INF1820 V2013 — Oppgave 3a
HMMer og chunking

Innleveringsfrist, onsdag 29. april

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

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

For beregne transisjonssannsynlighetene trenger vi informasjon om tagg-bigrammene 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 substantiv i Brown?
2. Hvor ofte forekommer bigrammet 'DT JJ'?

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(nnew"). Lag sannsynlighetsdistribusjoner for de to frekvensdistribusjonene du allerede har laget og finn ut av:

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

2 HMM-tagging (30 poeng)

I denne oppgaven skal vi sammenligne sannsynligheten for to taggsekvenser for samme setning. Denne oppgaven løser du på nøyaktig samme måte som beskrevet i Jurafsky & Martin, kapittel 5.5.1 (side 176–178).

Den engelske setningen "I saw her duck" kan ha to mulige taggsekvenser: PPSS VBD PP$ NN eller PPSS VBD PPO DB (PP$ er taggen for possessive pronomen, PPO for personlige pronomen i akkusativ). Setningen kan altså ha to betydninger, oversatt til norsk enten "jeg så anden hennes" (PP$ NN) elller "jeg så henne dukke" (PPO VB).

Som i eksempelet i boka starter setningene likt, og taggsekvensene har kun fire sannsynligheter som er forskjellige. Ved hjelp av sannsynlighetsdistribusjonene du laget i oppgave 1, beregn sannsynligheten for den delen av taggsekvensen der (a) og (b) er forskjellige:

(a)	VBD	PP$	NN
		her	duck
(b)	VBD	PPO	VB
		her	duck

Hvilken lesning er mest sannsynlig?

3 Chunking (40 poeng)

I denne oppgaven skal du lage en NP-chunker. Dette gjør vi med nltk.RegexpParser, slik det er beskrevet i kapittel 7 av NLTK-boka, delkapittel 7.2:

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

Bruk minst fem mønstre som ikke forekommer i boka, eller utvid mønstrene på fem forskjellige måter. For å utvikle chunkeren din tester du på treningssettet fra CoNLL2000. Dette korpuset er hentet fra Penn Treebank og bruker derfor PTB-taggsettet (som er gjengitt og beskrevet kort på innsiden av omslaget foran i Jurafsky & Martin). Korpuset henter du inn slik:

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

Du evaluerer slik (antatt at cp) inneholder chunkeren)[1]:

nltk.chunk.util.accuracy(cp, training)

Dokumenter chunkeren grundig og forklar hvordan den virker og hvorfor reglene er slik de er, og når du er ferdig evaluerer du chunkeren ved å beregne nøyaktigheten på *testkorpuset* fra CoNLL2000, som du henter slik:

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

For å gjøre den endelige evalueringen bruker du nltk.chunk.utill.accuracy() som beskrevet over.

[1] NB! Boka bruker en metode cp.evaluate() som ikke finnes lenger!