IN2110 – Obligatorisk innlevering 2a

*Leveres innen **15. april kl. 23.59** i Devilry.*

Les gjennom hele oppgavesettet før du setter i gang. Dersom du har spørsmål så kan du gå på gruppetime eller spørre på Discourse[1]. Du kan også sende epost til `in2110-hjelp@ifi.uio.no` dersom alternativene over av en eller annen grunn ikke passer for spørsmålet ditt.

Oppsett

Vi bruker det samme Python-miljøet som for de foregående oppgavene. Den innledende treningen av modellen gjøres fra kommandolinja. Selve programmeringen gjøres i en Jupyter Notebook som tidligere. Pass på at du laster ned alt innholdet i mappen for Oblig 2a, slik at du får med både dataene og undermappen med hjelpeprogrammet.

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 Jupyter Notebook-fil med kode og forklaringer, en modell-fil og en rapport i PDF-format. Rapporten skal inneholde alle svarene på oppgave 1 Dependensgrammatikk. Jupyter Notebook-filen skal inneholde alle kommentarer og tabeller fra oppgave 2 og 3. Legg til celler ved behov. Modell-filen som skal leveres er en zip-fil av en mappe som lages i Oppgave 2.

Tall bør presenteres i en tabell, slik som illustrert i tabell 1 under.

[1] `https://astro-discourse.uio.no/c/in2110-24v`

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. Notebook-filen skal inneholde de ferdig implementerte funksjonene, 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. Du velger selv hvordan du tegner grafen og både bilde av håndtegning og figurer fra et tegneprogram er godkjent. Dersom du er vant til å jobbe i LaTeX, anbefaler vi tikz-dependency biblioteket.

b) Transisjonsparsing

Vis en full sekvens av transisjonsoperasjoner for dependensgrafen fra forrige deloppgave med enten *arc standard* (se Jurafsky & Martin, kapittel 18, side 2, samt forelesningsnotatene) eller *arc eager*-algoritmen (se Jurafsky & Martin, kapittel 18, seksjon 18.2.4, side 12). Angi tydelig hvilken algoritme du benytter i besvarelsen din. Forklar kort hvordan de to algoritmene skiller seg fra hverandre.

Oppgave 2 Trene modellen

I denne obligen skal vi bruke *spaCy*[2], 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 modell for norsk dependensparsing.

Som treningsdata har vi den norske bokmåls-trebanken fra *Universal Dependencies*[3]. Dataene kommer i *CoNLL-U-formatet*[4] og må konverteres til spaCy v3 sitt binære dokumentformat. Dette har vi allerede gjort for dere og disse filene (`no_bokmaal-ud-train.spacy` og `no_bokmaal-ud-dev.spacy`) distribueres sammen med pre-koden og kan brukes for å trene en modell.

Når vi trener en ny modell med spaCy trenger vi en konfigurasjonsfil `.cfg` som også ligger i mappen for obligen. Det meste her er det best å ikke røre, men man kan sette stiene for trenings- og dev-data i den filen. Modellen trenes fra kommandolinjen, med parametre for config-filen, for trenings- og valideringsdata, og mappen for lagring av de trente modellene.

```
$ conda activate in2110
$ spacy train <config-fil> --output <model-mappe> \
     --paths.train <treningsdata> --paths.dev <valideringsdata>
```

For `<model-mappe>` velger du et navn på en undermappe som vil bli opprettet under treningen. Kommandoen kan da som eksempel se ut som dette:

```
$ spacy train config.cfg --output ./output \
   --paths.train no_bokmaal-ud-train.spacy \
   --paths.dev no_bokmaal-ud-dev.spacy
```

spaCy lagrer den nyeste 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. Dette tar mange minutter. Treningen stopper når modellen ikke forbedres lenger. Mappen `model-best` inneholder modellen du har trent, og skal leveres som zip-fil sammen med resten av besvarelsen.

Hvis spacy train-commandoen feiler med beskjed om at det er noe feil med formatet på treningsdataene, kan man prøve å konvertere disse på nytt fra conll. Det kan gjøres med `python -m spacy convert`, og et eksempel er vist under,

```
$ python -m spacy convert no_bokmaal-ud-train.conllu .
```

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

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:

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

Disse kan vi konvertere til spaCy-objekter

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

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. dev_docs inneholder gull-dataene for dev-settet vårt. Siden vi skal teste parseren vi har trent, trenger vi en versjon der parseren har estimert depedensrelasjonene. Vi lager derfor en versjon av dev-settet uten labels:

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

Hvert av doc-objektene i `dev_docs_unlabeled` kan parses slik:

```
>>> doc = nb(doc)
```

der nb er modellen vi har lastet inn tidligere. Test det gjerne ut med koden under:

```
>>> doc = dev_docs_unlabeled[13]
>>> doc
I rene ord.
>>> doc.has_annotation("DEP")
False
>>> doc = nb(doc)
>>> doc.has_annotation("DEP")
True
```

I Notebooken for obligen skal du:

- Laste inn bokmål dev-filen som beskrevet over.
- Konvertere denne til spaCy-objekter
- Lage en versjon uten labels
- Lage en funksjon som tar inn en liste av spaCy dok-objekter og parser disse

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.