INF1820 V2015 — Oppgave 3b
CFGer og semantikk

Innleveringsfrist, onsdag 20. mai

Lever inn svarene dine i Devilry (`https://devilry.ifi.uio.no`) i en fil som angir brukernavnet ditt, slik: `oblig3a_brukernavn.py`

En perfekt besvarelse på denne oppgaven er verdt 100 poeng.

1 En CFG for norsk (35 poeng)

Engelsk har samsvarsbøyning mellom subjekt og verb: subjekter i tredje person entall krever en spesiell form av verbet (*she likes*, *it likes* versus *I like*, *they like*). Engelsk har også, som norsk, noe kasusmarkering for pronomen. For eksempel, *I*, *he*, *they* er i nominativ kasus, mens *me*, *him*, *them* er i akkusativt kasus. I denne oppgaven skal du skrive en kontekstfri grammatikk som aksepterer setningene i (a), men ikke setningene i (b):

(a) Setninger grammatikken skal akseptere:

1. she likes him
2. they like him
3. they like the actress
4. they like the actresses
5. I know her
6. I know the actor whom she likes
7. I know the actress who likes him
8. the actress who likes him sings
9. the actresses who like him sing

(b) Setninger grammatikken ikke skal akseptere:

1. *she likes he
2. *they likes him
3. *me know her
4. *I know the actress whom likes him
5. *the actress who likes him sing
6. *the actress who like him sings
7. *the actresses who likes him sing
8. *the actresses who like him sings

NLTK inneholder flere forskjellige parsere som tildeler syntaktisk struktur til en setning automatisk, i henhold til en grammatikk. I denne oppgaven bruker vi `RecursiveDescent`-parseren som er beskrevet i del 8.3 av NLTK-boka. Du formulerer grammatikken direkte som en streng slik:

```
grammar = nltk.parse_cfg(u"""
  S -> NP VP
  VP -> V NP | V NP PP
  PP -> P NP
  V -> "saw" | "ate" | "walked"
  NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
  Det -> "a" | "an" | "the" | "my"
  N -> "man" | "dog" | "cat" | "telescope" | "park"
  P -> "in" | "on" | "by" | "with"
  """)
parser = nltk.RecursiveDescentParser(grammar)
```

Merk deg at flere alternativer for samme kategori kan formuleres med disjunksjon `VP -> V NP | V NP PP`, dvs. at `VP` enten kan bestå av en `V NP` eller `V NP PP`. Merk deg videre at RecursiveDescent-parseren ikke handterer venstre-rekursjon av typen `VP -> VP PP`, sa du må formulere grammatikken din uten denne formen for rekursjon.

Du kan teste grammatikken din på en setning slik:

```
sent = "Mary saw Bob".split()
for tree in parser.nbest_parse(sent):
   print tree
```

Dette vil skrive ut en analyse i klammenotasjon for hver analyse grammatikken tilordner setningen:

```
(S (NP Mary) (VP (V saw) (NP Bob)))
```

Implementer grammatikkfragmentet for engelsk som beskrevet over, test grammatikken din på setningene i (a) og (b) ovenfor og skriv ut resultatet. Pass på at grammatikken din faktisk generer et tre for alle setningene i (a) og ikke genererer noe tre for setningene i (b).

2 Syntaktisk annotasjon (30 poeng)

I denne oppgaven skal vi jobbe med annotering av syntaktisk struktur. Dataene du produserer her vil bli en del av Arnes doktorgradsarbeid om syntaktisk annotasjon, så her kan du bidra direkte til å flytte forskningsfronten!

Å gi en fullstendig syntaktisk beskrivelse av en hel setning er en tidkrevende prosess. Derfor deler vi her oppgaven opp i mange mindre avgjørelser, der du i hvert tilfelle skal ta stilling til et ja/nei-spørsmål om setningens struktur.

Gå til `http://sh.titan.uio.no:2910/user/new/inf1820/FGubnTHSRDXouwd3NKvak4FDTOQ` og opprett en ny bruker, og les hjelpeteksten (link øverst i høyre hjørne). Annoter så minst ti setninger. Mens du annoterer, merk deg hvilke avgjørelser som er vanskelige, samt setningene de forekommer i. Gi minst tre eksempler på vanskelige avgjørelser og beskriv kort hvorfor de er vanskelig. Antall setninger du har annotert kan du se på siden med informasjon om din bruker.

Merk at denne oppgaven kun vil vurderes etter svarene du gir på spørsmålet i forrige avsnitt, ikke på annotasjonene du produserer. Gruppelærerene har ikke tilgang på dataene, og hvis du velger et annet brukernavn enn IFI-navnet kan vi heller ikke koble dataene tilbake til deg.

3 Betydningsklassifikasjon med Naive Bayes (35 poeng)

Fila `wsd_tren.txt` inneholder (fiktive) treningsdata annotert med ordbetydning for lemmaet *skim*. Hver linje inneholder en liste med trekk og en kategori, skilt av mellomrom. Første element i hver linje er betydningen, resten er trekk. Første linje er:

```
Reading book day novel
```

Dette vil si at betydningen for dette eksempelet er READING og inneholder trekkene *book*, *day*, og *novel*.

Ved hjelp av treningsdataene i `wsd_tren.txt` lager du sannsynlighetsdistribusjoner med Python slik som vi gjorde for HMM-sannsynlighetene i forrige oblig, men denne gangen for de to distribusjonene vi bruker i Naive Bayes: $P(S)$, distribusjonen over betydninger, og $P(F|S)$, distribusjonen av trekk gitt betydning.

Ved hjelp av disse distribusjonene svarer du på følgende spørsmål:

- Hva er $P(\text{REMOVING})$, sannsynligheten til betydningen REMOVING?
- Ett av trekkene i fila er *day*. Hva er sannsynligheten for dette trekket gitt betydningen READING: $P(day|\text{READING})$?
- Fila `wsd_test.txt` inneholder et testeksempel på samme format som treningsdataene, men uten betydning:

  ```
  ? paper surface towards
  ```

  Bruk Naive Bayes-formelen og Python til å beregne den mest sannsynlige betydningen for dette eksempelet.

Husk at i Naive Bayes er den mest sannsynlige betydningen $\hat{s}$ gitt ved:

$$\hat{s} = \text{argmax}_{s \in S} P(s) \prod_{i=1}^{n} P(f_i|s)$$

Du kan lese mer om Naive Bayes for WSD i Jurafsky & Martin, 20.2.