different on your machine, please rely on auto-grader to check the correctness of your solution

```
list
de
state
disambiguation
new
county
school
john
route
river
township
saint
united
song
south
station
album
city
c
st
```

Here is the output of this application if “**1**” is used for the argument:

**note that the output may be different on your machine, please rely on auto-grader to check the correctness of your solution

```
list
de
state
school
new
disambiguation
john
route
album
county
st
river
film
highway
township
district
city
high
c
station
```

Python submission

If you choose to do this assignment in Java, skip this part and go to **"Java submission" part above

1 Requirements

This assignment will be graded based on **Python 2.7**

If you are new to Python programming language, take a look at:

https://docs.python.org/2.7/tutorial/

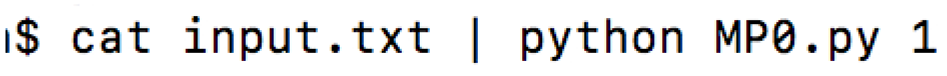
2 Procedures

Step 1: Download the project files from

https://github.com/UIUC-public/MP0

Step 2: Edit the java template file: **MP0.py**. All you need to edit is the part marked with **TODO**. You will find more information on what to do in the next section "Exercise: Select Word Count - Python".

Step 3: Run the file on the provided input. Your program should use a number as an argument. The detailed requirements for this assignment are explained in "Exercise: Select Word Count - Python" below. The following is an example of a command to run the application, using '1' as the argument. The command line might need some modifications for your platform:



```
$ cat input.txt | python MP0.py 1
```

Don't change the filename.

Step 4: After you are done with the assignment, compress your "**MP0.py**" file to a .zip file named as "**MP0.zip**", submit "**MP0.zip**".

3 Exercise: Select Word Count - Python

In this exercise, you are to implement an application to find the top **20** words used in Wikipedia titles (provided). To make the implementation easier, we have provided a boilerplate for this exercise in **MP0.py** file.

All you need to do is to make necessary changes in the file.

Your application takes a huge list of Wikipedia titles (one in each line) as an input. You need to make some preprocessing on the input, and then return the top **20** words that appear the most in a selection of titles. One possible procedure is the following:

1. Divide each sentence into a list of words using delimiters provided in the "**delimiters**" variable; Make all the tokens lowercase (including special characters) and remove any tailing and leading spaces.

More information about string operations can be found in:

https://docs.python.org/2/library/string.html

One possible approach to deal with special (non-English) characters is to use decode() and encode(). For more information see:

https://docs.python.org/2/howto/unicode.html

3. Ignore all common words provided in the “**stopWordsList”** variable.

4. Keep track of word frequencies. To make the application more interesting, you have to process **only the titles with certain indexes**. These indexes are accessible using the “**getIndexes**” method, which returns an Integer List with 0-based indexes to the input file. It is possible to have an index appear several times. In this case, just process the index multiple times.

More information about list operations can be found in:

https://docs.python.org/2/tutorial/datastructures.html#more-on-lists

One possible approach is to use Dictionaries in Python. For more information see:

https://docs.python.org/2/tutorial/datastructures.html#dictionaries

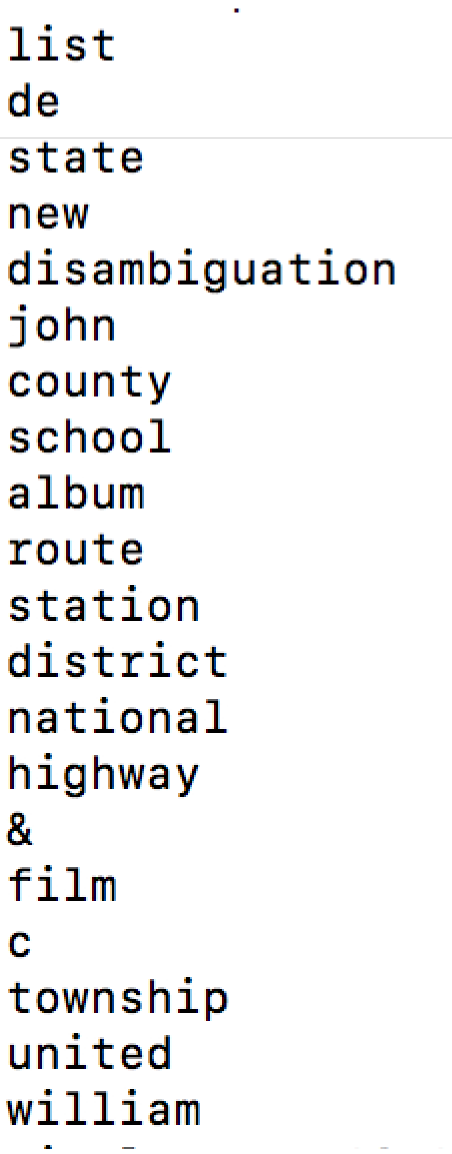
5. Sort the words by frequency in a descending order. If two words have the same number count, use the **lexigraphy**. For example, the following is sorted:

```
{(Orange, 3), (Apple, 2), (Banana, 2)}
```

6. Print out the top 20 items from the sorted list.

Here is the output of this application if **"0"** is used for the argument:

**note that the output may be different on your machine, please rely on auto-grader to check the correctness of your solution



```
list
de
state
new
disambiguation
john
county
school
album
route
station
district
national
highway
&
film
c
township
united
william
```

Here is the output of this application if “**1**” is used for the argument:

**note that the output may be different on your machine, please rely on auto-grader to check the correctness of your solution

```
list
de
state
new
county
john
disambiguation
school
album
river
route
c
district
film
st
national
united
highway
township
&
```

✓ Complete