INF4490 Mandatory Assignment 2: Multilayer Perceptron

Stein Raymond Rudshagen

February 7, 2017

$\mathbb{P}$ marks the programming exercises, using "Program language" to run the programs

Oppgave 1: Formelle språk (2 poeng)

1. La L = ab, abbb, abc, c og M = bba, a. Hva blir LM?
2. Hva blir ML?
3. Betrakt språket M*L. Hvilke av følgende uttrykk er i dette språket: a, aa, ab, ac, acc, aacc, aaabbb, ε?
4. Hvilke av disse uttrykkene er i språket (ML)* ?
5. La N=Ø og P={ε}. Hva blir LN og hva blir LP?

Answer:

1. LM blir LM = abba, aba, abbbbba, abbba, abcbba, abca, cbba, ca
2. LM = ML (med andre ord de er like)
3. M*L = ac, ε
4. (ML)* = ac, ε
5. N=Ø (tom mengde) $\rightarrow LN =$, $P = \{\varepsilon\}$ (tom string) $\rightarrow LP = L$

Oppgave 2: Endelige tilstandsmaskiner (4 poeng)

Denne oppgaven kan gjøres i JFLAP. Du anbefales likevel å løse den med papir og penn først for å få eksamenstrening. Så kan du bruke JFLAP til å kontrollere løsningen din.

1. Lag en ikke - deterministisk endelig tilstandsmaskin (NFA) som beskriver språket L1=L(a*b(a+c)* + ac(b+a)), der alfabetet er A=a, b, c. (Symbolet + er her disjunksjon)
2. Lag en deterministisk maskin (DFA) som beskriver det samme språket.
3. Lag en tilstandsmaskin som beskriver komplementspråket til L1.
4. Hvilke av følgende uttrykk er i L1?

Answer:

1. Tegnet det på ark først også tegnet den inn i jflap

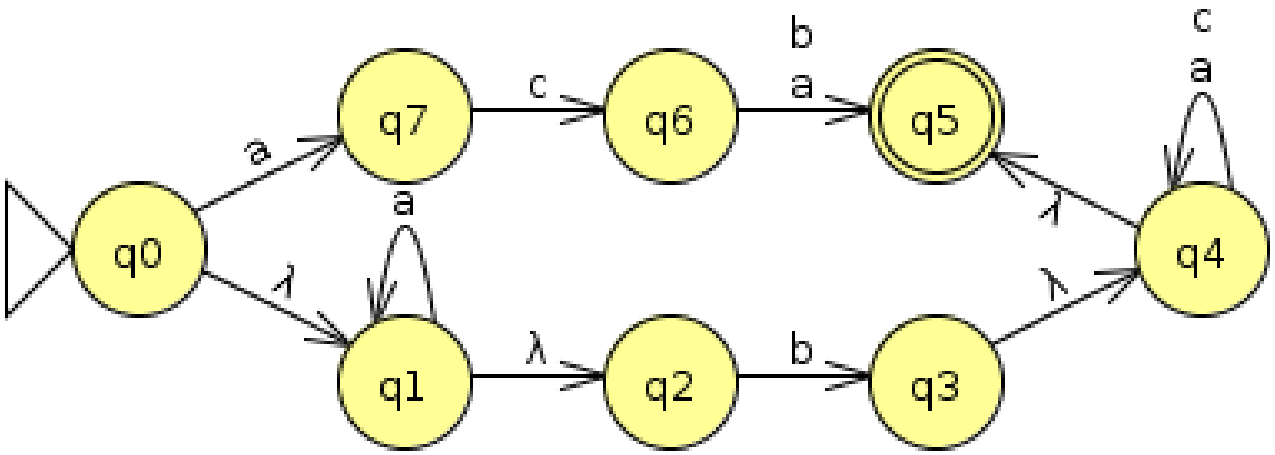

Figure 0.1: Printer ut de regulære uttrykk som tester fire forskjellige strenger hver.

2. Lagde først tabell for NFA til DFA, resultatet er på bildet under tabellen.

i	Q	a	b	c
0	[0,1,2]	[6,1,2]	[3,4,5]	-
1	[6,1,2]	[1,2]	[3,2,5]	[7]
2	[3,4,5]	[4,5]	-	[4,5]
3	[1,2]	[1,2]	[3,4,5]	-
4	[7]	[5]	[5]	-
5	[4,5]	[4,5]	-	[4,5]
6	[5]	-	-	-

Table 1: Tilstandstabell for NFA til DFA

3. Det er den samme tilstandsmaskinen som vist ovenfor.
4. L1 = 'ac(a|b)','ab','aab','aaa*b','b','b(a|c)*'

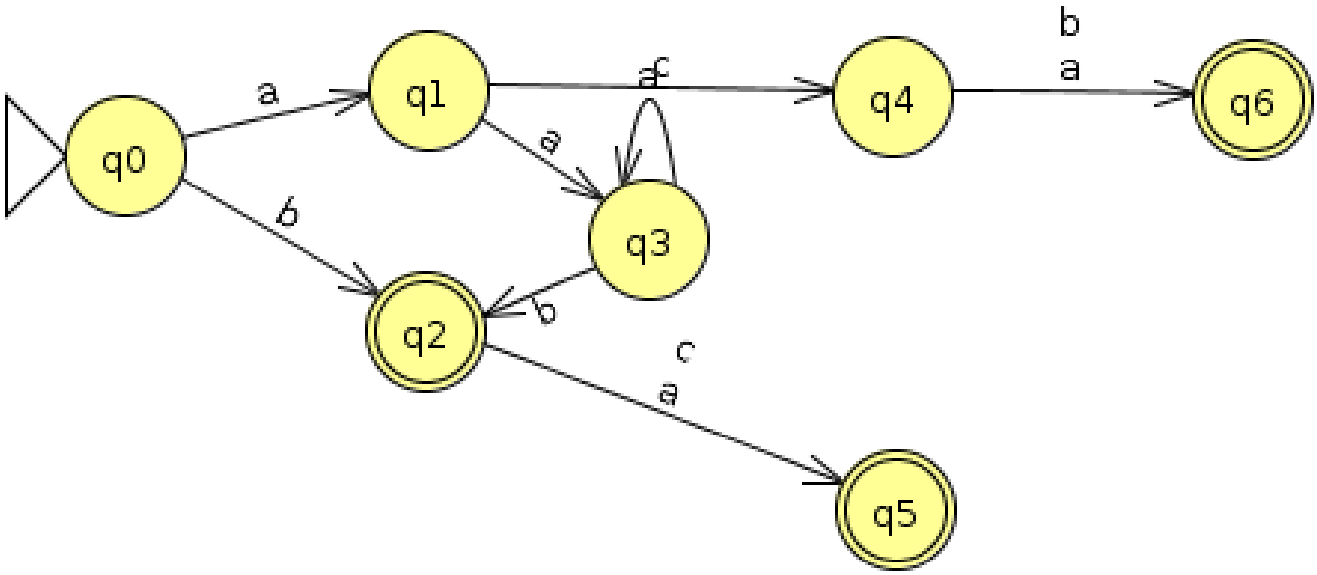

Figure 0.2: Printer ut de regulære uttrykk som tester fire forskjellige strenger hver.

Oppgave 3: Regulære uttrykk (4 poeng)

Vi skal her bruke hva vi vil kalle "rene" regulære uttrykk. Det er de vi gikk gjennom på forelesningen bygget opp ved

Regulære uttrykk	Beskriver språket
Ø	L(Ø) = Ø
ε	L(ε) = {ε}
a, for alle a ε A	L(a) = {a}
Hvis R og S er regulære uttrykk:	
(R+S)	L(R+S)=L(R)∪L(S)
(R T)	L(R T) = L(R)L(T)
(R*)	L(R*) = L(R)*

Table 2: Caption

La alfabetet A = {a, b, c}. Lag regulære uttrykk for følgende språk

1. Ord som inneholder minst tre b-er på rad.
2. Ord som ikke inneholder mer enn to b-er på rad.
3. Ord hvor antall b-er er delelig med 3 eller antall a-er er delelig med 2 (eller begge deler)

Answer:

1. (a+b+c)*bbb(a+b+c)*
2. (a+c)*((ba+bba+bc+bbc)(a+c)*)*(b+bb)
3. (((a+c)*b(a+c)*b(a+c)*b(a+c)*)*+((b+c)*a(b+c)*a(b+c)*)*)

Oppgave 4: Regulære uttrykk i Python (2 poeng)

Test løsningene dine fra oppgave 3 i Python. Dvs. for hver av de tre oppgavene, Kan jeg gjøre denne uten tekstfilen ?

1. Skriv det regulære uttrykket som et Python-regulært uttrykk!
2. Lag to eksempler på strenger som skal være i språket og to som ikkje skal være det!
3. Test uttrykket ditt for de fire strengene!

Answer:

Oppgave 5: Innlesning av tekst (4 poeng)

1. Les den inn i en interaktiv Python-sesjon som en streng. Kall den "pyt_raw". Oppskriften finner du i seksjon 3.1 i NLTK-boka, underavsnitt "Reading local files". Hvis du ser noe rusk i teksten, så rens den.

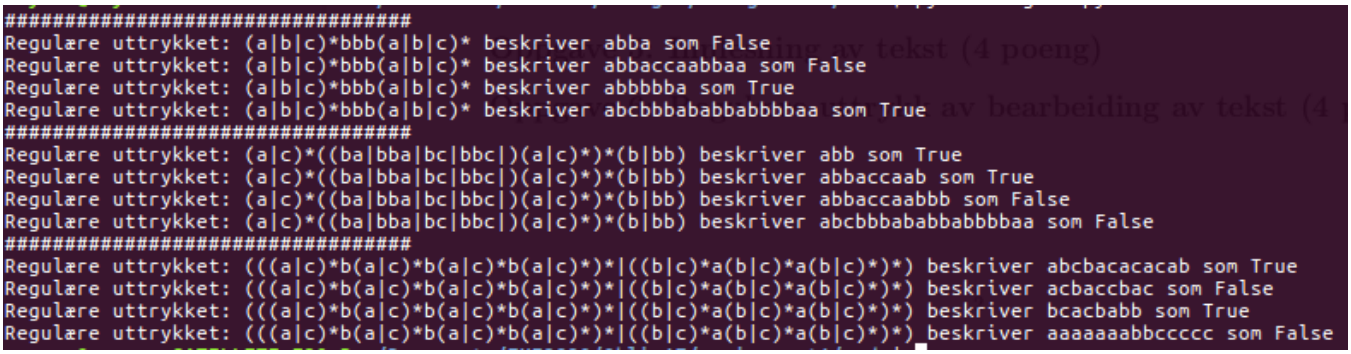

Figure 0.3: Printer ut de regulære uttrykk som tester fire forskjellige strenger hver.

2. Når vi videre skal arbeide med en tekst, kan det være en fordel å dele den opp i en liste av ord, der hvert ord er en streng. Den enkleste måten å gjøre dette på er ved å bruke split i python. »> pyt_words1 = pyt_raw.split() NLTK gir oss også et annet alternativ: »> pyt_words2 = nltk.word_tokenize(pyt_raw) Hva blir forskjellen på de to? Ser du fordeler ved å bruke word_tokenize? (Obs! NLTKs word_tokenize er optimalisert for engelsk og kan gi noen rare resultat for norsk.)

3. Et ord som *jeg* er det samme om det står først i en setning og skrives *Jeg*. Tilsvarende for andre ord. For en del anvendelser er det derfor en fordel å gjøre om teksten til bare små bokstaver før vi går videre. Gjør om alle ordene i pyt_words2 til små bokstaver, og kall resultatet pyt_low.

4. Plukk ut alle ordforekomstene i pyt_low som inneholder en av de norske bokstavene æ, ø, å. Hvor mange slike ordforekomster er det?

5. Vi er nå interessert i hvor mange forskjellige ord det er i teksten som inneholder en av de norske bokstavene. Plukk ut de unike forekomstene. Hvor mange er det?

6. Vi skal nå skrive de unike forekomstene til en fil. Når vi skal skrive noe til en fil, kan vi åpne fila ved »> f=open(<filnavn>, 'w') Vi kan skrive til fila ved å bruke »> f=write(<det vi vil skrive>) Og til slutt lukke fila ved »> f.close()

   Skriv de unike forekomstene av ord som inneholder æ, ø eller å til en fil kalt norske.txt, ett ord på hver linje.

Answer:

1. Fant ingen rusk, men lest over teksten. Antar pga navnet *raw* så menes det at vi skal lese inn fila og avventer til videre bearbeiding

2. Forskjellen på .split() og nltk_tokenize er at fil.split() spitter ordene default ved whitespace. nltk_tokenize bruker Penn Treebank Tokenizer som skiller ord ved hjelp av regulære uttrykk. Den kan derfor skille ut punktum og comma. Demonstrert nedenfor:

3. endret de til små bokstaver

4. Bokstavene forekommet 60 ganger i pyt_low

5. Bokstaven æ forekommet 2 ganger, bokstaven ø forekommet 11 ganger og bokstaven å forekommet 10 ganger

6. Skrev ut norske ord til Norske.txt

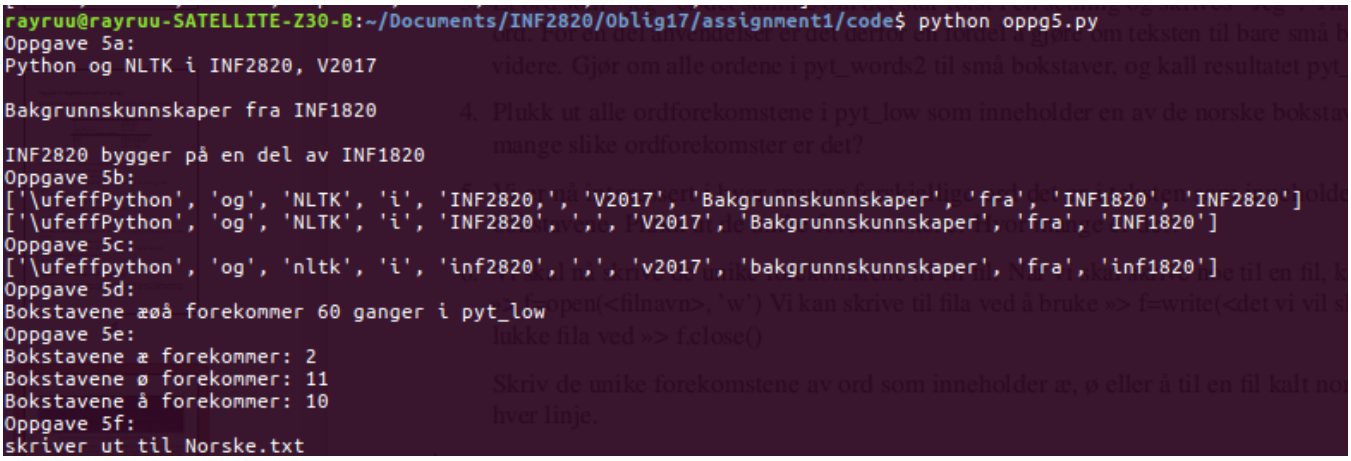

Figure 0.4: Skriver ut pyt_words1 og pyt_words2.

Oppgave 6: Regulære uttrykk av bearbeiding av tekst (4 poeng)

Legger ved programmet som erstatter filnavn og tall. Jeg endret ikke ord som INF2820 som hadde bokstaver i seg, bare rene tall. Jeg prøvde meg også på re.sub, men denne fikk jeg ikke til. Fikk hele tiden errors med funksjonen som jeg ikke forstod meg på.