INF1820, V2017 – Oblig 1b
Regulære uttrykk og endelige tilstandsmaskiner

Innleveringsfrist: 17 februar

I denne oppgaven skal vi jobbe videre med tekst i Python. Vi skal nå se nærmere på tokeniseringsprogrammet du skrev i forrige oblig og forbedre det ved bruk av regulære uttrykk.

Innlevering av oppgaven skjer i Devilry. Se emnesiden for mer informasjon om reglement rundt innlevering, samt bruk av Devilry. Registrer svarene dine i en fil som angir brukernavnet ditt slik:

`oblig1_brukernavn.py`

En perfekt løsning av denne oppgaven er verdt 100 poeng. Koden må kunne kjøres på IFIs maskiner og må inneholde kommentarer som forklarer hva koden gjør.

1 Tokenisering av norsk tekst (50 poeng)

a) **Tokenisering fra oblig1a** Ta utgangspunkt i tokeniseringsoppgaven fra oblig1a (enten din egen kode eller løsningsforslaget). Les inn filen `dev.txt` og skriv ut resultatet (ett ord per linje) til en fil.

b) **Inspisér resultatet**

- Filen `dev_gold.txt` (fra emnesiden) inneholder korrekt tokenisering av teksten i `dev.txt` på samme format (ett ord per linje). Bruk linux-kommandoen `diff` for å sammenligne de to filene og skriv resultatet til en fil:

  ```
  diff <minfil> dev_gold.txt >diff-fil
  ```

  Dette gir deg en liste med feil i tokeniseringen din.
- Beskriv feilene du finner og gi minst tre eksempler på forskjellige typer feil.

c) **Tokenisering med regulære uttrykk** Vi skal nå jobbe med å forbedre tokeniseringskoden vår, basert på feilanalysen fra forrige oppgave. Et minimumskrav til denne oppgaven er at programmet ditt har rettet opp i feilene du identifiserte i forrige oppgave.

- Skriv et regulært uttrykk som definerer et gyldig norsk ord. Her bør du bruke disjunksjon for å beskrive forskjellige typer ord (f.eks. ulike typer tegnsetting, bokstavsekvenser, osv.).
- Benytt deg av Python's `re.findall` for å hente ut ordene som matcher uttrykket ditt fra linjene med tekst. Denne metoden tar et regulært uttrykk og en streng og returnerer en liste med alle treff. F.eks. vil `re.findall("([0-9])","inf1820")` gi ut listen `['1', '8', '2', '0']`.
- Bruk `diff` fortløpende for å evaluere tokeniseringen din.

d) **Feilanalyse** Hvilke feil gjenstår? Gi minst to eksempler på feil som gjenstår og gi forslag til hvordan disse kan rettes opp.

2 Endelig tilstandsmaskiner (50 poeng)

I denne oppgaven skal vi bruke ekstern og fritt tilgjengelig programvare for å tegne FSA'er, nemlig JFLAP. Dette må dere laste ned selv fra `http://www.cs.duke.edu/csed/jflap/jflaptmp/`

- Last ned den nyeste versjonen som en .jar-fil. Du kan f.eks. bruke `wget`-kommandoen for å hente ned filen fra et terminalvindu slik:

  ```
  wget <URL>
  ```

- Kjør programmet fra kommandolinjen slik:

  ```
  java -jar JFLAP.jar
  ```

Eksperimenter litt med programmet slik at du forstår hvordan det fungerer og håndterer følgende:

- hvordan legger man til tilstander?
- hvordan markerer man en tilstand som start-og sluttilstand?
- hvordan legger man inn kanter fra en tilstand til en annen?
- hvordan tester man maskinen på input?

Dersom du ikke får til dette, bør du se på en tutorial som ligger på JFLAP's nettside (`http://www.cs.duke.edu/csed/jflap/tutorial/`).

Tegn en endelig tilstandsmaskin i JFLAP som gjenkjenner følgende språk, der alfabetet er $\{a, b\}$. Alle tilstandsmaskinene skal være deterministiske.

(a) $\{\ w \mid w$ inneholder minst tre b'er $\}$ (f.eks. skal $abbab$ og $ababaababbb$ aksepteres, men ikke $aaabaaa$ og ab).

(b) $\{\ w \mid$ hver oddetallsposisjon i w er en $b\ \}$ (Maskinen kan godt akseptere den tomme strengen).

(c) $\{\ w \mid w$ inneholder ikke substrengen $bba\ \}$ (Maskinen kan godt akseptere den tomme strengen).

(d) $\{\ w \mid w$ inneholder et partall antall a eller odde antall $b\ \}$

Lagre de ferdige maskinene som .gif-filer (File $\rightarrow$ Save Image As), og lever dem sammen med koden din, som separate filer med navnene `2a.gif, 2b.gif, 2c.gif` og `2d.gif`.