IN2110 – Obligatorisk innlevering 2a

*Leveres innen **tirsdag 21 april kl. 23.59** i Devilry.*

Du bør lese gjennom hele oppgavesettet før du setter i gang. Har du spørsmål så delta i gruppetimen, spør på Piazza eller send epost til `in2110-hjelp@ifi.uio.no`.

Oppsett

For IN2110 har vi laget et utviklingsmiljø for Python som inneholder programvare og data for obligene. For mer informasjon om bruk av miljøet på IFI's Linux maskiner eller via SSH, eller for installasjon på egen maskin, se;
`https://github.uio.no/IN2110/in2110-shared`

I oppgavene under skal dere bygge videre på pre-kode som er tilgjengelig under mappen `obliger/2a/` i kurs-repoet for gruppetimer og obliger:
`https://github.uio.no/IN2110/in2110-lab/`

Bakgrunn

I denne obligatoriske innleveringen skal vi jobbe med dependensparsing av norsk tekst. Vi skal gjøre oss kjent med input-formatet til de fleste parsere (det såkalte CoNLL-U formatet), samt jobbe oss gjennom en transisjonsbasert parsingsalgoritme for dependensparsing. Deretter skal vi benytte NLP-biblioteket spaCy til å trene en dependensparser på en norsk trebank samt evaluere kvaliteten på parseren. Til slutt skal vi se nærmere på hvordan parseren fungerer på andre varianter av norsk.

Innleveringsformat

Innleveringen skal bestå av en Python-fil med kode, en modell-fil og en rapport i PDF-format. Rapporten skal inneholde de tallene og figurene det spørres om, samt svar på spørsmål. Tall bør presenteres i en tabell, slik som illustrert i tabell 1 under.

Navn	UAS	LAS
System X	55.6	50.5
System Y	49.2	45.5
System Z	61.2	59.6

Tabell 1: Eksempel på tabell med fiktive resultater

Det forventes at dere bruker hele setninger – stikkord er ikke nok. Python-filen skal inneholde de ferdig implementerte funksjonene i pre-koden, samt koden dere har brukt for å produsere de resultatene dere presenterer i rapporten.

Oppgave 1 Dependensgrammatikk

id	form	lemma	upos	head	deprel
1	I	i	ADP	3	case
2	100	100	NUM	3	nummod
3	år	år	NOUN	8	obl
4	er	være	AUX	8	aux
5	Nobels	Nobels	PROPN	8	nsubj:pass
6	Fredspris	Fredspris	PROPN	5	flat:name
7	blitt	bli	AUX	8	aux:pass
8	delt	dele	VERB	0	root
9	ut	ut	ADP	8	compound:prt
10	.	$.	PUNCT	8	punct

Tabell 2: Setning annotert med pos-tagger og dependensrelasjoner.

a) Dependensgrafer

Tegn grafen for setningen i tabell 2.

b) Transisjonsparsing

Vis en full sekvens av transisjonsoperasjoner for dependensgrafen fra forrige deloppgave med enten *arc standard* (se Jurafsky & Martin, kapittel 15, side 10) eller *arc eager*-algoritmen (se Jurafsky & Martin, kapittel 15, side 16 samt forelesningsnotatene). Angi tydelig hvilken algoritme du benytter i besvarelsen din. Forklar kort hvordan de to algoritmene skiller seg fra hverandre.

spaCy

I denne obligen skal vi bruke *spaCy*[1], som er et Python-bibliotek som kan brukes til en rekke NLP-oppgaver som tokenisering, ordklassetagging, parsing, named entity recognition, m.m. spaCy lar oss også trene egne modeller og her skal vi trene vår egen dependensparser for norsk.

Oppgave 2 Trene modeller

Vi skal bruke den norske bokmåls-trebanken fra *Universal Dependencies*[2]. Dataene kommer i *CoNLL-U-formatet*[3] og må konverteres til spaCy sitt json-format før vi kan trene modellen. Dette har vi allerede gjort for dere og disse json-filene (`no_bokmaal-ud-train.json` og `no_bokmaal-ud-dev.json`) distribueres sammen med pre-koden og kan brukes for å trene en modell:

```
$ spacy train -p parser nb <model-mappe> <train-set-json> <dev-set-json>
```

spaCy lagrer modellen etter hver *epoch* – en full iterering over hele treningssettet – i modell-mappen. I tillegg lagres den beste modellen (den med best ytelse på dev-settet) i mappen `model-best`.

> Tren en modell på trenings-settet. Om det går veldig sakte kan dere bruke opsjonen `-n <n>` til `spacy train` for å begrense antall epochs. Dere kan begrense helt ned til 5 epochs (default er 30) og fortsatt få helt ok ytelse, men husk å skrive det i rapporten om dere gjør det.
>
> Lever inn modellen som ligger i `model-best`.

Oppgave 3 Evaluering

Vi skal nå jobbe med evaluering av parseren og skal derfor parse development-datasettet `no_bokmaal-ud-dev.conllu` med parseren vi har trent. Den ferdig trente modellen fra forrige oppgave kan vi laste inn i spaCy:

```
>>> import spacy
>>> nb = spacy.load("my-model/model-best")
```

UD-filene er i CoNLL-U-formatet, og kan ikke leses direkte av spaCy. For å lese inn kan dere bruke klassen `ConlluDoc` som allerede er importert i pre-koden:

```
>>> conllu_dev = ConlluDoc.from_file("no_bokmaal-ud-dev.conllu")
```

Disse kan vi konvertere til spaCy-objekter

```
>>> dev_docs = conllu_dev.to_spacy(nb)
```

[1] `https://spacy.io/`
[2] `https://universaldependencies.org/`
[3] `https://universaldependencies.org/format.html`

Dette gir dere en liste med Doc-objekter som hver representerer en setning. Hvert enkelt Doc-objekt er igjen en liste med Token-objekter (med relevante attributter som `head`, `i` og `dep_`, som angir henholdsvis hode, indeks og dependensrelasjon for et gitt token). Se `https://spacy.io/api/doc` for ytterligere dokumentasjon av APIet. Disse objektene inneholder nå gull-dataene for devsettet vårt. For å evaluere trenger vi også setninger uten annotasjoner som vi kan parse med parseren vår. Det kan vi få ved å kjøre

```
>>> dev_docs_unlabeled = conllu_dev.to_spacy(nb, keep_labels=False)
```

og vi kan parse et Doc-object slik

```
>>> doc = dev_docs_unlabeled[13]
>>> doc.is_parsed
>>> doc
I rene ord.
>>> nb.parser(doc)
>>> doc.is_parsed
True
```

a) Attachment score

Det er vanlig å evaluere dependensparsing med *unlabeled attachment score (UAS)* og *labeled attachment score (LAS)*. UAS og LAS er varianter av accuracy og er definert som følger:

$$\text{UAS} = \frac{\#\text{ words with correct head}}{\#\text{ words}} \tag{1}$$

$$\text{LAS} = \frac{\#\text{ words with correct head and deprel}}{\#\text{ words}} \tag{2}$$

Fullfør funksjonen `attachment_score()`. Den skal ta inn to argumenter: en liste med gull-setninger og en liste med prediksjoner. Den skal returnere en tuppel med to tall: UAS og LAS.

Du skal nå beregne UAS og LAS for parseren din når den testes på development-settet og beskrive resultatene dine i rapporten.

b) Evaluering på andre teksttyper

UD-trebanken vi har jobbet med hittil består i hovedsak av nyhetstekster på Bokmål. Vi ønsker nå å undersøke hvordan parseren vi har trent fungerer på andre typer tekst. De norske UD-trebankene inneholder også en nynorsk trebank, samt en talespråkstrebank (den såkalte LIA-trebanken) som vi skal evaluere parseren på.

Test parseren du har trent på datasettene `no_nynorsk-ud-dev.conllu` og `no_nynorsklia-ud-dev.conllu` og rapporter resultatene (UAS og LAS) i en tabell. Beskriv kort hva du observerer.

Ta for deg `no_nynorsklia-ud-dev.conllu`-filen og se nærmere på de ti første setningene (eller flere hvis du vil). Ser du noe som kan bidra til å forklare resultatene dine?