Løsningsforslag: Eksamen IN2110, vår 2020

Husk at et løsningsforslag nettopp er et *forslag*; det finnes som regel mer enn én god måte å løse ting på.

1 Representasjon av ord (20)

(a) Beskriv kort hva som menes med 'den distribusjonelle hypotesen'.

Svar: Den distribusjonelle hypotesen sier at semantikken (betydningen eller meningen) til et ord kan karakteriseres av hvordan ordet brukes (ordets 'distribusjon' i språket). Sagt på en annen måte; vi kan forvente at ord som forekommer i liknende kontekster gjerne også har liknende betydning. Det er altså en korrelasjon mellom distribusjonell likhet og semantisk likhet. Lingvister som Harris, Firth og flere diskuterte denne tanken allerede på 1950-tallet. En slik tilnærming til å analysere ords betydning kalles også gjerne for *distribusjonell semantikk*.

(b) Forklar kort (maksimalt en halv side) hvordan vi kan implementere den distribusjonelle hypotesen i en vektorrommodell.

Svar: Gitt dagens tilgang på store tekstmengder i digital form kan vi enkelt operasjonalisere den distribusjonelle hypotesen. For ordene vi ønsker å modellere vil vi registrere alle kontekstene vi observerer over et stort korpus. Dette kan gjøres f.eks ved å registrere trekk (*features*) som teller forekomster av andre ord i samme setninger som ordet vi ønsker å modellere (et eksempel på trekk som gjerne kalles *bag-of-words*). Etter å ha ekstrahert slike kontekstuelle trekk fra et stort korpus kan vi sammenlikne semantikken til ulike ord ved å sammenlikne trekkene deres. For gjøre dette trenger vi altså en måte å representere både ord og kontekster, samt en måte å sammnlikne dem på, og for dette brukes gjerne en vektorrommodell: Hvert ord representeres som en vektor og hver dimensjon tilsvarer et kontekstuelt trekk. For å finne ord som er distribusjonelt (og dermed semantisk) like hverandre kan vi måle avstanden (f.eks ved cosinus eller euklidsk avstand) mellom ordvektorene i rommet.

(c) Skisser kort noen mulige problemer eller utfordringer med å brukes en vektor-basert distribusjonell tilnærming for å representere ords betydning.

Svar: Ett probblem er at hvert ord i utgangspunktet kun er representert ved én vektor. Samtidig vet vi at flertydighet er svært utbredt i språket; veldig mange ord har mer enn én betydning. F.eks kan ordformen *rett* ha ulike ordklasser og bety flere ulike ting som 'måltid', 'rettighet', 'korrekt', handlingen 'å rette', med mer. I en distribusjonell tilnærming til semantikk som beskrevet over vil ulike betydninger av et ord slås sammen i én og samme representasjon, noe som introduserer en viss støy i modellen når vi ønsker å måle avstand mellom ulike ord. En annen utfordring er knyttet til såkalte antonymer, altså ord som har motsatt betydning, som f.eks kald/varm, høy/lav, død/levende, osv. Selv om slike ord representerer semantiske motsetninger, så vil de typisk ha veldig lik kontekstuell distribusjon og dermed ha vektorer som er nære hverandre i modellen.

(d) Én type vektorrepresentasjoner vi snakket om i forelesningene er såkalte *word embeddings*. Forklar kort (ett avsnitt) hva som kjennetegner disse.

Svar: Såkalte *word embeddings* er vektorer som sammenliknet med tradisjonelle ordvektorer har relativt *lav dimensjonalitet* (typisk noen få hundre dimensjoner eller mindre). Videre kan de karakteriseres som det vi på engelsk kaller *dense* (i motsetning til *sparse*), ved at de har få eller ingen ikke-null verdier. I motsetning til tradisjonelle ordvektorer der hver dimensjon tilsvarer ett diskré trekk, så er *embeddings* såkalt *distribuerte* (ikke bare distribusjonelle); en gitt dimensjon tilsvarer ikke lenger et gitt trekk, og informasjonen er 'spredt' over verdiene til alle dimensjonene. (Mens tradisjonelle ordvektorer gjerne kun er basert på frekvenstellinger i

et korpus så er *embeddings* gjerne laget via teknikker for dimensjonsreduksjon eller ved å estimere dem direkte gjennom nevrale modeller for klassifikasjon.)

(e) Her skal vi jobbe med pre-prosessering av tekst. Ta utgangspunkt i følgende tekst som eksempel:

The window washers washed the windows.

Forklar og vis med utgangspunkt i eksemplet over hva vi mener med tokenisering, lemmatisering og stemming. Ved telling av ordforekomster har vi gjerne skilt mellom to nivåer; *types* (ordtyper) og *tokens* (enkeltord). Oppgi det totale antall typer og tokens for hvert trinn; rå tekst, tokenisert, lemmatisert og til slutt med stemming. Kan du komme på andre former for normalisering av teksten som kan være aktuelle? Diskuter i så fall effekten av disse også.

Svar: Dersom vi tar som utgangspunkt at et gitt token er adskilt fra de andre med white-space og tellingene våre tar hensyn til stor/liten bokstav (*case*) så vil den 'rå' teksten ha *6 tokens* og *6 types*. Under følger tellinger for noen ulike varianter av pre-prosessering vi kan tenke oss.

Tokenisert; 7 tokens, 7 types:
The window washers washed the windows .

Med normalisering av stor/liten bokstav og fjerning av tegnsetting; 6 tokens, 5 types:
the window washers washed the windows

Med lemmatisering; 6 tokens, 4 types:
the window washer wash the window

Med stemming; 6 tokens, 3 types:
the window wash wash the window

Med stemming og fjerning av stopp-ord; 4 tokens, 2 types:
window wash wash window

2 Klassifikasjon (17)

(a) Forklar kort forskjellen på *klassifikasjon* og *klyngeanalyse* (*clustering*). Prøv å ta i bruk relevant fagterminologi. Forklar også kort hvilke fordeler og ulemper du ser med disse to ulike tilnærmingene?

Svar: Klassifikasjon og klyngeanalyse refererer til ulike familier av teknikker innenfor maskinlæring for å identifisere grupperinger i data. Mens klassifikasjon er et eksempel på det som kalles for *veiledet læring* ('supervised learning'), så er klyngeanalyse derimot et eksempel på *ikke-veiledet læring* ('unsupervised learning'). Fordelen med klyngeanalyse er dermed at man ikke trenger manuelt annoterte ('labeled') treningsdata i form av eksempler med forhåndsdefinert klassetilhørighet. I stedet finner algoritmen grupperinger (klynger) i data automatisk, med kun rå ('unlabeled') data som input. Ulempen er at man gjerne har mindre kontroll over hva slags (og hvor mange) grupperinger som identifiseres, og det kan også være vanskelig å evaluere resultatet. For veiledet klassifikasjon kreves derimot annoterte treningseksempler, noe som gjerne er kostbart å produsere og dermed begrenser omfanget, men til gjengjeld får vi gjerne mer kontroll over resultatet.

(b) Beskriv kort metodene logistisk regresjon og kNN for klassifikasjon. Sammenlikne egenskapene deres og beskriv kort deres fordeler og ulemper.

Svar: Både logistisk regresjon og kNN (k *nearest neighbors*) er eksempler på metoder for veiledet klassifikasjon. kNN er videre et eksempel på en gruppe teknikker som kalles *memory-based learning* eller *instance-based learning*. Metoden involverer strengt tatt ingen reell læring; i stedet lagres (eller memoriseres) bare alle eksemplene (instansene) i treningsdataene. For å klassifisere et nytt datapunkt måler man avstanden til alle treningseksemplene og så tilskriver man samme klasse som blant majoriteten av de k nærmeste naboene.

Ved logistisk regresjon derimot estimeres et sett av vekter – en vekt for hver trekktype. Ved hjelp av en optimeringsalgoritme (som f.eks *gradient descent*) settes vektene til verdier som skal minimere en såkalt tapsfunksjon som måler graden av feil i modellens prediksjoner over treningssettet. For å klassifisere et nytt datapunkt regner man ut en vektet sum av trekk som så gjøres om til en sannsynlighet ved hjelp av sigmoid-funksjonen.

Logistisk regresjon gir en svært kompakt modell med lav kompleksitet ved anvendelse. Vektene kan også brukes for å tolke det relative bidraget til ulike trekk i modellen. Den er derimot begrenset til lineært separerbare klasser. kNN er derimot en ikke-lineær modell. Selv om treningen av en logistisk regresjonsmodell allerede er relativt rask, er 'treningen' av en kNN-modell enda raskere. Prisen for dette betaler man når modellen anvendes siden man da må beregne avstanden mellom testeksempler og alle treningseksemplene, og man 'straffes' dermed for større treningssett. For logistisk regresjon er derimot testtiden konstant. Kompleksiteten til kNN er uavhengig av antall klasser. Logistisk regresjon kan også utvides til å håndtere mer enn to klasser, såkalt multinomial regresjon.

(For dette spørsmålet er det mye mer/annet som potensielt kunne vært sagt.)

(c) I emnet har vi snakket om ulike mål for å evaluere ytelsen til en klassifikator. To av målene vi har snakket om er *Precision* og *Recall*. Disse to målene er til en viss grad komplementære og i en del situasjoner vil det kunne finnes en avveining (*trade-off*) mellom å oppnå en høy verdi for enten *Precision* eller *Recall*. Forklar hva vi mener med dette. Beskriv også en vanlig måte for å kombinerer de to målene til ett mål.

Svar: Precision er definert som $P = \frac{TP}{TP+FP}$ og Recall er definert om $R = \frac{TP}{TP+FN}$ (der TP = *true positives*, FP = *false positives* og FN = *false negatives*). La oss som et illustrerende eksempel anta at vi jobber med binær klassifikasjon i form av et spam-filter (klassene er altså 'spam' og 'ikke-spam'). La oss si at vi klassifiserer all epost som 'spam'. Vi får da kun *true positives* og ingen *false negatives*, noe som gir maksimal utelling for Recall. Samtidig får vi såklart også et maksimalt antall *false positives* og dette føres til at Precision blir svært lav. Vi ser altså at det er en komlepentaritet i hvilke egeneskaper ved klassifikator som måles med henholdsvis R og P. Det er derfor vanlig å kombinere de to målene til ett i form av F1, definert ved $F1 = 2 \times \frac{P \times R}{P+R}$. (F1 er en instans av et mer generelt mål som kalles F-score der det er mulig å vekte det relative bidraget til R og P.)

3 Logistisk regresjon (15)

(a) Tren en logistisk regresjonsmodell på `train_df` (uten regularisering) som predikerer om wiki-siden handler om et insekt fra Ecuador basert på to numeriske trekk (*features*), nemlig antall tokens og antall lenker på siden. Hvilke *accuracy* oppnår du på testsettet `test_df`?

Svar:: Accuracy-scoren for den logistiske regresjonsmodellen er ganske lav – jeg har fått 0.518 ved å bruke den `LogisticRegression` klassifikatoren fra `scikit-learn` uten normalisering og LBFGS for optimering. Men det er godt mulig å oppnå litt forskjellige resultater avhengig av små detaljer som f.eks. optimeringsalgoritmen eller initialiseringsmetoden. Så så langt studentene får en accuracy som ligger mellom 0.50 og 0.53 kan de anses som riktig. Hvis du oppnår helt forskjellige resultater har de nok rotet med klassifikatoren eller med de to trekkene som skal brukes. Veldig bra hvis studentene også sammenligner dette med andre metoder (som f.eks k-nn, som klarer å komme til opptil 0.8) eller med en majority-class baseline, men det er ikke påkrevd.

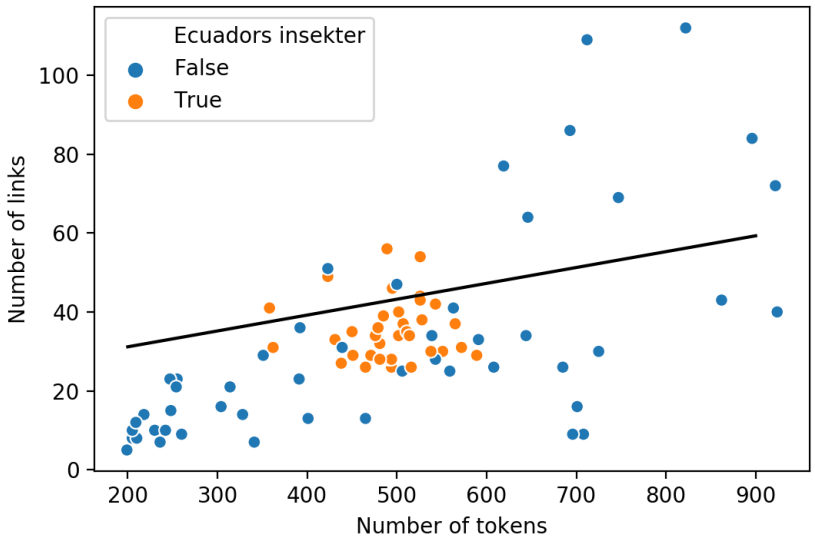
Poengfordeling: 5 hvis de får en riktig resultat for accuracy.

(b) Forklar hvorfor verdien du oppnådde for *accuracy* er så lav, basert på det du vet om logistisk regresjon. For å støtte din forklaring, tegn en *scatter plot* (du kan bruke funksjonen `scatterplot` i Seaborn hvor de to aksene står for de to numeriske trekkene, og fargen representerer outputklassen (insekt fra Ecuador eller ikke). Tegn deretter beslutningslinjen som er assosiert med den logistiske regresjonsmodellen du nettopp har trent.

Svar:: Grunnen til dette er ganske enkel, nemlig at de to klassene (insekt fra Ecuador eller ikke) er ikke lineært skillbar. Dette blir åpenbart hvis man tegne en scatter plot med datapunktene: vi ser at punktene om Ecuadors insekter er i midten av figuren, men de andre ligger både på venstre og høyre siden av denne gruppen. En logistisk regresjonsmodell er en lineær modell og vil derfor ikke være i stand til å skille de to klassene på en god måte. Fint om studenten også kan foreslå alternative modeller som er i stand til å lage ikke-lineære beslutningslinjer, men det er ikke nødvendig for å besvare spørsmålet.

Poengfordeling: 3 poenger hvis de klarer å tegne en scatterplot, 3 poenger hvis de også klarer å tegne beslutningslinjen på den, og 7 poenger hvis de kommer til riktig forklaring om hvorfor modellen ikke klarer å oppnå en god klassifisering (se over). Det er sikkert mange måte å utrykke dette, men jeg ønsker i det minste at de nevner at logistisk regresjon er en *lineær modell* og derfor ikke klarer å håndtere ikke-lineære skiller mellom klasser, slik som er tilfellet i dette eksempelet.

Scatterplot:



4 Sekvenslabellering (18)

(a) Gitt setningene i filen `norne.txt`, estimer de to transisjonssansynnlighetene $P(s_1 = \mathsf{NOUN}|s_0 = \mathsf{ADJ})$ og $P(s_1 = \mathsf{VERB}|s_0 = \mathsf{ADJ})$.

Svar: Ved å telle i korpuset kan vi finne at:

- $C(\mathsf{ADJ}) = 21215$
- $C(\mathsf{ADJ}, \mathsf{NOUN}) = 10018$
- $C(\mathsf{ADJ}, \mathsf{VERB}) = 773$

Det betyr altså at $P(s_1 = \mathsf{NOUN}|s_0 = \mathsf{ADJ}) = \frac{C(\mathsf{ADJ,NOUN})}{C(\mathsf{ADJ})} = 0.472$,
mens $P(s_1 = \mathsf{VERB}|s_0 = \mathsf{ADJ}) = \frac{C(\mathsf{ADJ,VERB})}{C(\mathsf{ADJ})} = 0.036$

Poengfordeling: 6 poenger (3 hvis de klarer å telle forekomstene riktig, og 3 hvis sluttresultaten er riktig)

(b) Gitt setningene i filen `norne.txt`, estimer de to emisjonsssansynnlighetene $P(o_1 = \textit{taler}|s_1 = \mathsf{NOUN})$ og $P(o_1 = \textit{taler}|s_1 = \mathsf{VERB})$. Dere kan ignorere smoothing.

Svar: Ved å telle i korpuset kan vi finne at:

- $C(\mathsf{NOUN}) = 45006$
- $C(\mathsf{VERB}) = 26154$
- $C(\mathsf{NOUN}, \mathsf{taler}) = 11$
- $C(\mathsf{VERB}, \mathsf{taler}) = 9$

Det betyr altså at $P(s_1 = \mathsf{NOUN}|s_0 = \mathsf{ADJ}) = \frac{\mathsf{NOUN,taler})}{C(\mathsf{NOUN})} = 0.00024$,
mens $P(s_1 = \mathsf{VERB}|s_0 = \mathsf{ADJ}) = \frac{C(\mathsf{VERB,taler})}{C(\mathsf{VERB})} = 0.00034$

Poengfordeling: 6 poenger (3 hvis de klarer å telle forekomstene riktig, og 3 hvis sluttresultaten er riktig)

(c) Gitt sansynnlighetene dere nettopp har beregnet, hva er den mest sannsynlige ordklassen for ordet "*taler*" som kommer etter "*flere*"?

Svar: Ordklassen NOUN er mest sansynnlig. Hvis vi ganger transisjon- og emisjonssansynnlighetene for NOUN får vi 0.000115, mens vi får 1.254e-05 for VERB. Siden spørsmålet spurte om en betinget sansynnlighet bør vi deretter normalisere resultaten, og det gir oss $P(s_1 = \mathsf{NOUN}|o_1 = \mathsf{taler}, s_0 = \mathsf{ADJ}) = 0.902$ og $P(s_1 = \mathsf{VERB}|o_1 = \mathsf{taler}, s_0 = \mathsf{ADJ}) = 0.098$.

NB: mer formelt kan man komme til samme svar ved å bruke Bayes:

$$
\begin{aligned}
P(s_1 = x|o_1 = \mathsf{taler}, s_0 = \mathsf{ADJ}) &= \frac{P(o_1 = \mathsf{taler}|s_0 = \mathsf{ADJ}, s_1 = x)P(s_1 = x|s_0 = \mathsf{ADJ})}{P(o_1 = \mathsf{taler}|s_0 = \mathsf{ADJ})} \\
&= \frac{P(o_1 = \mathsf{taler}|s_1 = x)P(s_1 = x|s_0 = \mathsf{ADJ})}{\sum_y P(o_1 = \mathsf{taler}, s_1 = y|s_0 = \mathsf{ADJ})} \\
&= \frac{P(o_1 = \mathsf{taler}|s_1 = x)P(s_1 = x|s_0 = \mathsf{ADJ})}{\sum_y P(o_1 = \mathsf{taler}|s_1 = y)P(s_1 = y|s_0 = \mathsf{ADJ})}
\end{aligned}
$$

Hvor X er enten NOUN eller VERB. Formelen over gir oss akkurat den samme resultaten, dvs. 0.902 for NOUN og 0.098 for VERB.

Men Bayes regelen var ikke del av pensumet (i det minste ikke i min del) så jeg forventer ikke at de skal kunne avlede formelen over. Det viktigste er de ganger transisjon- og emisjonssannsynlighetene.

Poengfordeling: 6 poenger (3 hvis de ganger transisjon- og emisjonssansynnlighetene, og 3 poenger hvis de konkluderer at NOUN er den mest sannsynlige ordklassen).

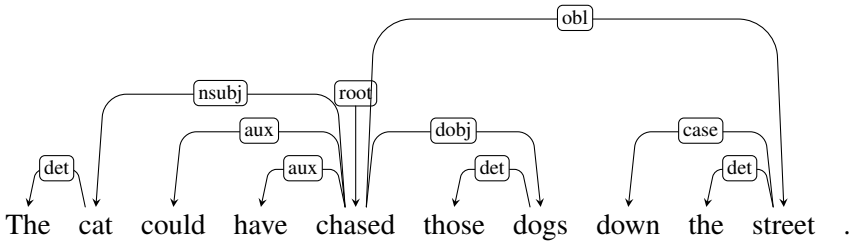
5 Dependenssyntaks og parsing (15)

Step	Stack	Word list	Action
0	[root]	[The, cat, could, have, chased, those, dogs, down, the, street]	SHIFT
1	[root, The]	[cat, could, have, chased, those, dogs, down, the, street]	LEFT-ARC$_{det}$
2	[root]	[cat, could, have, chased, those, dogs, down, the, street]	SHIFT
3	[root, cat]	[could, have, chased, those, dogs, down, the, street]	SHIFT
4	[root, cat, could]	[have, chased, those, dogs, down, the, street]	SHIFT
5	[root, cat, could, have]	[chased, those, dogs, down, the, street]	LEFT-ARC$_{aux}$
6	[root, cat, could]	[chased, those, dogs, down, the, street]	LEFT-ARC$_{aux}$
7	[root, cat]	[chased, those, dogs, down, the, street]	LEFT-ARC$_{nsubj}$
8	[root]	[chased, those, dogs, down, the, street]	RIGHT-ARC$_{root}$
9	[root, chased]	[those, dogs, down, the, street]	SHIFT
10	[root, chased, those]	[dogs, down, the, street]	LEFT-ARC$_{det}$
11	[root, chased]	[dogs, down, the, street]	RIGHT-ARC$_{dobj}$
12	[root, chased, dogs]	[down, the, street]	SHIFT
13	[root, chased, dogs, down]	[the, street]	SHIFT
14	[root, chased, dogs, down, the]	[street]	LEFT-ARC$_{det}$
15	[root, chased, dogs, down]	[street]	LEFT-ARC$_{case}$
15	[root, chased, dogs]	[street]	REDUCE
16	[root, chased]	[street]	RIGHT-ARC$_{obl}$
17	[root, chased, street]	[]	REDUCE
18	[root, chased]	[]	REDUCE
19	[root]	[]	DONE

(a) I tabellen over ser du en transisjonssekvens for en dependensparser.

(i) Tegn dependensgrafen som parseren produserer.

Svar: Parseren produserer dependensgrafen under:



(ii) Hvilken parsingsalgoritme (arc standard eller arc eager) er brukt her? Begrunn svaret ditt.

Svar: Parsingsalgoritmen er *arc eager*-algoritmen. Det ser vi ved at REDUCE-transisjonen anvendes og at RIGHT-ARC-transisjonen ikke fjerner dependenten når den anvendes. Dette gjør at høyrekanter kan legges til raskere og at parsingen er mer effektiv.

(b) Velg deg ut tre formelle kriterier som ofte stilles til dependensgrafer og gi en kortfattet forklaring for hver av dem. Vis deretter hvordan du kan endre dependensgrafen fra (a) slik at den bryter minst to av disse. Du kan her velge om du vil tegne de modifiserte grafene eller forklare endringene med ord.

Svar: Her kan man eksempelvis ta for seg noen av følgende:

- Connectedness: alle noder er festet til grafen, enhver node i har en annen node j slik at i->j eller j->i. Sterk variant: alle noder kan nås fra rotnoden. (Begge disse forklaringene er riktige).
- Acyclicity: ingen sykler i grafen, dvs at det ikke skal være mulig å komme tilbake til en node dersom man følger en sti fra den noden.

- Single-headedness: enhver node har kun ett hode
- Projectivity: ingen kryssende kanter, eller mer formelt en kant A->B er projektiv dersom alle noder mellom A og B er underordnet A.
- Andre gyldige svar er egenskapene anti-symmetrisk og anti-transitiv.

Grafen over vil bryte med prinsippene dersom:

- Connectedness: man sletter en kant, feks kanten fra cat->The
- Acyclicity: man legger til en kant fra The -> chased
- Single-head: legger til en kant fra could->cat

(c) Hva slags trekk bruker man typisk for klassifisering av transisjoner i dependensparsing? Illustrer svaret ditt ved å gi eksempler på minst tre typer trekk sammen med verdiene de vil få i steg 12 i transisjonssekvensen over.

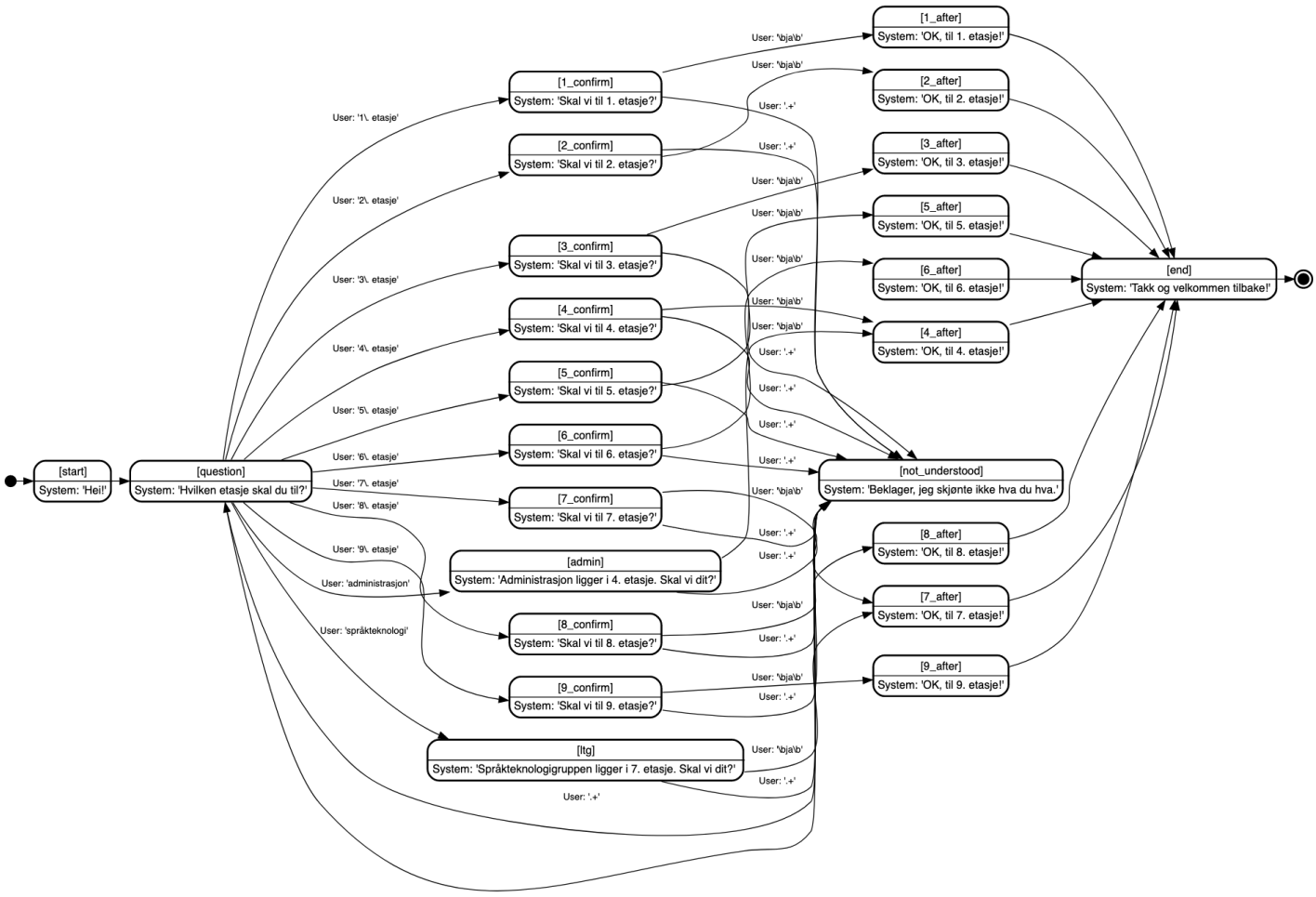
Svar: I dependensparsing definerer man trekk over konfigurasjoner, dvs egenskaper ved tuppelen <stack, buffer, arcs> ved hvert steg i parsing. Her kan man se på ulike egenskaper ved ordene og kantene som er produsert, feks ordform, lemma, ordklasse, dependensrelasjon og tidligere transisjoner.
Eksempler fra steg 12:

- word(S0) = dogs
- word(S1) = chased
- word(B0) = down
- word(B1) = the
- pos(S0) = noun
- dep(S0) = dobj
- dep(lc(S0)) = det

6 Interaktive systemer (15)

Svaret på dette spørsmålet må inneholde (a) PNG-filen som gjengir automaten deres sammen med (b) en kort forklaring (maksimalt en halv-side) av designvalgene deres i utviklingen av automaten. Du trenger ikke å levere noen kode.

Svar: Her er det vanskelig å komme med noen fasit, da det finnes mange ulike måter å utforme slike automater. Figuren 1 viser et eksempel av automat for denne oppgaven:



Figur 1: Eksempel av automat

Vi bør sjekke at automaten klarer å handtere dialoger som disse:

1. Hvilken etasje skal du til?
2. 4. etasje
3. Skal vi til 4. etasje?
4. Ja
5. (beveger seg til 4. etasje) Takk og velkommen tilbake!

1. Hvilken etasje skal du til?
2. vet ikke helt, hva foreslår du?
3. Beklager, jeg forsto ikke hva du sa. Kan du gjenta?
4. jeg skal til 3. etasje
5. ja
6. (beveger seg til 3. etasje) Takk og velkommen tilbake!

1. Hvilken etasje skal du til?
2. jeg ønsker å komme til 4. etasje
3. Skal vi til 4. etasje?
4. nei vent beklager 3. etasje
5. Beklager, jeg forsto ikke hva du sa. Kan du gjenta?
6. jeg skal til 3. etasje
7. ja
8. (beveger seg til 3. etasje) Takk og velkommen tilbake!

1. Hvilken etasje skal du til?
2. I hvilken etasje er språkteknologigruppen?
3. Språkteknologigruppen er i 7. etasje. Skal vi dit?
4. ja
5. (beveger seg til 7. etasje) Takk og velkommen tilbake!

1. Hvilken etasje skal du til?
2. I hvilken etasje er språkteknologigruppen?
3. Språkteknologigruppen er i 7. etasje. Skal vi dit?
4. nei egentlig må jeg først hente mat
5. Beklager, jeg forsto ikke hva du sa. Kan du gjenta?

NB: Delen i parentesen (beveger seg til ...) er valgfri.

Poengfordeling: 5 poenger hvis de har klart å bygge automaten og få en visuel gjengivelse av automaten. 5 poenger hvis automaten også klarer å handtere alle dialogene over. 5 poenger hvis de har skrevet en kort forklaring om automaten og har reflektert over hvordan den fungerer (og eventuelt hva automaten ikke klarer å handtere).