Obligatorisk innlevering 2a - Språkmodeller og ordklassetagging

Innleveringsfrist: mandag 20. oktober kl. 23.59

Innlevering av oppgaven skjer i Devilry. Det er viktig at du setter deg inn i retningslinjene for obligatoriske oppgaver. Besvar oppgavene i én fil som angir brukernavnet ditt slik:

`oblig2a_brukernavn.py`

Besvarelsen må kunne kjøres med Python i terminalen, slik som tidligere. Du skal også levere inn filen `setninger.txt` som du har fått utlevert med denne innleveringen. Ikke pakk filene inn i en ZIP-mappe.

Poeng

En perfekt løsning av denne innleveringen er verdt 100 poeng. Du må vise koden din i oppgave 3 og 4. Koden må kunne kjøres og må inneholde kommentarer som forklarer hva koden gjør.

1 Språkmodeller (eksamen våren 2017) (10 poeng)

Under finner du formelen for en språkmodell (en såkalt bigrammodell) og et lite tekstkorpus.

Formel:

$$P(w_1...w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1})$$

Korpus:

```
<s> Per synger ikke <\s>
<s> Kari synger <\s>
<s> Ola synger ikke <\s>
```

I denne oppgaven skal du ta utgangspunkt i denne formelen og i tekstkorspuset for å besvare følgende spørsmål:

1. Hvilke bigrammer forekommer i korpuset?
2. Hvordan beregner vi sannsynligheten for et ord gitt det foregående ordet $P(w_i|w_{i-1})$ fra et korpus?
3. Du skal nå bruke bigrammodellen og tekstkorpuset til å beregne sannsynligheten for setningen `"<s> Kari synger ikke <\s>"`. Vis hvilke sannsynligheter du trenger, og hvordan disse beregnes fra korpuset. Du trenger ikke å regne ut den totale sannsynligheten for setningen.

2 Ordklasser (10 poeng)

Gitt ordklassene i tabellen under, tildel ordklasser til alle ordene i setningene. Du må velge ett alternativ for hvert ord.

Tag	Ordklasse
CC	konjunksjon
DET	determinativ
JJ	adjektiv
NN	substantiv
PR	preposisjon
PO	pronomen
RB	adverb
SB	subjunksjon
VB	verb

a) `Jesper drikker saft uten sugerør`

b) `Hun løper raskt som en atlet`

3 Språkmodeller med Python (50 poeng)

I denne oppgaven skal du trene en bigrammodell med NLTK. Ta utgangspunkt i stegene og Python-koden som ble forklart i gruppeoppgave 4. I motsetning til i gruppeoppgaven skal vi her trene en bigrammodell og ikke en trigrammodell, så du blir nødt til å gjøre noen endringer i koden. Vi skal videre bruke Gutenberg-korpuset, og fokusere på dokumentet `bible-kjv.txt` som inneholder bibeltekst. Du skal i denne oppgaven skrive Python-kode som gir deg svar på følgende spørsmål:

1. Hva er totalt antall tokens i teksten?
2. Hva er totalt antall ordtyper i teksten?
3. Hva er de 20 mest frekvente ordtypene i teksten?
4. Hva er frekvensen til henholdsvis ordene *"earth"*, *"death"*, og *"life"*?
5. Hvilke bigram forekommer i den syvende setningen?
6. Hvilke trigram forekommer i den åttende setningen?
7. Tren en bigrammodell og generér en tekst ved bruk av denne. Sørg for at den genererte teksten inneholder minst 50 ord (inkludert markører for setningsstart og -slutt).
8. Hva er sannsynligheten for den genererte teksten?

4 Ordklassetagging med regulære uttrykk (30 poeng)

I denne oppgaven skal du lage en ordklassetagger ved bruk av regulære uttrykk samt evaluere taggeren din på et ordklassetagget korpus.

Taggeren med regulære uttrykk i NLTK-boka (del 5.4.2, under overskriften The Regular Expression Tagger) sjekker kun noen få uttrykk, så her er det masse rom for forbedring: for eksempel kan vi tagge alle ord som slutter på *"-able"* som adjektiv.

I del 5.4.2 av NLTK-boka er de regulære uttrykkene definert i en liste med "regler", eller "mønstre" slik:

[1]:
```
import nltk

patterns = [
    (r'.*ing$', 'VBG'),               # gerunds
    (r'.*ed$', 'VBD'),                # simple past
    (r'.*es$', 'VBZ'),                # 3rd singular present
    (r'.*ould$', 'MD'),               # modals
    (r'.*\'s$', 'NN$'),               # possessive nouns
    (r'.*s$', 'NNS'),                 # plural nouns
    (r'^-?[0-9]+(\.[0-9]+)?$', 'CD'), # cardinal numbers
    (r'.*', 'NN')                     # nouns (default)
]

regexp_tagger = nltk.RegexpTagger(patterns)
```

Taggeren kan testes på en ikke-tagget setning fra korpuset slik: *(Dette påvirker ikke selve taggeren.)*

[ ]:
```
regexp_tagger.tag(CORPUS_NON_TAGGED_SENTENCE)
```

For å teste nøyaktigheten til taggeren din, kan du bruke metoden `evaluate()`:

[ ]:
```
regexp_tagger.evaluate(CORPUS_TAGGED_SENTENCES)
```

1. Definer en tagger ved hjelp av `nltk.RegexpTagger` der du har minst 10 uttrykk i tillegg til de som er nevnt i boka. Dokumentér alle reglene dine, og gi minst ett eksempel for hvert uttrykk på ord som dekkes. Husk å håndtere liten og stor bokstav. Enkeltord kan også brukes i de regulære uttrykkene, slik at f.eks. *"the"* alltid vil tagges som determinativ.
2. Bruk `adventure`-kategorien i Brown mens du utvikler taggeren din ved å teste nøyaktigheten til taggeren med metoden `evaluate()`. Når du er fornøyd med reglene dine, skal du teste taggeren på kategorien `fiction` i Brown, og rapportere resultatene. Det er viktig at du ikke endrer reglene dine etter dette. **Hint:** Ta utgangspunkt i NLTK-boka, del 5.4.2. **Merk:** Poengene du får på denne oppgaven er ikke basert på nøyaktigheten til taggeren i den endelige evalueringen på `fiction`-kategorien. Det er helt vanlig at nøyaktigheten til en tagger synker når man tester den på et annet korpus enn den det ble utviklet på.
3. Til slutt skal programmet ditt lese inn filen `setninger.txt` som er lagt ut sammen med denne obligen og tagge den med taggeren du har laget. Husk å tokenisere teksten først. Siden hvert token allerede er adskilt med mellomrom, kan du gjøre dette med `split()`. Skriv ut resultatet til en ny fil. Denne kan du kalle `taggede_setninger.txt`. Diskuter minst 3 av feilene taggeren gjør, og kom med forslag til hvordan den kan forbedres.