Status Quo efter lektion 2

I har alle kørt ca. disse kommandoer, med kort forklaring:

- Skift til corpus mappen

```
cd ~/lt-course/lecture01/corpus
```

- Lav fuldformsfrekvens separat for alle 3 givne corpora

```
cat Ataqqinartuaraq.txt | tr '., !"”:?-' '\n' | sort | uniq -c | sort -n -r | more
```

```
cat Aviscorpus.txt | tr '., !"”:?-' '\n' | sort | uniq -c | sort -n -r | more
```

```
cat UkiutTrettenit.txt | tr '., !"”:?-' '\n' | sort | uniq -c | sort -n -r | more
```

- Lav fuldformsfrekvens samlet for alle givne corpora

```
cat * | tr '., !"”:?-' '\n' | sort | uniq -c | sort -n -r | more
```

- Opret mappen work i jeres home-mappe til at arbejde videre i

```
cd
mkdir work
```

- Kopier de 3 corpora til mappen work

```
cd ~/lt-course/lecture01/corpus
cp * ~/work
```

- Gå til mappen work og check at corpora nu eksisterer her

```
cd ~/work
ls
```

- Lav fuldformsfrekvens separat for alle 3 givne corpora, men denne gang skriv resultatet til en fil for hvert corpus

```
cat Ataqqinartuaraq.txt | tr '., !"”:?-' '\n' | sort | uniq -c | sort -n -r > Ataqqinartuaraq-frekvens.txt
```

```
cat Aviscorpus.txt | tr '., !"”:?-' '\n' | sort | uniq -c | sort -n -r > Aviscorpus-frekvens.txt
```

```
cat UkiutTrettenit.txt | tr '., !"”:?-' '\n' | sort | uniq -c | sort -n -r > UkiutTrettenit-frekvens.txt
```

- … og check at de blev lavet

```
ls
```

Herefter åbnede i de 3 nye -frekvens.txt filer i en grafisk text editor for bedre at kunne sammenligne dem.

Så downloadede i corpuset https://uni.oqaasileriffik.gl/Facebook.txt uden for terminalen, og lagde det ind i

work-mappen så i kunne arbejde med det fra terminalen.

- Lav fuldformsfrekvens af det nye Facebook-corpus, og skriv til en fil

```
cat Facebook.txt | tr '., !"”:?-' '\n' | sort | uniq -c | sort -n -r > Facebook-frekvens.txt
```

Og så åbnede i den nye Facebook-frekvens.txt i text editoren ved siden af de andre fra før.

- Vis de 10 første linier af en fil, hvilket her betyder de 10 mest frekvente ord fordi vi arbejder med frekvenslister

```
cat Ataqqinartuaraq-frekvens.txt | head
```

- … eller de 50 først linier

```
cat Ataqqinartuaraq-frekvens.txt | head -n 50
```

- … eller de 10 sidste linier

```
cat Ataqqinartuaraq-frekvens.txt | tail
```

- … eller de 50 sidste linier

```
cat Ataqqinartuaraq-frekvens.txt | tail -n 50
```

- Vis hvor mange unikke ord der er i listen

```
cat Ataqqinartuaraq-frekvens.txt | wc
```

- … og filtrer listen så vi kun ser ord med 1 forekomst

```
cat Ataqqinartuaraq-frekvens.txt | grep ' 1 ' | more
```

- … og tæl dem

```
cat Ataqqinartuaraq-frekvens.txt | grep ' 1 ' | wc
```

- I stedet, filtrer listen så vi kun ser ord med mere end 1 forekomst

```
cat Ataqqinartuaraq-frekvens.txt | grep -v ' 1 ' | more
```

- … og tæl dem

```
cat Ataqqinartuaraq-frekvens.txt | grep -v ' 1 ' | wc
```

- Filtrer listen så vi kun ser ord der slutter på poq

```
cat Ataqqinartuaraq-frekvens.txt | egrep 'poq$' | more
```

- … eller ord der slutter på voq

```
cat Ataqqinartuaraq-frekvens.txt | egrep 'voq$' | more
```

- … eller ord der starter med aki

```
cat Ataqqinartuaraq-frekvens.txt | grep ' aki' | more
```

Noter & Hints

Terminologi: Terminal, shell, command line, cmdline, cmd, konsol, console, kommandoprompt, command prompt - alle udtryk for det samme begreb, nemlig et tekstbaseret vindue hvor man skriver kommandoer.

Som filnavn er `*` et wildcard pattern (jokertegn, mønster) der matcher alt. For sig selv betyder `*` alle filer og mapper i den mappe man står i. Man kan skrive den som del at en navn for at indskrænke matchet, e.g. `*-frekvens.txt` matcher alle filer (og mapper) der slutter med -frekvens.txt. Der kan være flere `*` i et pattern, e.g. `*word*` matcher alle filer (og mapper) der har word et eller andet sted i navnet. Se også https://en.wikipedia.org/wiki/Wildcard_character

Fordi wildcards også kan matche mapper, er det vigtigt at navngive sine filer på en måde hvor man kan se forskel. Det er gyldigt at have en text-fil med navnet Ataqqinartuaraq uden .txt på, men så er det svært at fange den med et wildcard pattern. Hvis man altid sætter et beskrivende suffix på sine filer, er det nemmere at genkende dem igen.

Afsnittet om `~` fra Note 01 er relevant igen.

I en terminal betyder `>` at herfra skal der skrives til en fil, og navnet på den fil kommer lige efter `>` tegnet.

Terminaler har det svært med space i argumenter til kommandoer, hvilket inkluderer fil- og mappe-navne. Man kan godt have sådanne filer, men så skal man tage hensyn. E.g., hvis man har en fil Ataqqinartuaraq Frekvens.txt med space i navnet og vil læse den med `cat`, så skal man skrive enten `cat Ataqqinartuaraq\ Frekvens.txt` eller `cat 'Ataqqinartuaraq Frekvens.txt'` eller `cat "Ataqqinartuaraq Frekvens.txt"` - altså enten escape spacet med `\ ` eller sætte hele filnavnet i single-`'` eller dobbelt-`"` quotes. Så det er meget nemmere at sørge for at filerne bruger InterCaps eller dash `-` eller underscore `_` i stedet for spaces.

Terminologi: Hyphen, dash, minus, streg betyder alle tegnet `-`, altså den helt normale bindestreg.

Forskellen på `grep` og `egrep`, er at `egrep` håndterer mere komplekse Regular Expression patterns. I dagens eksempler gør det faktisk ikke forskel om man bruger grep eller egrep, fordi `$` håndteres ens af begge. Men der vil komme patterns hvor det gør en forskel.

Mht. `$`, så i en Regular Expression (regex) betyder det slutningen af linien - det kommer vi mere ind på senere. Regex kan tænkes på som komplekse wildcards.

Men der er en vigtig detalje med kontekst af symboler. I en terminal betyder `$` en variabel, og hvis man bare skriver `$` i et argument uden single-`'` quotes omkring, så kan den bliver erstattet med noget fra terminalen. Single-`'` quotes beskytter argumenter fra sådanne bivirkninger - det gør dobbelt-`"` quotes ikke. I kan afprøve dette ved at køre `echo $PATH` uden quotes og sammenligne med `echo '$PATH'` med single-quotes og med `echo "$PATH"` med dobbelt-quotes.

Det samme gælder for `|` og `>` og mange andre symboler. Hvis de skal gives som argument til et program, så skal de stå i `'` quotes.

I eksemplerne benytter jeg kun `'` når det kan være nødvendigt. E.g. `head -n 50` ville være det samme som `head -n '50'` - men ingen del af argumentet `50` kan misforstås af terminalen, så jeg bruger ikke `'`. Hvis man er i tvivl, så skader det ikke at bruge `'` om alle argumenter.