Status Quo efter lektion 5

Samle alting på et sted

Første miniopgave var at kopiere alt fra lt-course's undermapper's corpus- og scripts-mapper til `~/work`. Dette kan gøres på flere måder.

- Først, sørg for at lt-course er den nyeste version
```
cd ~/lt-course
git pull
```

- Og, for dem der ikke havde en ~/work
```
cd
mkdir work
```

- Så cd til og kopier fra lecture01's corpus og scripts-mapper
```
cd
cd lt-course
cd lecture01
cd corpus
cp -v * ~/work/
cd ..
cd scripts
cp -v * ~/work/
```

...og så tilsvarende med lecture02's corpus-mappe og lecture03's corpus- og scripts-mapper. Her `cp -v` for at vise hvad den gør, fordi `cp` ellers ikke siger noget.

- Eller, man kan bruge wildcards til at gøre det i ét hug:
```
cd ~/work
cp -v ~/lt-course/lecture*/*/* ./
```

Når alle `*` bliver foldet ud så matches alt vi vil have. Første led `~/lt-course/lecture*/` matcher alle lecture01, lecture02, og lecture03-mapperne. Herfra beder vi om alle deres undermapper med `*/`, som matcher corpus og scripts. Og så til sidst alle filer og mapper deri med `*`. Når man kender strukturen på en mappe, så kan man skrive sådanne præcise wildcards.

Opbrydning i sætninger

Så kiggede vi på måder at bryde en text op i sætninger. Først bliver vi nødt til at definere hvad en sætning er. Dette er analogt til problemet fra lektion 1 med hvad et ord er.

- Vi kan sige at en sætning er afgrænset af punktum, spørgsmåltegn, eller kolon, og så benytte `tr` fra før

```
cat Ataqqinartuaraq.txt | tr ':?.' '\n' | more
```

Men, når man læser sætninger er det vigtigt at vide om den faktisk slutter på punktum eller spørgsmåltegn. `tr` sletter tegnene, så den kan ikke bruges. Der kan i stedet bruges regular expressions (regex) via `sed -E` eller `perl -pe`. Desværre er macOS's `sed` ikke i stand til at håndtere \n særlig godt, men det kan `perl`.

- Bryd sætninger, men behold tegnene

```
cat Ataqqinartuaraq.txt | perl -pe 's/([.?:])/\1\n/g' | more
```

Regex'en `s/([.?:])/\1\n/g` betyder:

- `s///` er en substitute regex, altså search-replace
- `[.?:]` matcher en af tegnene punktum, spørgsmålten, eller kolon.
- `([.?:])` fanger (capture group) dette match i en variabel
- `s/([.?:])/\1\n/` erstatter det der står før midterste `/` med det der står efter
- `\1\n` betyder den første capture group og et line break

Altså, erstat punktum eller spørgsmåltegn eller kolon, med sig selv plus et linieskift. Og så lige flag `g` (global) som betyder at dette skal gøres så mange gange det er muligt på linien. Uden `g` udføres search-replace kun 1 gang per linie.

- Dette kan så bruges til e.g. at lave statistik på sætningslængder i et corpus.

```
cat Ataqqinartuaraq.txt | perl -pe 's/([.?:])/\1\n/g' | sort | uniq | awk '{print length, $0}' | sort -nr | more
```

Finde forkortelser

Derefter var opgaven at finde forkortelser. Igen, første problem er at definere hvad en forkortelse er - hvordan ser den ud i en tekst?

- En forkortelse slutter på punktum?

```
cat Aviscorpus.txt | egrep -o '\S+\.' | more
```

Ikke præcist nok. Alle ord i slutningen af en sætning slutter på punktum.

- En forkortelse slutter på punktum, og er efterfulgt af et lille bogstav?

```
cat Aviscorpus.txt | egrep -o '\S+\. [a-z]+' | more
```

Bedre, men mangler stadig nogen.

- Fornavne er store bogstaver efterfulgt af et ord

```
cat Aviscorpus.txt | egrep -o '[A-Z]+\. \S+' | more
```

...og så videre. Man kan lave mange regler med mange kontekster. Heldigvis er langt de fleste regler allerede i

analysatoren, men til grovsortering er det altid godt at kunne noget grep.

Mht. `egrep`, så betyder `-o` at i stedet for at outputte hele linien der matcher, så skal den kun outputte lige nøjagtigt den del af linien der matcher.

Finde proprier

Tilsvarende, så skulle vi finde alle proprier.

- Ord der starter med stort er gode kandidater til proprier?

```
cat Aviscorpus.txt | egrep -o '[A-Z]\S+' | more
```

Men, det giver også alle ord i starten af sætningen.

- Ord der starter med stort, men står efter et lille bogstav eller tal

```
cat Aviscorpus.txt | egrep -o '[a-z0-9] [A-Z]\S+' | more
```

Det giver langt bedre resultat, men outputtet har det sidste bogstav af ordet før propriet, og det kan vi ikke bruge til så meget.

- ...samme som før, men print kun det der står i 2. kolonne, hvilket i dette tilfælde er propriumskandidaten

```
cat Aviscorpus.txt | egrep -o '[a-z0-9] [A-Z]\S+' | awk '{print $2}' | more
```

`awk` opfatter kolonner som alt der ikke er space, og `$2` er kolonne nummer 2. Den specielle variabel `$0` er hele linien. I behøver nok ikke lære mere `awk` end hvad der står på disse sider - det er et helt programmeringssprog, men for sprogteknologi bruges det ikke til meget andet end disse småting.

Finde kontekst af ord

Så har vi nogle proprier eller andre interessante ord, og vi vil gerne vide i hvilken kontekst de bliver brugt.

- E.g., find konteksen for Kalaallit

```
cat Aviscorpus.txt | egrep -o 'Kalaallit \S+' | more
```

- ...og få deres frekvens

```
cat Aviscorpus.txt | egrep -o 'Kalaallit \S+' | sort | uniq -c | sort -nr | more
```

- Eller hvis man gerne vil have hele konkordancer af ord

```
cat Aviscorpus.txt | egrep -o '(\S+ ){0,5}Kalaallit( \S+){0,5}' | more
```

Her findes ordet Kalaallit og op til 5 ord før og efter. Hvis der er færre end 5 ord i en retning, så er det ok.

Se forskellene på uddelte corpora vs. analyserede

De uddelte filer med navne som DisambigueretOgLettereJusteret.txt er alle en slags guld-corpora. De er

analyserede corpora som et eller flere mennesker (i dette tilfælde Per Langgård) har gået gennem og rettet hvor det var nødvendigt. Det er ikke alt der er rettet i disse filer, så de er ikke helt guld.

Analysatoren er ikke perfekt - den er meget god, men sprog er noget bøvlet noget. Så for at se forskellene på hvad der kommer ud af analysatoren og hvad Per har ment skulle rettes kan vi bruge `diff`. Men først skal vi have et corpus analyseret med systemet som det er.

- Kør et corpus gennem analysatoren

```
   cat Ataqqinartuaraq.txt | ./kal-analyse > Ataqqinartuaraq-analyzed.txt
```

- Se forskellene på nuværende analyse og det som Per har kigget på

```
   diff -u5 Ataqqinartuaraq-analyzed.txt
AtaqqinartuaraqDisambigueretOgLettereJusteret.txt | more
```

`diff -u` giver output i *unified diff format*, hvor ændringer vises med - og + først på linien. Hvis man ikke giver et tal så vises 3 linier før og efter ændringerne, hvilket som regel ikke er nok til analyser.

Desværre har Per kørt en ældre version hvor dobbelte DIRTALE-tags ikke er fjernet, så diff giver alt for meget. Det skal ryddes op.

- Fjern DIRTALE-tags hvis der står to eller flere identiske af dem

```
   cat AtaqqinartuaraqDisambigueretOgLettereJusteret.txt | perl -pe
's/(DIRTALE\S+)( \1)+/\1/g' >
AtaqqinartuaraqDisambigueretOgLettereJusteret-clean.txt
```

- Sammenlign analysen med den rensede fil

```
   diff -u5 Ataqqinartuaraq-analyzed.txt
AtaqqinartuaraqDisambigueretOgLettereJusteret-clean.txt | more
```

En anden systematisk forskel er at Per ikke har kørt analysen med hybrid-opsplitning, så ord som e.g. `pisoqarfiusumiittumut` har i analysen fået splittet sin `ittumut` af til et separat token, men dette er ikke sket i Per's corpus. Det kan vi ikke så nemt rette på, så det må man bare ignorere.

Om den nye analyse eller om Per's rettelser er bedre kan jeg ikke sige noget om - jeg kan ikke grønlandsk endnu.