INF1820, V2014 – Obligatorisk oppgave 3a
HMMer og chunking

Innlevering: 11/4

Registrer svarene dine i en fil som angir brukernavnet ditt slik:

`oblig3a_brukernavn.py`

Som vanlig skjer innlevering av oppgaven i Devilry. Se emnesiden for mer informasjon om reglement rundt innlevering, samt bruk av Devilry.

En perfekt løsning av 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 og hvordan vi lett kan beregne dem med innebygd funksjonalitet i NLTK. Bruk `nltk.ConditionalFreqDist` til å beregne frekvensen til (tagg, ord)-par for hele Brown-korpuset. For å finne ut hvordan du bruker denne klassen kan du lese i NLTK-boka (kapittel fire og fem viser hvordan den brukes) eller, hvis du er eventyrlysten, lese den tekniske dokumentasjonen på `http://nltk.org/_modules/nltk/probability.html#ConditionalFreqDist` og `http://nltk.org/api/nltk.html#nltk.probability.FreqDist`. Ved hjelp av denne distribusjonen, finn ut av:

1. Hva er det meste frekvente verbet?
2. Hvor ofte forekommer *time* som substantiv? Og som verb?
3. Hva er det mest frekvente adjektivet?

For beregne transisjonssannsynlighetene trenger vi informasjon om taggbigrammene i Brown. Dette lager vi med funksjonen `nltk.bigrams()`, som tar inn en liste og returnerer en liste som inneholder bigrammene i listen:

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

Lag så en liste over bigrammene i Brown og lagre den i en variabel, og bruk denne til å lage en `ConditionalFreqDist` du bruker til å svare på følgende:

1. Hvilken tagg forekommer oftest etter et verb i Brown?
2. Hvor ofte forekommer bigrammet 'DT NN'?

Fra en frekvensdistribusjon (som her har navnet `cfd`) kan vi så lage en sannsynlighetsdistribusjon basert på MLE-metoden (Maximum Likelihood Estimation) ved hjelp av:

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

Fra dette objektet kan vi hente ut det mest sannsynlige adjektivet med `cpd["JJ"].max()` eller sannsynligheten til *new* gitt adjektivtagg med: `cpd["JJ"].prob(new")`. Lag sannsynlighetsdistribusjoner for de to frekvensdistribusjonene du allerede har laget og finn ut av:

1. Hva er det mest sannsynlige verbet?
2. Hva er $P(NN|DT)$, sannsynligheten for substantivtagg etter en bestemmertagg?

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:

(a)	VBD	PP$	NN
	saw	her	duck

(b)	VBD	PPO	VB
	saw	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 5 mønstre som ikke forekommer i boken (eller utvid mønstrene fra boken på 5 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. Korpuset får du tilgang til slik:

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

```
nltk.chunk.util.accuracy(cp, training_chunks)
```

(NB! `cp.evaluate`-metoden som brukes i NLTK-boken eksisterer ikke lenger.)

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

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

For å evaluere chunkeren benytter du `nltk.chunk.util.accuracy()`, som beskrevet over.