Oppgaver INF 2820 V2012

Obligatorisk innleverinsoppgave 1, deloppgave B

- **Frist 24.2 kl 23.59 – strikt – se beskrivelse på semestersiden**
- **Besvarelsene skal leveres i devilry**

Forberedelser

Vi skal arbeide med kontekstfrie grammatikker og med verktøy for å evaluere dem. Vi skal bruke python og nltk. Først anbefales du å kjøre kommandoen
~oe/bin/inf2820 (innlogget som deg selv på IFIs linuxmaskiner.) Dette trenger du bare å gjøre en gang. Dermed vil du

- Få gjort foreberedelser til siste del av semesteret der du skal jobbe med lkb
- Få tilgang på kommandoen "nltk" som gir deg et emacs-oppsett for å arbeide med python og nltk
- Lagt opp en sti slik at du får direkte tilgang til /projects/nlp/nltk_data der nltk_data ligger. Du trenger da ikke laste dem til ditt eget område.

Dernest må du arbeide deg gjennom NLTK-boka kapittel 8 t.o.m. seksjon 8.3. Spesielt:

- Vi vil bruke nltk sitt format for å skrive kontekstfrie grammatikker. En grammatikk blir altså et Python-objekt. Vi vil ikke ha så mye bruk for den indre strukturen av dette objektet foreløpig.
- Vi vil skrive våre egne grammatikker til filer og lage Python-objekter av dem ved å bruke nltk.data.load()
- For parsingen vil vi i dette settet hovedsakelig bruke nltk.ChartParser() og ikke nltk.RecursiveDescentParser()

Oppgave 6: Basisgrammatikk

Vi tar utgangspunkt i oppgave3 (fra oblig1a). Vi skal nå lage en kontekstfri grammatikk (CFG) for samme fragment som vi der laget en NFA for. Se først på fragmentet uten preposisjonalfraser (PP-ledd) og lag en grammatikk for denne. Kall den basis_cfg. Lag så en chart-parser fra denne og kall den basis_parser. Test at parseren gjør det den skal. I grammatikken skal du bruke naturlige kategorinavn (ikke-terminaler) som NP og VP slik det er hintet til i oppgaveteksten for oppgave 3 og vist i eksempler i NLTK-boka, seksjon 8.1-8.3. Utvid leksikonet slik at du har minst 20 av hver av:

- Substantiv, egennavn
- Substantiv fellesnavn (disse 20 skal også forekomme både i bestemt og ubestemt form)
- Adjektiv
- Intransitive verb
- Transitive verb

Og minst 5 hver i de øvrige verb-klassene.

Innlevering: Fil som inneholder basis_cfg.

Oppgave 7: PP-ledd

Utvid basis_cfg til en grammatikk med PP-ledd, kall den pp_cfg, og lag chart-parser pp_parser. Du skal ha med minst 10 forskjellige preposisjoner. Det er her et poeng at vi kan få frem flertydigheter og at de reflekteres i trær. Se på setningene:

a) Dyret i huset ved vannet sov.
b) Kari sov i huset ved vannet.
c) Kari likte huset ved vannet.
d) Kari likte dyret i huset ved vannet.

Hver av setningen (a-c) har to analyser. Sjekk at du får dette. For hver av de tre eksemplene forklar med en til to setninger intuitivt forskjellen mellom de to analysene. Hvor mange analyser får (d)?

Innlevering: Fila som inneholder pp_cfg. Kjøringseksempler som viser trærne du får for setningene (a-d). Forklaringene det spørres etter.

Oppgave 8: Testing

Til oppgave 3 lagde vi en test-rutine i test_sentences.py. Modifiser denne slik at den nå virker for en chart_parser i stedet for en nfa. Bruk den til å teste pp_parser (og dermed pp_cfg) og se at du får riktig resultat.

Innlevering: Modifisert test_sentences.py og resultatet av å teste med test(pp_parser).

Oppgave 9: Utvidet grammatikk

Vi skal nå utvide grammatikken pp_cfg til å takle noen flere fenom. For det første skal vi ta med koordinasjon, og vi skal koordinere S-er, VP-er og NP-er, som i (a-c), resp.

a) Kari smilte og barnet sov.
b) Kari sov og smilte.
c) Ola og Kari smilte.

Dernest skal vi ta med to enkle typer relativer som kan uttrykkes med

d) som VP
e) som NP TV

En slik relativ kan modifisere en NP.

Hvor mange analyser får du nå for

f) Ola som fortalte at Kari sov og smilte likte barnet og huset ved vannet

Kall den utvidete grammatikken my_cfg og chart-parseren for my_parser.

Innlevering: Den utvidete grammatikken. Svar på hvor mange trær du får for (f) og kjøringseksempel med utskrift av trærne.

Oppgave 10: Utvidet testrutine

Test-rutinen vi utviklet i oppgave 8 har to begrensninger. Den inneholder både setningene vi vil teste mot, og rutiner for å utføre dette. Det kan være en fordel å skille mellom disse. Og den sier bare om en setning er i språket beskrevet av grammatikken eller ikke. Den tar ikke stilling til hvor mange analyser setningen har. Vi skal nå endre på dette.

For å skille mellom operasjoner og data, vil vi legge dataene i en fil. Så skal vi i en annen fil skrive en funksjon som tar to argumenter og kalles slik
parser_test(<parser>, <navn på eksempelfil>)

For å kunne ta stilling til om vi har riktig antall analyser vil vi ikke skille mellom setninger og ugrammatiske ordsekvenser, men bare markere hvor mange analyser de har, for eksempel:
3: Kari likte barnet og huset ved vannet
0: Kari og Ola
1: Kari sov

Når det gjelder resultatet, utskriften, ønsker vi at den skal ha formen

Gold:	Found:	Sentence:
3	2	Kari likte barnet og huset ved vannet
0	0	Kari og Ola
1	1	Kari sov

Her er ”gold” det samme som fasiten, det vi ønsker å finne, mens ”found” er det antall vi faktisk fant. Vi skulle altså ha funnet en analyse til av den førstes etningen.

OBS Denne testen vil bare si om vi har funnet riktig antall analyser, ikke om de analysene vi har funnet er riktige. Det kan godt være at en eller begge de to analysene vi finner for den første setningen ikke er blant de tre vi ønsker å finne.

Hint: For innlesning fra fil se på hvordan det gjøres i nfa.py.
For pen utskrift kan du ha bruk for
```
>>> " ".join(['Kari', 'sov', 'og', 'barnet'])
'Kari sov og barnet'
>>>
```
Det er på en måte en reversering av
’Kari sov og barnet’.split()
For tabeller kan du ha bruke for at ”\t” gir en tabulator

Innlevering: Filen med parser_test. Resultat av kjøringen parser_test(my_parser, ’testsentences.txt’)
’testsentences.txt’ finner du i /projects/nlp/in2820/cfg/ på IFIs linuxmaskiner

- SLUTT