INF1820 V2017 – Oblig 3a
HMM'er og chunking

Innleveringsfrist: fredag 31 mars

Lever inn svarene dine i Devilry (`https://devilry.ifi.uio.no/`) i filer som angir brukernavnet ditt, slik: `oblig3a_brukernavn.py`. Pass på at fila di kan kjøres som et program; det skal ikke være en REPL-sesjon limt inn i en fil.

En perfekt besvarelse på denne oppgaven er verdt 100 poeng.

1 Betinget sannsynlighet (30 poeng)

I denne oppgaven ser vi på hvordan vi kan beregne sannsynlighetene som er byggesteinene i en HMM tagger, dvs. **observasjonssannsynligheter** (*emission probabilities*) $P(w_i|t_i)$ og **transisjonssannsynligheter** (*transition probabilities*) $P(t_i|t_{i-1})$, og hvordan vi lett kan beregne disse med innebygd funksjonalitet i NLTK.

1. Bruk `nltk.ConditionalFreqDist()` til å beregne frekvens av (tagg, ord) par for hele Brown-korpuset. Dersom du er usikker på hvordan denne funksjonen fungerer kan du lese mer om dette i NLTK-boken (kapittel fire og fem). Frekvensdistribusjonen skal ha ordklassetaggene som betingelser, dvs. at `cfd.conditions()` gir en liste med ordklassetagger (feks. NN, DT, samt en rekke andre tagger) for distribusjonen `cfd`.

2. Besvar følgende spørsmål ved hjelp av frekvensdistribusjonen fra forrige oppgave:

   (a) hva er det mest frekvente substantivet?

   (b) hvor ofte forekommer substantivet *linguist*?

   (c) hva er det mest frekvente adjektivet?

   Her skal du altså skrive Python-kode som beregner og skriver ut svaret på oppgave (a)-(c) over.

3. Vi gjør om frekvensdistribusjonen (her med variabelnavnet `cfd`) til en sannsynlighetsdistribusjon (basert på Maximum Likelihood Estimation (MLE)) med funksjonen `nltk.ConditionalProbDist()`, slik:

   ```
   cpd = nltk.ConditionalProbDist(cfd, nltk.MLEProbDist)
   ```

   `cpd['JJ'].max()` gir deg det mest sannsynlige adjektivet,
   og `cpd['JJ'].prob("new")` gir deg sannsynligheten for *new*, gitt en adjektivtagg (P(*new*|JJ)).

4. For å beregne transisjonssannsynlighetene trenger vi frekvensinformasjon for bigrammer av tagger. Her bruker vi `nltk.bigrams()` som tar inn en liste og gir ut alle bigrammer basert på denne:

   ```
   >>> liste = ["the","man","saw","her"]
   >>> nltk.bigrams(liste)
   [('the', 'man'), ('man', 'saw'), ('saw', 'her')]
   ```

I motsetning til eksempelet over skal du her trekke ut bigrammer av tagger (og ikke ord):

```
>>> liste = ["DT","NN","VBD","PP$"]
>>> nltk.bigrams(liste)
[('DT', 'NN'), ('NN', 'VBD'), ('VBD', 'PP$')]
```

Trekk ut alle bigrammer av tagger fra Brown-korpuset med `nltk.bigrams()` og lagre disse i en variabel.

5. Bruk `nltk.ConditionalFreqDist()` til å beregne frekvens for bigrammene og besvar følgende spørsmål (her skal du altså skrive Python-kode som beregner og skriver ut svaret på oppgave (a)-(b) under.):

   (a) hvilken tagg er det som oftest etterfølger et substantiv i Brown-korpuset?

   (b) hvor ofte forekommer bigrammet 'DT NN'?

6. Igjen kan du utlede en sannsynlighetsdistribusjon fra frekvenstellingene over bigrammene med `nltk.ConditionalProbDist()`.

   (a) hva er den betingede sannsynligheten for et substantiv, gitt et determinativ (P(NN|DT))? Svaret skal også her beregnes og printes fra Python-skriptet ditt.

2 HMM-tagging (30 poeng)

I denne oppgaven skal vi sammenligne sannsynligheten for to taggsekvenser for samme setning. Ta en titt på J&M kapittel 5.5.1, side 176-178 og gå fram på nøyaktig samme måte for å løse denne oppgaven.

Her er setningen med to taggsekvenser:

(a)	PPSS	VBD	PP$	NN
	I	saw	her	duck

(b)	PPSS	VBD	PPO	VB
	I	saw	her	duck

(PPSS er taggen for personlige pronomen i nominativ som ikke er 3. person entall, f. eks. *I, they*. PP$ er taggen for possessive pronomen, og PPO er taggen for personlige pronomen i akkusativ).

Som i eksempelet fra J&M starter setningene på samme måte og taggsekvensene har kun fire sannsynligheter som skiller seg fra hverandre. Vi bortser her fra flertydigheten for *saw*.

Bruk koden fra Oppgave 1 og hele Brown-korpuset for å beregne transisjonssannsynligheter, f.eks. P(PP$|VBD), og observasjonssannsynligheter, f.eks. P(*her*|PP$). Bruk disse sannsynlighetene til å beregne sannsynligheten for den delen av taggsekvensen der (a) og (b) er forskjellig (dvs de to ulike sekvensene for *her duck*).

Hvilken lesning er mest sannsynlig?

3 Chunking (40 poeng)

I denne oppgaven skal du lage en NP chunker og evaluere den. For å få til dette skal du bruke `nltk.RegexpParser`, som beskrives i NLTK-boken, kapittel 7:

```
grammar = "NP: {<DT>?<JJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
```

Bruk minst fem mønstre som ikke forekommer i boken (eller utvid mønstrene fra boken på fem forskjellige måter). Mens du utvikler chunkeren din kan du teste den på treningskorpuset fra CoNLL2000. Dette korpuset er hentet fra Penn Treebank og bruker altså ordklassetaggsettet derfra (som er gjengitt og beskrevet kort på innsiden av omslaget foran i Jurafsky & Martin). Korpuset får du tilgang til slik:

```
from nltk.corpus import conll2000
training_chunks = conll2000.chunked_sents("train.txt", chunk_types=["NP"])
```

Hvis du trenger inspirasjon til mønstrene dine kan du skrive ut treningskorpuset og inspisere elementene merken 'NP'. Du evaluerer chunkeren din på treningskorpuset slik (gitt at `cp` er variabelen som inneholder chunkeren din og `training_chunks` er variabelen som inneholder treningskorpuset):

```
cp.evaluate(training_chunks)
```

Dokumentér chunkeren grundig og forklar hvordan den fungerer. Evaluér chunkeren ved å beregne dennes nøyaktighet på **testkorpuset** fra CoNLL2000:

```
test_chunks = conll2000.chunked_sents("test.txt", chunk_types=["NP"])
```

Kopiér output fra evalueringen inn i kodefilen din.