Writing MapReduce Application in Python

하둡 맵리듀스 어플리케이션을 제작할 때 비단 자바 뿐만 아니라 다양한 언어를 구사할 수 있습니다. 여기서는 데이터 과학에서 사용되고 있는 파이썬 언어로 WordCount 어플리케이션을 제작하고자 합니다.

1. 맵 파일 생성하기

맵 단계에서 맵 스크립트를 작동시키고자 mapper.py 파일을 코딩하게 됩니다. 파일을 생성하고자 리눅스 명령어로 맵 리듀스 디렉토리를 하나 생성합니다.

여기서 ‘stdin’은 standard input 으로 맵과 리듀스 단계 코드 사이로 데이터를 패싱하는데 도움을 주는 하둡 스트리밍 API 이라고 합니다. 그 API는 두 가지로 ‘Stdout’ (standard output)은 인풋을 읽는 과정의 sys.stdin 의 파이선을 사용하며 sys.stdout 을 사용하고자 아웃풋을 프린트하는 과정을 함께 사용할 수 있죠. STDIN는 인풋을 단어별로 쪼개는 과정이라고 생각하시면 되며, STDOUT은 그들의 단어들을 카운트하는 맴의 라인 리스트라고 보시면 됩니다.

만약 어플리케이션 실행 중 파이썬이 실행되지 않는다면

그래서 “/usr/bin/env: ‘python’: No such file or directory” 이런 메시지를 만나거나

“Python --version” 실행으로 파이썬을 찾을 수 없다면

그래서 “Command ‘python’ not found”라는 메시지를 만난다면

리눅스 터미널에서 sudo apt-get install python-is-python3을 실행시키면 됩니다.

~/hadoop$ python --version

~/hadoop$ sudo apt update

~/hadoop$ sudo apt-get install python-is-python3

~/hadoop$ python3 -V

$ pwd

$ cd $HADOOP\_HOME

~/hadoop$ mkdir mapreduce

~/hadoop$ cd mapreduce

~/hadoop/mapreduce$ pwd

~/hadoop/mapreduce$ nano /home/neuavenue/hadoop/mapreduce/mapper.py

|  |
| --- |
| #!/usr/bin/env python  import sys  # STDIN(Standard Input) 기초 입력(인풋데이터)로 부터 온 데이터 라인별로 실행  for line in sys.stdin:      # 문장의 스페이스 꼬리부분들과 앞부분을 제거하는 함수      line = line.strip()     #문장 내의 라인들을 쪼개는 작업      words = line.split()    # 해당 모든 단어별로 임의의 숫자를 증가시켜면서 STDOUT(Standard Output)     # 기본 아웃풋 곧 출력물이자 결과값을 읽는 작업..     # 파일 내의 모든 단어 갯수만큼 루프를 돌면서 1로 밸류값을 지정하고,     # 모든 단어를 키 값으로 넣어 전체를 나열하여 결과값으로 프린트.     # %s - 단어들을 스트링으로 출력하고자 할 때 사용    # %t -  프린트할 때 탭을 출력하고자 할 때 사용      for word in words:          print( '%s\t%s' % (word, 1)) |

~/hadoop/mapreduce$$ ls -l

~/hadoop/mapreduce$$ chmod +x mapper.py

1. 리듀스 파일 생성하기

~/hadoop/mapreduce$ nano reducer.py

|  |
| --- |
| #!/usr/bin/env python  # 데이터 구성요소들을 소트하고자 할 때 사용하는 것으로 # 키 밸류 쌍(key-value pair)으로 검색(retrieve)하고자 할 때 사용함. # 결국 키 밸류 쌍을 구성하고자 itemgetter를 사용함.  from operator import itemgetter  # sys  import sys  current\_word = None # 값을 초기화할 경우 혹은 정의되지 않는 변수값 정의  current\_count = 0 # 현재 변수를 숫자로 초기화하며 숫자 0 제로를 정의하는 숫자형 변수  word = None  # STDIN (맵 단계를 거쳐 나온 결과값이 이제 입력값으로 시작됨  for line in sys.stdin:     # 빈 스페이스들을 제거하는 과정      line = line.strip()     # 맵 단계에서 가져온 입력값(Input data)을 파싱하는 과정     # 개별 단어들을 키로서 파싱할 때 ‘\t’ 곧 탭을 다음에 넣으라는 지시    # 동시에 별류(Value)는 모든 값을 1로 지정.      word, count = line.split('\t', 1)     # 스트링으로 가지고 온 카운트 값을 Int 값으로 변환하여 컨버팅하는 과정      try:          count = int(count)      except ValueError:         # 숫자가 없는 카운트 값은 무시하고 진행          continue     # if 문장으로 비교하고자 하는 현재 텍스트 단어(current\_word)가     # 맵에서 가지고 온 단어(word)와    # 비교하여 같으면 current\_count 값을 count 값 만큼 증가시키라는 말,     # 만약 같은 단어가 있다면 하나씩 더 증가하게 됨.     # 비교하고자 하는 현재 텍스트와 맵에서 가지고 온 단어와 다르면    # 화면에 출력하게 된다.      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     # 최종적으로 비교하려는 현재 텍스트 단어와 맵에서 가지고 온 단어가 같은     # 값이라면 최종적으로 단순히 출력하게 하려는 코드.  if current\_word == word:      print( '%s\t%s' % (current\_word, current\_count)) |

|  |
| --- |
| #!/usr/bin/env python  from operator import itemgetter  import sys  current\_word = None  current\_count = 0  word = None  for line in sys.stdin:  line = line.strip()  word, count = line.split('\t', 1)  try:       count = int(count)  except ValueError:       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)) |

~/hadoop/mapreduce$ ls -l

~/hadoop/mapreduce$ chmod +x reducer.py

~/hadoop/mapreduce$ ls -l

1. 파이썬 코드 테스트

맵과 리듀스 파이선 파일들을 점검하고자 코드를 테스트하는 방법을 배울 것입니다.

~/hadoop/mapreduce$ echo Here is a lion that is roaring my ear | ~/hadoop/mapreduce/mapper.py

~/hadoop/mapreduce$ echo Here is a lion that is roaring my ear | ~/hadoop/mapreduce/mapper.py | sort -k1,1 | ~/hadoop/mapreduce/reducer.py

1. 인풋 데이터(입력 데이터) 다운로드

구글 혹은 파이어폭스 웹 브라우저에서 Gutenberg.org 사이트로 이동하셔서

원하시는 ebook 파일 두 개 이상을 다운 받습니다.

~/hadoop/mapreduce$mkdir input

<http://www.gutenberg.org/ebooks/64846>

<http://www.gutenberg.org/cache/epub/13998/pg13998.txt>

혹은 온라인 북의 컨텐츠를 복사하셔서 우분투 리눅스 로컬 시스템에 새로운 파일을 생성합니다.

~/hadoop/mapreduce/input$ nano the-plant-of-illusion-by-Donald-A-Wollheim.txt

~/hadoop/mapreduce/input$ nano Ireland-And-The-Home-Rule-Movement.txt

1. 인풋 데이터 HDFS 파일 시스템 로딩

실제 맵리듀스 잡을 실행하기 전, ebook 콘텐츠 인풋 파일을 하둡 분산 파일 시스템으로 데이터 로딩합니다. 이에 앞서, 하둡 파일 시스템에 새로운 디렉토리를 생성합니다.

~/hadoop/mapreduce$ hdfs dfs -mkdir /user/mapreduce

~/hadoop/mapreduce$ hdfs dfs -ls /user/mapreduce

~/hadoop/mapreduce$ hdfs dfs -copyFromLocal ~/hadoop/mapreduce/input /user/mapreduce/

1. 맵리듀스 잡 실행

~/hadoop/mapreduce$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.2.1.jar \

-file ~/hadoop/mapreduce/mapper.py -mapper ~/hadoop/mapreduce/mapper.py \

-file ~/hadoop/mapreduce/reducer.py -reducer ~/hadoop/mapreduce/reducer.py \

-input /user/mapreduce/input/\* -output /user/mapreduce/output-ebook