과제04: 데이터 처리 실습

- 리눅스 기본 텍스트 유틸리티와 AWK를 사용하는 연습 과제이다.
- 주어진 안내를 따라 기본적인 실습을 수행하며 각 명령이 어떻게 작동하는지 살펴보고 이해한 바를 간략히 설명하여 보고서를 작성한다. 특히 파이프로 연결된 명령들의 각 단계에서 어떤 형태의 데이터를 주고 받으며 작동하는지 주의 깊게 확인한다.
- 아래 주어진 안내는 다운로드 받은 데이터 중 하나의 디렉토리만 다루고 있다. 나머지 데이터에 대해서도 유사한 분석을 수행한다. 명령들을 변형 시켜 사용하여 보고 다양한 유용한 정보를 추출하여 분석하여 본다.
- 보고서에서 출력 결과를 지나치게 길게 붙여넣지 않도록 한다. 명령과 출력 결과만 나열하지 않도록 한다. 각 데이터 처리에 설명을 적절히 작성하도록 한다.
- 보고서는 마크다운으로 작성하고 hw04.pdf 파일로 제출한다.

1. 데이터

리눅스 텍스트 처리 도구를 이용한 실제 데이터 처리 실습을 해 봅시다. 데이터는 AI 허브에 공개된 인공지능 학습용 데이터를 실습 대상으로 삼겠습니다. AI 허브는 과기부 산하 한국지능정보사회진흥원(NIA)에서 운영하는 인공지능 관련 정보 공유 플랫폼입니다. 정부 지원 사업으로 구축된 다양한 분야의 인공지능 학습용 데이터가 공개되어 있습니다.

- https://aihub.or.kr/

가입하면 자유롭게 데이터를 다운로드 받을 수 있습니다. 수정과 재배포에는 제약이 있으니 주의하셔야 합니다. 다음 데이터가 실습 예제입니다.

- 음성/자연어 > 도서자료 요약

도서자료 요약 데이터를 다운로드 받고 압축을 풀어 봅시다.

```
$ cd '도서자료 요약'
$ ls
Training  Validation
$ cd Training
$ ls
'[원천]도서요약_train.zip'
$ 7z l '[원천]도서요약_train.zip'

... (파일 목록 출력) ...

$ 7z x '[원천]도서요약_train.zip'
$ ls
'[원천]도서요약_train.zip'    기타    예술    기술과학    사회과학
```

2. 파일 개수

압축을 풀면 4개의 디렉토리가 생깁니다. 각 디렉토리에 들어가서 어떤 파일들이 들어 있는지 살펴보세요. 예를 들어, 다음처럼 `less` 명령을 내려보세요. 우선 첫번째 파일의 내용이 화면에 나타납니다. 다음 파일을 보려면 콜론(`:`)을 입력한 후 `n`을 입력하면 됩니다.

```
$ cd 기타
$ less *.json
```

파일이 몇 개 있는지 확인하려면 다음처럼 명령을 내리면 됩니다. 모두 6753개의 파일이 있다는 것을 알 수 있습니다.

```
$ ls | wc
   6753    6753  180265
```

상위 디렉토리로 올라가서 다음 명령을 내려보세요. 각 폴더에 몇 개의 파일이 있는지 알 수 있습니다. 사회과학 디렉토리에 115,440개의 파일이 들어있네요.

```
$ cd ..
$ find . -type file | cut -d/ -f2 | sort | uniq -c
      1 [원천]도서요약_train.zip
   6753 기타
  13891 예술
  23918 기술과학
 115440 사회과학
```

이렇게 파이프(`|`)로 연결된 명령에 익숙하지 않으시다면 하나씩 단계적으로 어떤 결과가 나오는지 살펴보세요. 다음처럼 순서대로 명령을 늘려가며 확인해 보세요.

```
$ find .
$ find . | cut -d/ -f2
$ find . | cut -d/ -f2 | sort
$ find . | cut -d/ -f2 | sort | uniq -c
```

3. 주제별 빈도

사회과학 도서 디렉토리로 들어가서 살펴봅시다. 파일명이 `PCY_2020`으로 시작하는 경우에 한국십진분류표(KDC)에 따른 주제별 빈도는 다음과 같습니다.

```
$ cd 사회과학
$ grep '"kdc_label":' PCY_2020* | cut -d: -f3 | sort | uniq -c | sort -nr
    435  "안전관리",
    412  "외교",
    367  "국민권익·인권",
    224  "보육·가족및여성",
    222  "법무및검찰",
    213  "유아및초·중등교육",
    198  "에너지및자원개발",
    149  "병무행정",
    133  "통일",
     98  "국정홍보",
     94  "국정운영",
     59  "교육일반",
     42  "일반행정",
     37  "산업·중소기업일반",
     26  "평생·직업교육",
     23  "노인·청소년",
     19  "취약계층지원",
     19  "고용노동",
     19  "금융",
      5  "고등교육",
      5  "철도",
      2  "기획재정",
```

파일명이 `PCY_2019`로 시작하는 경우의 주제에 따른 빈도는 다음과 같습니다. 주제별 분포가 연도에 따라 차이가 나는 것 같네요.

```
$ grep '"kdc_label":' PCY_2019* | cut -d: -f3 | sort | uniq -c | sort -nr
   1546  "고용노동",
    569  "국정운영",
    536  "에너지및자원개발",
    512  "산업진흥·고도화",
    489  "법무및검찰",
    487  "외교",
    478  "국민권익·인권",
    434  "일반행정",
    414  "병무행정",
    384  "보육·가족및여성",
    384  "평생·직업교육",
    368  "교육일반",
    339  "취약계층지원",
    172  "노인·청소년",
    156  "통일",
     85  "안전관리",
     85  "국정홍보",
     61  "유아및초·중등교육",
     57  "법제",
     56  "산업·중소기업일반",
     38  "경찰",
     36  "지방행정·재정지원",
     33  "해운·항만",
     28  "주택",
     18  "무역및투자유치",
     15  "사회복지일반",
     15  "물류등기타",
     11  "지역및도시",
     10  "항공·공항",
      9  "기초생활보장",
      6  "기획재정",
      3  "산업금융",
```

그런데 파일명은 `PCY_2020`으로 시작하지만 실제 출판년도는 2018, 2019, 2020인 도서들이 섞여있는 것을 확인할 수 있습니다.

```
$ grep '"published_year":' PCY_2020* | cut -d: -f3 | sort | uniq -c
     46  "2018",
   2249  "2019",
    506  "2020",

$ grep '"published_year":' PCY_2019* | cut -d: -f3 | sort | uniq -c
     61  "2011",
      2  "2016",
    551  "2017",
   4972  "2018",
   2248  "2019",
```

4. 출판연도별 파일 개수

이 디렉토리에는 파일이 너무 많아 모든 파일을 한번에 처리하려고 하면 다음과 같은 오류가 날 수 있습니다.

```
$ grep '"published_year":' *.json
-bash: /usr/bin/grep: Argument list too long
```

다음과 같이 `xargs` 명령을 이용해 이 문제를 피할 수 있습니다. 출판연도별 도서의 수는 다음과 같습니다.

```
$ ls | xargs grep '"published_year":' | cut -d: -f3 | sort | uniq -c
     47  "1991",
     82  "1993",
     45  "1996",
     21  "1997",
     74  "1998",
    190  "1999",
    102  "2000",
    105  "2001",
     82  "2002",
     26  "2003",
    223  "2006",
     16  "2007",
     72  "2008",
     40  "2009",
    185  "2010",
    373  "2011",
    962  "2012",
   2222  "2013",
   7797  "2014",
  18171  "2015",
  21366  "2016",
  12922  "2017",
  10951  "2018",
   6099  "2019",
    806  "2020",
  32461  null,
```

5. 출판연도별 주제 빈도

이제 연도별 주제 빈도를 계산해 봅시다.

```
$ ls | xargs grep '"published_year":' | cut -d: -f3 > ../published_year.txt
$ ls | xargs grep '"kdc_label":' | cut -d: -f3 > ../kdc_label.txt
$ cd ..
$ paste published_year.txt kdc_label.txt | grep '"2020"' | cut -f2 | sort | uniq -c | sort
-nr
    136  "병무행정",
    108  "행정학",
    103  "법무및검찰",
     94  "국정홍보",
     71  "사회학, 사회문제",
     50  "유아및초·중등교육",
     46  "보육·가족및여성",
     42  "일반행정",
     34  "법학",
     29  "경제학",
     26  "정치학",
     19  "외교",
     16  "금융",
     10  "사회과학",
      8  "국방, 군사학",
      7  "통계학",
      6  "교육학",
      1  "풍속, 민속학",

$ paste published_year.txt kdc_label.txt | grep '"2019"' | cut -f2 | sort | uniq -c | sort
-nr
    519  "국민권익·인권",
    459  "안전관리",
    439  "외교",
    434  "일반행정",
    402  "에너지및자원개발",
    373  "경제학",
    349  "교육학",
    326  "법무및검찰",
    296  "보육·가족및여성",
    280  "사회학, 사회문제",
    274  "병무행정",
    251  "고용노동",
    243  "행정학",
    234  "국정운영",
    171  "정치학",
    169  "유아및초·중등교육",
    133  "통일",
    133  "법학",
     97  "교육일반",
     89  "국정홍보",
     88  "평생·직업교육",
     57  "법제",
     49  "노인·청소년",
     48  "취약계층지원",
     37  "산업·중소기업일반",
     36  "지방행정·재정지원",
     33  "사회과학",
     20  "국방, 군사학",
```

```
     16  "주택",
     11  "지역및도시",
     10  "항공·공항",
      6  "무역및투자유치",
      5  "고등교육",
      5  "철도",
      4  "기획재정",
      3  "금융",
```

6. 도서 설명의 단어 빈도

도서 내용에 많이 나타나는 단어들의 빈도를 계산할 수도 있습니다. 파일명이 `PCY_2020`으로 시작하는 경우만으로 한정하면 다음과 같습니다. 이때 단어는 단순히 공백문자로 구분된 단위를 말합니다. 고빈도의 단어들은 기능적인 요소들인 경우가 많아 텍스트의 내용을 이해하는 데에는 크게 도움이 되지 않습니다. 이러한 기능적 단어들을 제외하고 보면 좋지만 그러기 위해서는 형태소 분석, 키워드 추출과 같은 자연어처리가 필요합니다.

```
$ grep '"passage":' PCY_2020*  | cut -d: -f3 | awk '{for(i=1;i<=NF;i++) print $i}' | sort
| uniq -c | sort -nr | head -20
   2924 수
   1751 있다.
   1689 대한
   1635 및
   1467 있는
    891 그
    746 등
    721 이러한
    711 것으로
    697 할
    692 것이다.
    690 위한
    628 이
    621 통해
    572 위해
    551 것이
    547 하는
    541 따라
    491 또한
    471 그리고
```

과제05: 데이터 처리 실습

- 과제04와 하나의 파일로 작성한다. hw04.pdf을 제출하면 된다.
- 리눅스 기본 명령어와 AWK에 추가하여 CSV, JSON 처리 도구를 사용하는 연습 과제이다.
- CSV, JSON 처리 도구는 매우 다양한 기능을 제공한다. 아래 안내에는 간략하게 기본적인 구도만 소개하였다. 도움말을 참고하여 기능들을 사용하는 예시와 설명을 보고서로 작성한다.
- 과제04의 데이터 파일들 전체를 대상으로 데이터 처리를 하는 실습을 수행한다.

- 추가적으로 관심 있는 다른 데이터를 분석하여도 좋다.

7. CSV과 JSON을 위한 명령행 도구

다음은 CSV 형식과 JSON 형식을 명령행에서 처리할 수 있는 도구들입니다. `xsv`와 `cvskit`는 CSV를 읽을 수 있는 도구이며 유사하게 작동합니다. `jq`는 JSON을 읽을 수 있는 도구입니다.

- `xsv` : https://github.com/BurntSushi/xsv
- `cvskit` : https://github.com/wireservice/csvkit
- `jq` : https://github.com/stedolan/jq

CSV 파일은 테이블형 데이터이기 때문에 리눅스 기본 텍스트 유틸리티와 크게 다르지 않게 작동합니다. JSON은 계층적인 구조를 가질 수 있기 때문에 복잡하여 사용이 어렵게 느껴질 수 있습니다.

다음과 같이 예술 디렉토리로 이동하여 `jq` 명령을 이용하여 앞서와 같이 `kdc_label` 값의 빈도를 세어 봅시다.

```
$ jq -r '.metadata.kdc_label' *.json  | sort | uniq -c | sort -nr
   3474 공연예술 및 매체예술
   3065 예술
   2196 문화예술
   1578 문화재
   1477 문화체육관광일반
   1375 관광
    210 음악
    208 체육
    100 오락,스포츠
     93 연극
     52 오락, 운동
     48 회화,도화
     15 건축술
```

위 명령에서 `.metadata.kdc_label`은 JSON 파일에서 찾으려는 속성의 경로를 나타냅니다. 파일을 예시하면 다음과 같습니다. 최상위 중괄호 아래에 `passage`와 `metadata`라는 속성이 있고 `metadata` 안에 `kdc_label`이라는 속성이 있습니다. 이 속성값을 찾아서 출력해준 것입니다. `-r` 옵션은 속성값을 있는 그대로 출력(`--raw-output`)하라는 의미입니다. 이 옵션을 주지 않으면 따옴표 안에 들어있는 형태로 출력됩니다.

```
$ head -12 PCY_202006180835523551_0.json
{
  "passage_id": "PCY_202006180835523551_0",
  "metadata": {
    "doc_id": "PCY_202006180835523551",
    "doc_type": "도서",
    "doc_name": "2019 콘텐츠산업 통계조사 : 2018년 기준",
    "author": null,
    "publisher": "문화체육관광부",
    "published_year": "2020",
    "kdc_label": "문화예술",
    "kdc_code": "600"
  },
```

속성값의 경로 대신에 파이프를 이용할 수도 있다. `.metadata | .kdc_label`이라고 하면 우선 `.metadata`를 추출한 후에 그 결과를 파이프로 넘겨 받아서 `.kdc_label`를 추출하라는 뜻이다.

```
$ jq '.metadata | .kdc_label' *.json  | head
"공연예술 및 매체예술"
"공연예술 및 매체예술"
"공연예술 및 매체예술"
"공연예술 및 매체예술"
"공연예술 및 매체예술"
"공연예술 및 매체예술"
"공연예술 및 매체예술"
"공연예술 및 매체예술"
"공연예술 및 매체예술"
"공연예술 및 매체예술"
```

파이프를 이용하면 다음과 같이 한 속성 안에 들어있는 하위 속성 여러 개를 뽑는 것도 가능하다.

```
$ jq '.metadata | .kdc_label, .published_year' *.json  | head
"공연예술 및 매체예술"
"2007"
"공연예술 및 매체예술"
"2007"
"공연예술 및 매체예술"
"2007"
"공연예술 및 매체예술"
"2007"
"공연예술 및 매체예술"
"2007"
```

결과를 다음처럼 배열(array)로 묶어주면 더 쓸모가 있다.

```
$ jq '.metadata | [.kdc_label, .published_year]' *.json  | head
[
  "공연예술 및 매체예술",
  "2007"
]
[
  "공연예술 및 매체예술",
  "2007"
]
[
  "공연예술 및 매체예술",
```

이렇게 얻어진 단순한 JSON은 CSV, TSV 등으로 변환하는 것이 가능하다.

```
$ jq '.metadata | [.doc_id, .publisher, .published_year, .kdc_label]' *.json  | jq -r
'@tsv'  | head
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
CNTS-00047966809	연극과인간	2007	공연예술 및 매체예술
```

출판년도(`published_year`)와 주제(`kdc_label`)를 추출하여 빈도를 세어보자. `sort`와 `uniq`를 이용하여 빈도를 센 후 `sed` 명령에서 첫번째 컬럼의 빈도값와 두번째 컬럼 사이의 스페이스를 탭으로 바꾸는 정규표현식을 이용하였다. 다음에서 명령의 첫번째 줄의 `$`와 마찬가지로 두번째 줄의 `>`는 입력하지 않는다. 이것은 명령이 이어진다는 표시일 뿐이다.

```
$ jq '.metadata | [.published_year, .kdc_label]' *.json  | jq -r '@csv' |
> sort | uniq -c |  sed -r 's/^[ ]+([0-9]+) /\1,/' > label_year_freq.txt
$ head label_year_freq.txt
538,"2000","공연예술 및 매체예술"
1122,"2001","공연예술 및 매체예술"
442,"2002","공연예술 및 매체예술"
807,"2003","공연예술 및 매체예술"
4,"2004","오락,스포츠"
170,"2004","공연예술 및 매체예술"
48,"2005","회화,도화"
60,"2005","공연예술 및 매체예술"
185,"2007","음악"
272,"2007","공연예술 및 매체예술"
```

위에서 얻어진 CSV 파일을 다음과 같이 정렬할 수 있다. 기본 `sort` 명령을 이용할 수도 있지만 CSV의 묶음 따옴표를 제대로 처리하지 못하므로 CSV 도구가 유용하다.

```
$ xsv sort -n -s1 -NR label_year_freq.txt  | head
1122,2001,공연예술 및 매체예술
990,2018,예술
807,2003,공연예술 및 매체예술
706,,문화예술
538,2000,공연예술 및 매체예술
452,2016,문화체육관광일반
447,2010,예술
447,2019,문화재
442,2002,공연예술 및 매체예술
441,2015,문화체육관광일반
```

CSV 파일을 터미널에서 테이블 형태로 출력해 주는 기능도 있다.

```
$ xsv sort -n -s3 label_year_freq.txt | xsv table | head
15    2015  건축술
538   2000  공연예술 및 매체예술
1122  2001  공연예술 및 매체예술
442   2002  공연예술 및 매체예술
807   2003  공연예술 및 매체예술
170   2004  공연예술 및 매체예술
60    2005  공연예술 및 매체예술
272   2007  공연예술 및 매체예술
19    2009  공연예술 및 매체예술
44    2010  공연예술 및 매체예술
```

기초 통계를 계산해 주는 기능도 제공한다. 연도와 주제에 모두 61개 조합이 존재하며 평균 개수 198개, 최소 1개, 최대 1122개임을 확인할 수 있다.

```
$ xsv stats -n -s1 --cardinality  label_year_freq.txt | xsv table
field  type     sum    min  max   min_length  max_length  mean                stddev
cardinality
0      Integer  13891  1    1122  1           4           198.44285714285715
229.90753642741345  61
```

8. 파일 정리

네 개 디렉토리에 나누어 들어있는 파일중 이름이 동일한 것이 있는지 확인해 봅시다.

```
$ find . -type file | cut -d/ -f3 | sort | uniq -c | awk '$1 > 1'
```

아무 것도 출력되지 않는다면 모든 파일의 이름이 고유하게 서로 다르다는 뜻입니다. 그렇다면 이 파일들을 모두 한 디렉토리에 모아도 문제가 생기지 않을 것입니다. 다음과 같이 모든 파일을 한 디렉토리에 모을 수 있습니다.

```
$ mkdir all
$ mv 기타/* all/
```

파일 수가 많을 때에는 오류가 날 수 있습니다. 다음처럼 `find` 명령을 이용할 수도 있습니다. 해당 디렉토리로 들어가서 이동 또는 복사 명령을 내리면 오류가 나지 않을 수도 있습니다.

```
$ find 사회과학/ -type file -exec mv {} all/ \;
```

이제 이렇게 한 디렉토리에 전체 파일을 모아놓은 상태에서 데이터 처리를 하세요.