Data Processing: Formats and Tools

a topic in

DM565 – Formal Languages and Data Processing

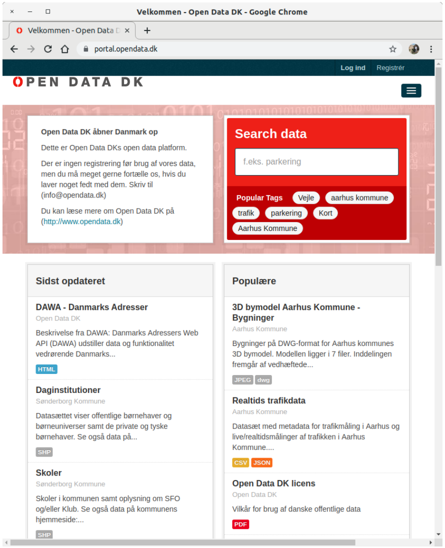
Kim Skak Larsen

Department of Mathematics and Computer Science (IMADA)
University of Southern Denmark (SDU)

kslarsen@imada.sdu.dk

September, 2019

Motivation



Velkommen - Open Data DK - Google Chrome

Velkommen - Open Data D

portal.opendata.dk

Log ind Registrér

OPEN DATA DK

Open Data DK åbner Danmark op

Dette er Open Data DKs open data platform.

Der er ingen registrering før brug af vores data, men du må meget gerne fortælle os, hvis du laver noget fedt med dem. Skriv til (info@opendata.dk)

Du kan læse mere om Open Data DK på (http://www.opendata.dk)

Search data

f.eks. parkering

Popular Tags: Vejle, aarhus kommune, trafik, parkering, Kort, Aarhus Kommune

Sidst opdateret

DAWA - Danmarks Adresser
Open Data DK
Beskrivelse fra DAWA: Danmarks Adressers Web API (DAWA) udstiller data og funktionalitet vedrørende Danmarks...
HTML

Daginstitutioner
Sønderborg Kommune
Datasættet viser offentlige børnehaver og børneuniverser samt de private og tyske børnehaver. Se også data på...
SHP

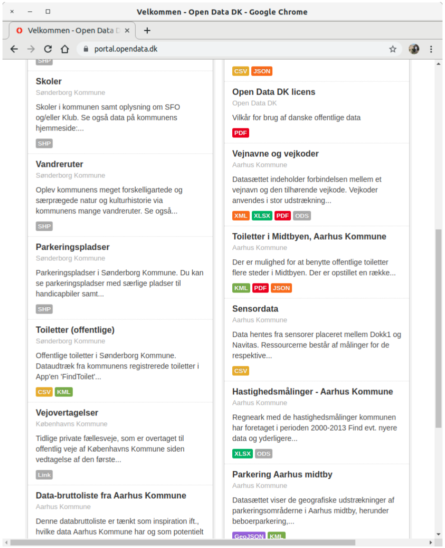
Skoler
Sønderborg Kommune
Skoler i kommunen samt oplysning om SFO og/eller Klub. Se også data på kommunens hjemmeside:...
SHP

Populære

3D bymodel Aarhus Kommune - Bygninger
Aarhus Kommune
Bygninger på DWG-format for Aarhus kommunes 3D bymodel. Modellen ligger i 7 filer. Inddelingen fremgår af vedhæftede...
JPEG dwg

Realtids trafikdata
Aarhus Kommune
Datasæt med metadata for trafikmåling i Aarhus og live/realtidsmålinger af trafikken i Aarhus Kommune....
CSV JSON

Open Data DK licens
Open Data DK
Vilkår for brug af danske offentlige data
PDF



Velkommen - Open Data DK - Google Chrome

Velkommen - Open Data D

portal.opendata.dk

SHP

Skoler
Sønderborg Kommune
Skoler i kommunen samt oplysning om SFO og/eller Klub. Se også data på kommunens hjemmeside:...
SHP

Vandreruter
Sønderborg Kommune
Oplev kommunens meget forskelligartede og særprægede natur og kulturhistorie via kommunens mange vandreruter. Se også...
SHP

Parkeringspladser
Sønderborg Kommune
Parkeringspladser i Sønderborg Kommune. Du kan se parkeringspladser med særlige pladser til handicapbiler samt...
SHP

Toiletter (offentlige)
Sønderborg Kommune
Offentlige toiletter i Sønderborg Kommune. Dataudtræk fra kommunens registrerede toiletter i App'en 'FindToilet'...
CSV KML

Vejovertagelser
Københavns Kommune
Tidlige private fællesveje, som er overtaget til offentlig veje af Københavns Kommune siden vedtagelse af den første...
Link

Data-bruttoliste fra Aarhus Kommune
Aarhus Kommune
Denne databruttoliste er tænkt som inspiration ift., hvilke data Aarhus Kommune har og som potentielt

CSV JSON

Open Data DK licens
Open Data DK
Vilkår for brug af danske offentlige data
PDF

Vejnavne og vejkoder
Aarhus Kommune
Datasættet indeholder forbindelsen mellem et vejnavn og den tilhørende vejkode. Vejkoder anvendes i stor udstrækning...
XML XLSX PDF ODS

Toiletter i Midtbyen, Aarhus Kommune
Aarhus Kommune
Der er mulighed for at benytte offentlige toiletter flere steder i Midtbyen. Der er opstillet en række...
KML PDF JSON

Sensordata
Aarhus Kommune
Data hentes fra sensorer placeret mellem Dokk1 og Navitas. Ressourcerne består af målinger for de respektive...
CSV

Hastighedsmålinger - Aarhus Kommune
Aarhus Kommune
Regneark med de hastighedsmålinger kommunen har foretaget i perioden 2000-2013 Find evt. nyere data og yderligere...
XLSX ODS

Parkering Aarhus midtby
Aarhus Kommune
Datasættet viser de geografiske udstrækninger af parkeringsområderne i Aarhus midtby, herunder beboerparkering,...
GeoJSON KML

Data Formats

HTML, JSON, XML, CVS, XHTML, TSV, TXT, DOC, MOV, TIFF, DOCX, PDF, XLS, XLSX, PDF, TEX, PNG, GIF, MPG, SQL, JPG, RTF, MARKDOWN, . . .

Data Formats in Rough Categories

Tabular Data
sql, csv, tsv, (xls, etc.), . . .

Parentheses Structures
xml, json, markdown, (html), . . .

Others – Data Extraction Problematic
mpg, jpg, pdf, doc, . . .

Tabular Data Formats

These are data formats implementing a list of records (tuples, rows) such as

Animal	Cuteness
Giant Panda	1.0
Sea Otter	0.95
Meerkat	0.9
Rabbit	0.8
Red Panda	0.8
Leopard	0.7
Clown Fish	0.4
Python	0.1
Rat	0.07
Tarantula	0.00001

Tabular Data Formats

Tabular data can be encoded as text files using a designated separator character between fields and newline between two records.

`tsv` – tab-separated values – is one such standard, where **`\t`** (tab) is the separator character.

`csv` – comma-separated values – is another standard, where the separator character is a comma. Sometimes this term is used broadly for the general idea, and one can specify the separator character (to be tab, for instance).

These formats may come with a header record, i.e., a first line specifying the names of the different fields.

`sql` tables, **`xlsx`** documents, or similar may contain tabular data, but often with more complex additional information. However, the tabular information itself can often be exported to a **`csv`** file, for instance.

Thus, tabular information can often be processed by line-based tools.

CSV-like formats

No absolute standard, but special characters such as comma (if that is the separator) and newline must be quoted or escaped.

Likely “rules” to check data for:

- Header record, possibly optional.
- Same number of comma-separated fields in each record.
- What should be escaped and how? Probably double quotes. Thus, fields with commas and newlines should be quoted and a quote should be doubled.

Be aware regarding the following:

- Spaces are probably part of a field.
- Is an empty line white-space or an empty record?

CSV-like formats: example

The one record

Animal	Cuteness
Giant, "The Cutie", Panda	1.0

should likely be represented as

```
"Giant, ""The Cutie"", Panda",1.0
```

CSV-like formats: Resources

https://frictionlessdata.io/specs/tabular-data-package/

https://frictionlessdata.io/specs/csv-dialect/

https://docs.python.org/3/library/csv.html

CSV-like formats: Python CSV Module

```
> cat example.csv
Giant Panda,1.0
Sea Otter,0.95
Meerkat,0.9
Rabbit,0.8
Red Panda,0.8
Clown Fish,0.4
Python,0.1
Tarantula,0.00001
>
```

CSV-like formats: Python CSV Module

```
> cat readWriteCSV.py
import csv
exampleFile = open('example.csv')
exampleReader = csv.reader(exampleFile)
exampleData = list(exampleReader)
exampleFile.close()
print(exampleData)
outputFile = open('output.csv', 'w')
outputWriter = csv.writer(outputFile)
for record in exampleData:
    outputWriter.writerow(record)
outputWriter.writerow(['Leopard', '0.7'])
outputFile.close()
> python readWriteCSV.py
```

CSV-like formats: Python CSV Module

prints

```
[['Giant Panda', '1.0'], ['Sea Otter', '0.95'], ['Meerkat', '0.9'], ['Rabbit',
'0.8'], ['Red Panda', '0.8'], ['Clown Fish', '0.4'], ['Python', '0.1'], ['Tara
ntula', '0.00001']]
```

and output.csv contains

```
> cat output.csv
Giant Panda,1.0
Sea Otter,0.95
Meerkat,0.9
Rabbit,0.8
Red Panda,0.8
Clown Fish,0.4
Python,0.1
Tarantula,0.00001
Leopard,0.7
>
```

CSV-like formats: Python CSV Module

Can specify various things such as

- **`delimiter`**
- **`lineterminator`**
- ...

CSV-like formats

One can convert back and forth between CSV formats and many other formats. Most spreadsheets and database management systems support the format.

Some editors support the format such that one can get a better editing experience, e.g., getting a column-based layout.

Ex: emacs has modes for operating on TSV or CSV files.

Data Transformation

Recall the natural steps in a data transformation process:

- data discovery
- data mapping
- code generation
- code execution
- data review

Parts of the process are repeated if the data review is not completely successful.

Command-Line Tools

It is what developers use. . .

“Native” in Linux and iOS.

For this, Microsoft provides Windows Subsystem for Linux.

You saw a very brief introduction to command-line tools by Jakob first year.

Command-Line Tools for Data Discovery

Pitfalls

Character encoding: ascii, UTF-8, ISO-8859, . . .

Line separator: Unix style newline (LF) or MS-DOS-style (CR/LF).
LF is `\n` (ascii 10), CR is `\r` (ascii 13).

Tools

`wc` – print newline, word, and byte counts for each file argument.

`file` – determine file type of file argument.

`recode` – convert between character sets, e.g., `recode l1..u8`.

`od` – octal dump, i.e., actually see the bytes, e.g.,
`od -tcuC` – show byte value in decimal and ascii character if printable.

Command-Line Tools for Code Generation

Example tools include

- grep
- sed
- gawk (GNU awk)
- sort
- uniq
- tr
- cut
- paste
- join

Many use regular expressions, as do editors, programming languages, etc.

Regular Expressions in Practice

Major differences between regular expressions in practice and regular expressions from formal languages textbooks:

- Alphabets are large (in the hundreds); not just $\{0, 1\}$ or $\{a, b\}$.
- Some symbols in the alphabet are not printable characters.
- The operators of regular expressions are characters, and they are *also* in the alphabet.

These issues create problems that we discuss now.

Regular Expressions in Practice

We introduce short-hands such as (examples depend on the tool)

- `.` matches any one character different from `\n`.
- `[a-z]` matches any one character in the given range.
- `^` matches the empty string, but only at the beginning of a line.

We "make" the most popular non-printable characters representable, such as `\n`, `\t`, `\r`, ...

We *escape* either the operators or the characters with the same representation as the operators, e.g.,

- We have seen examples where union (or) is written `\|`.
- We have seen examples where the parentheses in `(.*)` *groups* the regular expression `.*`, so then the parenthesis character must be represented by `\(`. And of course backslash must be backslashed!

The choice of what to escape is tool-dependent.

Command-Tools

The tools can do more than we show; sometimes *much* more.

Basic and often sufficient information can be found via the man-pages, e.g., **`man grep`**.

Many of the tools have online manuals or tutorials available and books can be purchased.

Command-Tool: grep – comments to slides

We will use the lecture notes from New York University:

`https://cs.nyu.edu/~mohri/unix08/lect4.pdf`

Comments to those slides

egrep, **fgrep**, etc. are deprecated; use **grep -E**, **grep -F**, etc.

The word file mentioned in the slides is great for testing. On IMADA's system, the file is not located at **/usr/dict/words**, but at **/usr/share/dict/words**.

As an example, words starting and ending with a "k" can be found by

```
grep "^k.*k$" /usr/share/dict/words
```

Command-Tool: grep – comments to slides

The primary difference between **grep** and **grep -E** is whether (certain) operator symbols are treated as symbols in the alphabet or as operator symbols; use in the other context must then be escaped.

If we want to find lines where parentheses are used at least twice, as in the line

```
I believe (I'm almost certain) that my editor (emacs) is the best!
```

this can be obtained with **grep** as

```
grep "\((.*).*\)\{2,\}"
```

or with **grep -E** (the old **egrep**) as

```
grep -E "(\(.*\).*){2,}"
```

Command-Tools: sed and awk

We will use the lecture notes from New York University:

`https://cs.nyu.edu/~mohri/unix08/lect5.pdf`