IN1140 H2021 – Oblig 2b
Syntaks og Semantisk klassifisering

Innleveringsfrist: tirsdag 05.11 kl.23.59

Lever inn svarene dine i Devilry (`https://devilry.ifi.uio.no/`) i filer som angir brukernavnet ditt, slik: `oblig2b_brukernavn.py`. Pass på at filen din kan kjøres som et program.
En perfekt besvarelse på denne oppgaven er verdt 100 poeng. Merk at begge obligene (1(a+b), 2(a+b)) må vurderes som bestått for å kunne ta eksamen.

1 Kontekstfrie grammatikker (eksamen V2017 – 10 poeng)

Anta at vi har følgende grammatikk:

```
S -> NP VP
NP -> Per | Kari | Ola | boka
NP -> D N
D -> en
N -> bok
PP -> P NP
P -> med
VP -> V NP
VP -> V NP NP
VP -> VP PP
V -> gir
V -> slår
```

Setningen "Per slår Ola med boka" er strukturelt flertydig.

1. Beskriv de to ulike tolkningene med egne ord.
2. Den vedlagte grammatikken gir kun én analyse for denne setningen. Hvordan kan du utvide grammatikken slik at den kan representere flertydighet av denne typen?
3. Er grammatikken rekursiv? Hvis ja, på hvilken måte? Hvis nei, gi et eksempel på en rekursiv regel.

2 Chunking (15 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 merket '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.

3 Naive Bayes klassifisering (15 poeng)

Vi ønsker å utvikle et system som automatisk skal angi om en setning er positiv eller negativ. Vi har et lite korpus som er annotert med sentiment som presentert i tabell 1.

	Kat	Dokument
Train	POS	en fortreffelig start på konserten
	POS	et samspilt og førsteklasses orkester
	POS	har aldri laget et ordentlig dårlig album
	NEG	konserten viste seg å være én av de dårligste i mitt liv
	NEG	oppskrytt orkester
	NEG	kjedelig konsert
	NEG	et ordentlig dårlig album
Test	?	førsteklasses artist men dårlig og kjedelig album
	?	fortreffelig orkester og flott album

Table 1: trenings- og testkorpus

Vi ønsker å beregne den mest sannsynlige klassen k for en testsetningen, gitt en trekkvektor v. Vi bestemmer oss for å løse denne oppgaven ved Naive Bayes-klassifisering, angitt ved følgende formel:

$$\hat{k} = \underset{k \in K}{\operatorname{argmax}} \; P(k) \prod_{j=1}^{n} P(v_j|k)$$

1. Forklar hvordan Naive Bayes-formelen er utledet fra den opprinnelige sannsynligheten $P(k|\overrightarrow{v})$ og beskriv hvilke antagelser som er gjort.
   Hint Husk at hovedantagelsen i Naive-Bayes er:

$$\hat{k} = \underset{k \in K}{\operatorname{argmax}} \; P(k|\overrightarrow{v})$$

2. Beregn den mest sannsynlige klassen for hver av testsetningene "*førsteklasses artist men dårlig og*

kjedelig album" og "*fortreffelig orkester og flott album*". Anta en Naiv Bayes-klassifiserer og bruk `add-one` smoothing.

3. Er du enig med klassifiseringen av setningene? Hvis ikke, forklar og foreslå hva du tror må til for å få Naive Bayes til å angi riktig klasse.

4 Naive Bayes klassifisering med Python (60 poeng)

Begynn med å gjøre deg kjent med Naive Bayes-klassifisering. Dersom du ikke allerede er det, se lysark fra forelesning og/eller les kapittel 4 fra Jurafsky & Martin. Det er også lurt å se på filen `doc_classification_BOW.py` fra emnesiden (ved foilene til forelesningen om maskinlæring) da den inneholder en del hint som kan hjelpe med å løse oppgaven.

Her skal du bruke et begrenset utvalg av det såkalte *NoReC* korpuset der vi skal fokusere på bokanmeldelsene. Siden korpuset ikke er en del av NLTK har vi skrevet litt kode som du kan bruke for å laste inn korpuset og bruke det slik korpora fra NLTK blir brukt. Du finner også kode som deler dataene din inn i separate datasett for trening, utvikling, og testing (`train_set`, `dev_set` og `test_set`). Du må laste ned korpuset fra emnesiden, og ha mappen *NoReC* (og eventuelt *negNoReC*) i samme mappe som kodefilen din. I denne oppgaven skal du trene flere Naive Bayes klassifiserere. Bruk Python til å utføre beregningene dine.

Spørsmål 1: Bruk dataene fra *NoReC* til å trene en Naive Bayes klassifiserer som bruker de 1000 mest frekvente ord som trekk. Her skal du fordele dataene din i `train_set` og `test_set`. Hva synes du om resultatene? Sjekk topp 10, 20, og 30 mest informative trekk og diskutere hva du tror kan være problemet, og hva du tror kan forbedre resultatene.

Spørsmål 2: Å kun bruke de X mest frekvente ord som trekk er kanskje ikke den beste måten å få pålitelige klassifiseringer.
Hvilke trekk tror du kan forbedre din klassifiserer? Velg minst tre av følgende modeller:

1. **Bag-Of-Words (BOW)**: uten stoppord og tegnsetting.
2. **BOW + bigrams**: uten stoppord og tegnsetting, kombinasjon av unigrams og bigrams.
3. **BOW + bigrams + trigrams**: uten stoppord og tegnsetting, kombinasjon an unigrams, bigrams, og trigrams.
4. **BOW + bigrams + trigrams + negasjon**: som 3, men håndterer også negasjon ved å legge til prefikset "`NOT_`" før hvert ord som etterfølger en negasjon frem til setningslutt eller tegnsetting. Dette har vi allerede gjort, og dere har derfor tilgang til et annet korpus *negNoReC* som inneholder denne informasjonen. Vi har brukt følgende liste av negasjon: `['ikke', 'ingen', 'aldri', 'neppe', 'ingenting', 'mindre', 'knapt']`. Korpuset skal nå brukes isteden for *NoReC*, og kan lastes ned og brukes på samme måte.

Her skal du bruke fordelingen `train_set`, `dev_set` og `test_set`. Du skal trene modellene dine på `train_set` og beregne nøyaktighet (accuracy) for hver modell på `dev_set`. Først etter å ha trent alle modellene dine og valgt den som gir det beste resultatet, kan du beregne nøyaktigheten på `test_set`.

Her skal du:

- for hver modell rapportere accuracy og inspisere resultatene ved å se på topp 10 mest informative trekk.
- rapportere hvilken modell som gir høyest accuracy.
- basert på din analyse av den beste modellen, kan du tenke deg andre trekk som kunne ha forbedret den ytterligere?