IN2110 – Obligatorisk innlevering 2b

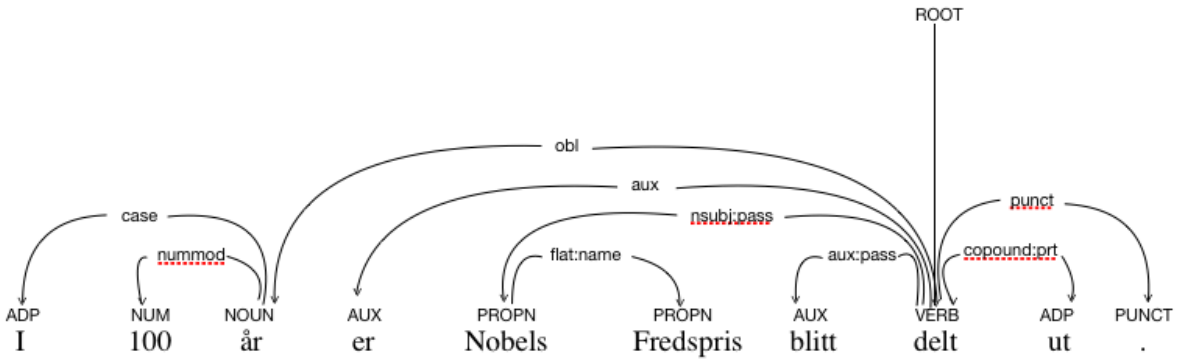
Joanna Ewa Korzeniowska (joannaek)

Adam Dariusz Osinski (adamdos)

Justyna Ozog (justynao)

Oppgave 1 Dependensgrammatikk

a) Dependenstrær



b) Transisjonsparsing – arc standard

Step	Stack	Word List	Action	Relation Added
0	[root]	[I, 100, år, er, Nobels, Fredspris, blitt, delt, ut, .]	shift	
1	[root, I]	[100, år, er, Nobels, Fredspris, blitt, delt, ut, .]	shift	
2	[root, I, 100)	[år, er, Nobels, Fredspris, blitt, delt, ut, .]	shift	
3	[root, I, 100, år]	[er, Nobels, Fredspris, blitt, delt, ut, .]	leftARC	(100<-år)
4	[root, I, år]	[er, Nobels, Fredspris, blitt, delt, ut, .]	leftARC	(I<-år)
5	[root, år]	[er, Nobels, Fredspris, blitt, delt, ut, .]	shift	
6	[root, år, er]	[Nobels, Fredspris, blitt, delt, ut, .]	shift	
7	[root, år, er, Nobels]	[Fredspris, blitt, delt, ut, .]	shift	
8	[root, år, er, Nobels, Fredspris]	[blitt, delt, ut, .]	rightARC	(Nobels->Fredspris)
9	[root, år, er, Nobels]	[blitt, delt, ut, .]	shift	
10	[root, år, er, Nobels, blitt]	[delt, ut, .]	shift	
11	[root, år, er, Nobels, blitt, delt]	[ut, .]	leftARC	(blitt<-delt)
12	[root, år, er, Nobels, delt]	[ut, .]	leftARC	(Nobels<-delt)
13	[root, år, er, delt]	[ut, .]	leftARC	(er<-delt)
14	[root, år, delt]	[ut, .]	leftARC	(år<-delt)
15	[root, delt]	[ut, .]	shift	
16	[root, delt, ut]	[.]	rightARC	(delt->ut)
17	[root, delt]	[.]	shift	
18	[root, delt, .]	[]	rightARC	(delt ->.)
19	[root, delt]	[]	rightARC	(root->delt)
20	[root]	[]	Done	

c) Transisjonsparsing – arc eager

Step	Stack	Word List	Action	Relation Added
0	[root]	[I, 100, år, er, Nobels, Fredspris, blitt, delt, ut, .]	shift	
1	[root, I]	[100, år, er, Nobels, Fredspris, blitt, delt, ut, .]	shift	
2	[root, I, 100]	[år, er, Nobels, Fredspris, blitt, delt, ut, .]	leftARC	(100<-år)
3	[root, I]	[år, er, Nobels, Fredspris, blitt, delt, ut, .]	leftARC	(I<-år)
4	[root]	[år, er, Nobels, Fredspris, blitt, delt, ut, .]	shift	
5	[root, år]	[er, Nobels, Fredspris, blitt, delt, ut, .]	shift	
6	[root, år, er]	[Nobels, Fredspris, blitt, delt, ut, .]	shift	
7	[root, år, er, Nobels]	[Fredspris, blitt, delt, ut, .]	rightARC	(Nobels->Fredspris)
8	[root, år, er, Nobels, Fredspris]	[blitt, delt, ut, .]	shift	
9	[root, år, er, Nobels, Fredspris, blitt]	[delt, ut, .]	leftARC	(blitt<-delt)
10	[root, år, er, Nobels, Fredspris]	[delt, ut, .]	reduce	
11	[root, år, er, Nobels]	[delt, ut, .]	leftARC	(Nobels<-delt)
12	[root, år, er]	[delt, ut, .]	leftARC	(er<-delt)
13	[root, år]	[delt, ut, .]	leftARC	(år<-delt)
14	[root]	[delt, ut, .]	rightARC	(root->delt)
15	[root, delt]	[ut, .]	rightARC	(delt->ut)
16	[root, delt, ut]	[.]	reduce	
17	[root, delt]	[.]	rightARC	(delt->.)
18	[root, delt, .]	[]	reduce	
19	[root, delt]	[]	reduce	
20	[root]	[]	done	

Oppgave 2 Trene modeller

Vi har brukt opsjonen **–n <n>** til **spacy train** for å begrense antall epochs. Vi har begrenset tallet til 5. Vi har brukt "my-model" som navnet til mappen til modellen vi trente opp med spacy train, med filen no_bokmaal-ud-train.json og filen no_bokmaal-ud-dev.json. Modeller ble lagret i "my-model" mappen.

Oppgave 3 Evaluering

a) POS tagging

Ytelsen til accuracy-funksjonen i 2b er i liten grad bedre, enn i HMM-taggeren fra oblig 2a.

Vi har lastet opp vår best model og konverterte setninger til spacy objekter.

Vi opprettet to lister og ved hjelp av en for løkke gikk vi gjennom listen av setninger (gold standard og predikerte) og lagret data (tag_ til hver ord) i listene. Hver liste er ei liste av setninger, og hver setning er ei liste av ord.

Accuracy metoden er kopiert fra forrige oblig, men vi rett og slett sammenligner gold standard tag med den predikerte tag.

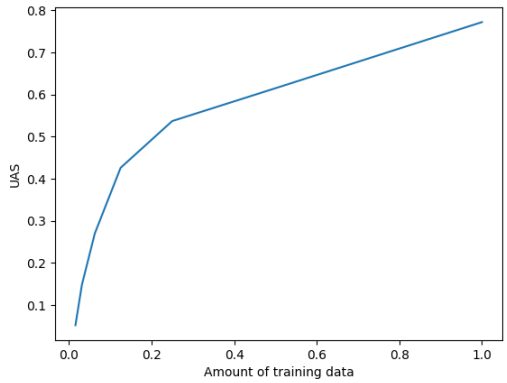
Som resultat fikk vi 0.9674997937804174, som er en ganske høy resultat. Accuracy til HMM taggen var noe rundt 0.91, som viser at POS-taggeren skårer litt bedre.

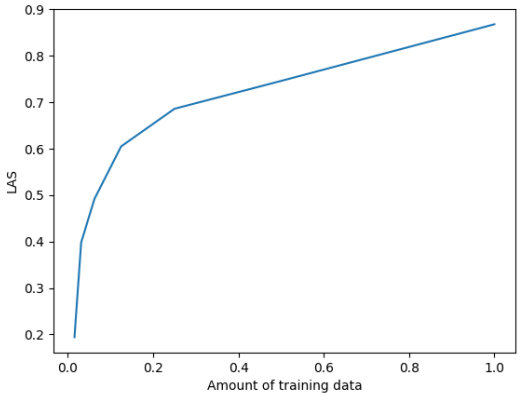
b) Parsing

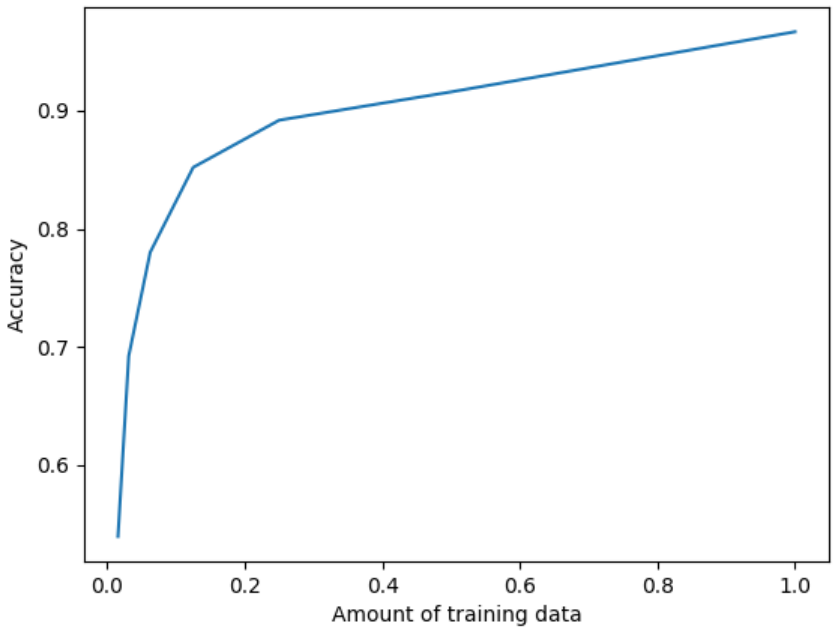
For å beregne UAS og LAS brukte vi ganske likt metode som for å beregne accuracy. Istedenfor å legge data i lister, har vi bare parset setningene ved hjelp av for løkken. I selve metoden oppretter vi tre variabler som tar vare på antall ord, antall ord med riktig head, og de med riktig dependens relasjon.

Så kjører vi for løkken som sammenligner et og et ord. Siden true[i][j].head ga oss token objekt som ikke kunne sammenlignes, så måtte vi ta .text parameteren av objektet.

c) Læringskurver



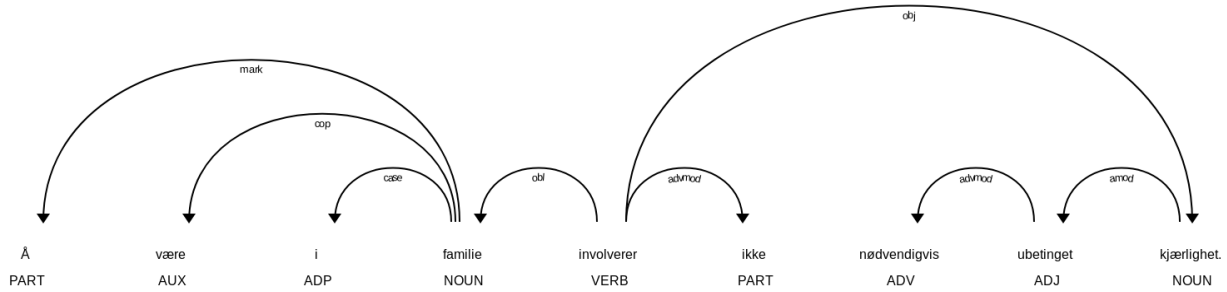




Vi ser en tydelig funksjon. Jo mer data, jo bedre resultat, men fra 25% av data og oppover (50, 100) øker resultatet lineart og gir ganske gode accuracy, UAS og LAS. Dobling av data gir mindre og mindre forskjeller mellom resultater på de forskjellige stegene.

Vi brukte **–n-eksempel <number sentences>** parameteren for å begrense mengde data. Vi har telt antall setninger i kjøring av programmet og brukt dette tallet i <number sentences> parameteren.

d) Data fra annet domene



Modellen gir ganske gode resultater, vi finner ikke noen feil. Vi har brukt en setning fra et innlegg på facebook. Visualisering viser at modellen i veldig stor grad klarer å gi en riktig dependens-tre.

Vi ser at visualisering ikke merker root, men det er ikke vanskelig å se at "involverer" er det.

I tillegg vises ikke "." som en egen node, men ignoreres på en måte.

Mer enn det klarer vi ikke å påpeke.