Løsningsforslag til eksamen i IN2110, vår 2021

Husk på at løsningsforslagene som står her er nettopp det: forslag. Det finnes typisk flere måter å gjøre ting på.

1 Kategorier (8 poeng)

1.1 K-means og Rocchio (4 poeng)

- Mens Rocchio er et eksempel på en algoritme for klassifikasjon basert på veiledet læring, så er K-means et eksempel på klyngeanalyse og basert på ikke-veiledet læring.
- Begge metodene representerer en gitt klasse (eller klynge) ved dens centroide, altså en vektor som beregnes som gjennomsnittet av trekkvektorene til alle medlemenne av klassen.
- Rocchio fungerer ved å beregne centroidene til alle klassene basert på treningseksemplene, for så å klassifisere nye instanser ut i fra nærmeste centroide.
- K-means fungerer ved å iterativt beregne centroider og så tilskrive medlemskap basert på centroide-avstand frem til et stoppekriterie er nådd,
- og i praksis kan vi si at altså si at den gjør Rocchio-klassifikasjon i hver iterasjon.

1.2 Veiledet læring (4 poeng)

- Læring fra annoterte/merkete eksempler, altså eksempler der riktig klasse er forhåndsdefinert
- Eksempler på algoritmer og anvendelser KNN, Rocchio, logistisk regresjon (tekst-klassifikasjon), HMM (NER), maskinoversettelse, og dependensparsing.

2 Ord og dokumenter (12 poeng)

2.1 Bag-of-words (3 poeng)

- BoW: Hver ordtype i vokabulæret tilsvarer en dimensjon, og verdien i en gitt trekkvektor er gitt ved antall forekomster av ordet i dokumentet (muligens vektet).
- Svakheter:
- Ignorerer rekkefølge og struktur og kan dermed ikke ta høyde for f.eks komposisjonalitet.
- Det er heller ingen informasjonsdeling mellom trekk, slik at trekket som koder antall forekomster av f.eks. “kaffe” ikke er mer assosiert med “espresso” enn med “katt”. (Gi et poeng for forklaring; det trenger ikke være akkurat disse eksemplene så klart.)

2.2 TF-IDF (3 poeng)

- Ord som forekommer relativt ofte i et gitt dokument men mer sjelden i dokumentsamlingen som helhet vil bli gitt en høyere vekt, mens ord som forekommer i mange dokumenter vil bli vektet ned.
- Merk at det vil finnes mange ulike måter å foklare dette på, og det er greit.

2.3 Embeddings (3 poeng)

- Tradisjonelle ordvektorer er gjerne høy-dimensjonale, er såkalt "sparse", dvs. at de har få aktive (ikke-null) trekkverdier, og en gitt dimensjon / trekk tilsvarer en diskré egenskap (f.eks antall forkomster i nærheten av ordet "bake").
- Embeddings har typisk lavere dimensjonalitet, er såkalt "dense", dvs. at de har mange eller kun aktive trekk (altså verdier != 0), og informasjonen er distribuert, dvs. at et gitt trekk ikke tilsvarer en gitt egenskap ved et ord, i stedet er informasjonen kodet i totaliteten av trekk-aktivering i hele vektoren.

2.4 Antonymer (3 poeng)

- Antonymer er ordpar med motsatt betydning, f.eks kald/varm, opp/ned.
- Selv om de har motsettende betydning så brukes de gjerne på samme måte og i like kontekster og de vil dermed kunne fremstå som synonymer eller semantisk like i henhold til en distribusjonell tilnærming til semantikk.

3 Evaluering (12 poeng)

3.1 Om rett og galt (2 poeng)

- TP: Tilfeller som blir markert som positive (altså medlemmer av en gitt klasse) av både klassifikatoren og gull-standard annotasjonen.
- FP: Tilfeller som blir markert som positive av klassifikatoren men negative (altså ikke medlemmer av den gitte klassen) i gull-standard annotasjonen.
- TN: Tilfeller som blir markert som negative av både klassifikatoren og gull-standarden.
- FN: Tilfeller som blir markert som negative av klassifikatoren men positive i gull-standarden.

3.2 Evalueringsmål (8 poeng)

Formler:

- $P = \frac{TP}{TP+FP}$
- $R = \frac{TP}{TP+FN}$
- $F_1 = 2 \times \frac{P \times R}{P+R}$
- $Acc = \frac{TP+TN}{N}$, der $N = TP + FP + TN + FN$

Eksempler på forklaringer:

- P: andelen av positive prediksjoner som er riktige
- R: andelen av faktisk positive eksempler ble identifisert
- F1: kombinerer P og R i ett balansert mål ("harmonisk gjennomsnitt")
- Acc: andelen riktige prediksjoner

3.3 En tenkt klassifikator (6 poeng)

- TP = 40
- FP = 80
- TN = 870
- FN = 10
- P = TP/TP+FP = 40/40+80 = 0.3333
- R = TP/TP+FN = 40/40+10 = 0.8000
- F = 2 * ((0.3333 * 0.8000) / (0.3333 + 0.8000)) = .4706
- A = 40 + 870 / 1000 = 0.9100

3.4 Valg av mål (4 poeng)

- Accuracy er kjent for å være lite informativt når vi har ubalanserte klasser slik som her. En enkel 'baseline' basert på å gi majoritetsklassen ('nøytral') til alle kommentarer vil her gi en svært høy accuracy på 950/1000 = 0.95. I slike tilfeller vil F1 være mer egnet enn accuracy.
- Eksemplet illustrerer også at det ofte er en avveining melllom precision og recall; i dette tilfellet har modellen relativt høy recall men relativt lav precision, og det kan dermed være informativt å rapportere precision og recall individuelt i tillegg til F1.

4 Logistisk regresjon (10 poeng)

4.1 Trekkene (2 poeng)

Svaret her er 8 eller 9 trekk, avhengig om man teller "foran" som et stopword eller ikke. Ordene som vil fungere som bag-of-words trekk er: butikken, åpen, parkere, selger, vesker, Gucci, åpningstid, parkeringsplass (og muligens "foran").

4.2 Parametre (2 poeng)

Vi har 3 ulike klasser, så for hver klasser har vi 8 eller 9 vekter (en per trekk, se se forrige svar) + 1 (skjæringspunkt). Totalt sett har vi altså 27 eller 30 parametre, igjen avhengig om antall ordene som brukes som bag-of-words trekk.

4.3 Parameterverdier (3 poeng)

(Her er det mange ulike løsninger.)

La oss først definere som notasjon w_[word]_[klasse] vekten som er assosiert med forekomsten av "word" i ytringen for en av de tre klassene (SpørOmÅpningstid, SpørOmParkering, SpørOmGucciVesker).

Da kan vi for eksempel sette som vekter:s

- w_selger_SpørOmGucciVesker = 1
- w_gucci_SpørOmGucciVesker = 1
- w_vesker_SpørOmGucciVesker = 1
- w_åpningstid_SpørOmÅpningstid = 1
- w_åpen_SpørOmÅpningstid = 1

- w_parkere_SpørOmParkering = 1
- w_parkeringsplass_SpørOmParkering = 1

og alle andre vektene (og skjæringspunktene) til 0. Det er mange alternative løsninger (for eksempel kunne man ha en annen verdi enn 1), men det viktigste er at løsningen faktisk gir perfekt klassifisering på treningsett.

4.4 Analyse (3 poeng)

Det er mange problemer med modellen som ble bygd her, men den kanskje viktigste er at den er veldig lite robust, siden den baserer seg kun på forekomsten av noen fåtall ord. Ytringer som inneholder andre lignende ord eller morfologiske varianter (for eksempel "parkering", "veskene", "sekk", osv.) vil ikke fungere i det hele tatt.

Besvarelsen bør nevne minst én mulig løsning for å forbedre modellen, som f.eks.:

- få takk i et større treningsett, slik at modellen kan dekke flere ord
- bruke ordvektorer (da helst med en hidden layer før den softmax) i sted for bag-of-words, slik at modellen kan handtere ord som er semantisk nær hverandre
- bruke lemmatisering slik at man ikke trenger å bekymre seg for morfologiske varianter
- bruke "data augmentation" til å lage alternative ytringer basert på de som finnes allerede i treningsett
- osv.

5 Sekvensmodeller (10 poeng)

5.1 Viterbi (10 poeng)

Vi anvender Viterbi-algoritmen skritt for skritt (altså bokstav etter bokstav). Sannsynlighet for de fleste tiltenkte bokstavene er heldvigvis 0. Vi skriver her bare sannsynlighetene som er > 0.

Bokstav 1 (observasjon: "s"):

$$\begin{aligned} v_1(s) &= P_{emit}(s \mid s) * P_{trans}(s \mid <start>) \\ &= 0.85 * 0.11 = 0.0935 \end{aligned}$$

$$\begin{aligned} v_1(a) &= P_{emit}(s \mid a) * P_{trans}(a \mid <start>) \\ &= 0.05 * 0.05 \ = 0.0025 \end{aligned}$$

$$\begin{aligned} v_1(d) &= P_{emit}(s \mid d) * P_{trans}(d \mid <start>) \\ &= 0.05 * 0.08 = 0.004 \end{aligned}$$

Bokstav 2 (observasjon "p"):

$$\begin{aligned} v_2(p) = P_{emit}(p \mid p) * \max(&v_1(a) * P_{trans}(p \mid a), \\ &v_1(d) * P_{trans}(p \mid d), \\ &v_1(s) * P_{trans}(p \mid s), \\ &v_1(w) * P_{trans}(p \mid w)) \end{aligned}$$

$$\begin{aligned} &= 0.85 * \max(5 \times 10^{-05}, 0.0, 0.002805, 0.0) \\ &= 0.00238425 \end{aligned}$$

$$v_2(o) = P_{emit}(p \mid o) * \max(v_1(a) * P_{trans}(o \mid a), \\ v_1(d) * P_{trans}(o \mid d), \\ v_1(s) * P_{trans}(o \mid s), \\ v_1(w) * P_{trans}(o \mid w))$$

$$= 0.05 * \max(0.0, 8 \times 10^{-05}, 0.008415, 0.0) \\ = 0.00042075$$

$$v_2(\text{ø}) = P_{emit}(p \mid \text{ø}) * \max(v_1(a) * P_{trans}(\text{ø} \mid a), \\ v_1(d) * P_{trans}(\text{ø} \mid d), \\ v_1(s) * P_{trans}(\text{ø} \mid s), \\ v_1(w) * P_{trans}(\text{ø} \mid w))$$

$$= 0.05 * \max(0.0, 0.0, 0.000935, 0.0) \\ = 4.675 \times 10^{-05}$$

$$v_2(\text{å}) = P_{emit}(p \mid \text{å}) * \max(v_1(a) * P_{trans}(\text{å} \mid a), \\ v_1(d) * P_{trans}(\text{å} \mid d), \\ v_1(s) * P_{trans}(\text{å} \mid s), \\ v_1(w) * P_{trans}(\text{å} \mid w))$$

$$= 0.05 * \max(0.0, 4 \times 10^{-05}, 0.00187, 0.0) \\ = 9.35 \times 10^{-05}$$

Bokstav 3 (observasjon "r"):

$$v_3(r) = P_{emit}(r \mid r) * \max(v_2(o) * P_{trans}(r \mid o), \\ v_2(p) * P_{trans}(r \mid p), \\ v_2(\text{ø}) * P_{trans}(r \mid \text{ø}), \\ v_2(\text{å}) * P_{trans}(r \mid \text{å}))$$

$$= 0.85 * \max(0.000109395, 0.00028611, 2.38425 \times 10^{-05}, 3.8335 \times 10^{-05}) \\ = 0.0002431935$$

$$v_3(e) = P_{emit}(r \mid e) * \max(v_2(o) * P_{trans}(e \mid o), \\ v_2(p) * P_{trans}(e \mid p), \\ v_2(\text{ø}) * P_{trans}(e \mid \text{ø}), \\ v_2(\text{å}) * P_{trans}(e \mid \text{å}))$$

$$= 0.05 * \max(8.415 \times 10^{-06}, 0.000429165, 4.675 \times 10^{-07}, 1.87 \times 10^{-06}) \\ = 2.145825 \times 10^{-05}$$

$$v_3(f) = P_{emit}(r \mid f) * \max(v_2(o) * P_{trans}(f \mid o), \\ v_2(p) * P_{trans}(f \mid p), \\ v_2(\text{ø}) * P_{trans}(f \mid \text{ø}), \\ v_2(\text{å}) * P_{trans}(f \mid \text{å}))$$

$$= 0.05 * \max(4.2075 \times 10^{-06}, 2.38425 \times 10^{-05}, 4.675 \times 10^{-07}, 9.35 \times 10^{-07}) \\ = 1.192125 \times 10^{-06}$$

$$v_3(t) = P_{emit}(r \mid t) * \max(v_2(o) * P_{trans}(t \mid o), \\ v_2(p) * P_{trans}(t \mid p), \\ v_2(\text{ø}) * P_{trans}(t \mid \text{ø}), \\ v_2(\text{å}) * P_{trans}(t \mid \text{å}))$$

$$= 0.05 * \max(1.2622 \times 10^{-05}, 4.7685 \times 10^{-05}, 1.87 \times 10^{-06}, 7.48 \times 10^{-06}) \\ = 2.38425 \times 10^{-06}$$

Bokstav 4 (observasjon "å"):

$$v_4(\text{å}) = P_{emit}(\text{å} \mid \text{å}) * \max(v_3(e) * P_{trans}(\text{å} \mid e), \\ v_3(f) * P_{trans}(\text{å} \mid f), \\ v_3(r) * P_{trans}(\text{å} \mid r), \\ v_3(t) * P_{trans}(\text{å} \mid t))$$

$$= 0.9 * \max(0.0, 7.15275 \times 10^{-08}, 2.431935 \times 10^{-06}, 2.38425 \times 10^{-08}) \\ = 2.1887415 \times 10^{-06}$$

$$v_4(p) = P_{emit}(\text{å} \mid p) * \max(v_3(e) * P_{trans}(p \mid e), \\ v_3(f) * P_{trans}(p \mid f), \\ v_3(r) * P_{trans}(p \mid r), \\ v_3(t) * P_{trans}(p \mid t))$$

$$= 0.05 * \max(2.145825 \times 10^{-07}, 0.0, 0.0, 0.0) \\ = 1.0729125 \times 10^{-08}$$

$$v_4(\text{æ}) = P_{emit}(\text{å} \mid \text{æ}) * \max(v_3(e) * P_{trans}(\text{æ} \mid e), \\ v_3(f) * P_{trans}(\text{æ} \mid f), \\ v_3(r) * P_{trans}(\text{æ} \mid r), \\ v_3(t) * P_{trans}(\text{æ} \mid t))$$

$$= 0.05 * \max(0.0, 2.38425 \times 10^{-08}, 2.431935 \times 10^{-06}, 2.38425 \times 10^{-08}) \\ = 1.2159675 \times 10^{-07}$$

Bokstav 5 (observasjon "i"):

$$v_5(i) = P_{emit}(i \mid i) * \max(v_4(p) * P_{trans}(i \mid p), \\ v_4(æ) * P_{trans}(i \mid æ), \\ v_4(å) * P_{trans}(i \mid å))$$

$$= 0.85 * \max(5.3645625 \times 10^{-10}, 0.0, 0.0) \\ = 4.559878125 \times 10^{-10}$$

$$v_5(k) = P_{emit}(i \mid k) * \max(v_4(p) * P_{trans}(k \mid p), \\ v_4(æ) * P_{trans}(k \mid æ), \\ v_4(å) * P_{trans}(k \mid å))$$

$$= 0.05 * \max(0.0, 3.6479025 \times 10^{-09}, 1.3132449 \times 10^{-07}) \\ = 6.5662245 \times 10^{-09}$$

$$v_5(o) = P_{emit}(i \mid o) * \max(v_4(p) * P_{trans}(o \mid p), \\ v_4(æ) * P_{trans}(o \mid æ), \\ v_4(å) * P_{trans}(o \mid å))$$

$$= 0.05 * \max(7.5103875 \times 10^{-10}, 0.0, 0.0) \\ = 3.75519375 \times 10^{-11}$$

$$v_5(u) = P_{emit}(i \mid u) * \max(v_4(p) * P_{trans}(u \mid p), \\ v_4(æ) * P_{trans}(u \mid æ), \\ v_4(å) * P_{trans}(u \mid å))$$

$$= 0.05 * \max(2.145825 \times 10^{-10}, 0.0, 0.0) = 1.0729125 \times 10^{-11}$$

Det betyr at det mest sannsynlige bokstavet ved slutten er "k". Da kan vi følge den motsatte veien gjennom bak-pointers for å finne at bokstavet som kommer før er "å", og deretter "r", "p", og til slutt "s". Og vi kommer til løsningen, nemlig at ordet blir rettet til "språk".

6 Dependenssyntaks og dependensparsing (20 poeng)

6.1 Dependenssyntaks (10 poeng)

1. Universal Dependencies:

   - Universal Dependencies er et dugnadsbasert initiativ der målet er å samle trebanker for så mange språk som mulig i et felles format for dependensrepresentasjoner, der analysevalg er gjort med mål om å være universelt gyldige, dvs kunne implementeres for alle språk.
   - Graf B angir en UD-analyse av setningen. Dette valget er basert på både hodevalg i grafen og de syntaktiske relasjonene. Når det gjelder hodevalg, går UD langt i å velge innholdsord (substantiver og verb) som hoder framfor mer syntaktiske elementer som hjelpeverb, infinitivsmerker, subjunksjoner og preposisjoner. Feks er preposisjonen på"i eksempelet dependens til det nominale hodet bordet"(av typen "case"), noe som er motivert ved at mange språk jo ikke har pre/postposisjoner, men snarere kasusmarkering. Vi ser fokuset på innholdsord som hode også i valget av prøvesom rot, framfor hjelpeverbet skalsom er det finitte verbet som markerer tid.

2. Ja, begge grafene er projektive. Det ser vi ved at ingen kanter i grafen krysser hverandre. Mer formelt kan vi si at grafen er projektivt fordi det stemmer at dersom i -> j, så vil enhver k slik at i < k < j eller j < k < i, også være (transitivt) underordnet i.

6.2 Dependensparsing (10 poeng)

```
0 [root] [Han skal prøve å danse på bordet] SHIFT
1 [root Han] [skal prøve å danse på bordet] LEFT-ARC(SUBJ) (Han <- skal)
2 [root] [skal prøve å danse på bordet] RIGHT-ARC(ROOT) (root -> skal)
3 [root skal] [prøve å danse på bordet] RIGHT-ARC(INFV) (skal -> prøve)
4 [root skal prøve] [å danse på bordet] RIGHT-ARC(DOBJ) (prøve -> å)
5 [root skal prøve å] [danse på bordet] RIGHT-ARC(INFV) (å -> danse)
6 [root skal prøve å danse] [på bordet] RIGHT-ARC(ADV) (danse -> på)
7 [root skal prøve å danse på] [bordet] RIGHT-ARC(PUTFYLL) (på -> bordet)
8 [root skal prøve å danse på bordet] [] REDUCE
9 [root skal prøve å danse på] [] REDUCE
10 [root skal prøve å danse] [] REDUCE
11 [root skal prøve å] [] REDUCE
12 [root skal prøve] [] REDUCE
13 [root skal] [] REDUCE
14 [root] [] DONE
```

7 Maskinoversettelse (10 poeng)

7.1 Morfologi (3 poeng)

Språk med rik morfologi (f.eks. ved agglutinerende eller flekterende språk) har ofte mange mulike kombinasjoner (bøyninger, sammensetting av ord osv) som kan brukes til å lage et ord. Det betyr at en maskinoversettelsemodell som har et slikt språk som kilde eller mål må kunne håndtere et stort vokabular som inkluderer slike morfologiske varianter.

Dette er problematisk, da mange av disse morfologiske variantene kommer til å ha få forekomster i parallelle korpora. Og hvis et ord aldri observeres eller forekommer kun noen få ganger i korpuset vil oversettelsemodellen ikke klare å lære gode oversettelsemønstre.

En løsning er å jobbe med orddeler (sub-word units) istedet for fullstendige ord.

7.2 BLEU (3 poeng)

BLEU er basert på å beregne presisjonen på N-grams mellom oversettelsen som er produsert av systemet og en eller flere referanseoversettelser skrevet av menneskelige oversettere. Vanligvis bruker BLEU et geometrisk gjennomsnitt mellom presisjoner for N-grams med N fra 1 (unigrams) til 4 (quadrigrams).

Språk som har en fri ordstilling vil føre til problemer med de høyere N-grams (> 1) hvis ordstillingen i den systemproduserte oversettelsen er annerledes enn i fasiten(e). Vi vil nemlig få mange uoverenstemmelser mellom N-grams fra systemet og Ngrams fra fasitene, og dermed en lav presisjon og en lav BLEU-verdi, selv når oversettelsen er riktig.

7.3 Decoding (4 poeng)

I maskinoversettelse handler dekoding om å generere outputsetningen i målspråket, ord for ord (eller orddel for orddel hvis man jobber med sub-word units, eller bokstav for bokstav hvis man jobber med character-based MT). Dekoding starter rett etter encoding av kildesetningen, og tar vektorene fra encoding som input. Dekoding starter

med å predikere det første ordet (eller orddel) i målsetningen gitt inputvektorene, og forsetter med de andre ordene til man kommer til en spesiell sluttsymbol som indikerer at dekoding er ferdig. Prediksjonen av det neste ordet er avhengig av både vektorene fra kildesetningen og ordene som allerede har blitt produsert.

Greedy search er den enkleste søkemekanismen i dekoding. Da velger man kun ordet (eller orddelen) som har den høyeste sannsynligheten og ignorerer resten. Fordelen av greedy search er at den er rask og enkel, men man risikerer at et ord valgt på trinn t fører til problemer med prediksjonene på trinn t+1, t+2 osv. Da ender man med en suboptimal løsning fordi man har valgt feil ord og ikke kan gå bakover for å rette den. Greedy search er begrenset til lokale avgjørelser (valg av ett ord av gangen), men en god oversettelse handler ikke bare om lokale valg - vi må se hele oversettelsen i sammenheng!

Beam search er en måte å løse denne utfordringen. I stedet for å kun jobbe med én oversettelsehypotese jobber vi med en flere alternative oversettelsehypoteser. Siden antall mulige hypoteser vokser eksponentielt med setningslengden begrenser vi antall hypoteser med et maksimum antall k (="beam size") alternativer . Beam search vurderer kun de k beste hypotesene (basert på den samlede sannsynligheten for sekvensen så langt) på hvert trinn. Beam search krever flere beregninger enn greedy search, men kan gi oversettelser av bedre kvalitet.

Exaustive search er hva som skjer når antall alternativer ikke er begrenset av et maskimalt antall alternativer. Da kan man teoretisk sett få den aller "beste" oversettelsen - men det er kun teoretisk, siden antall hypoteser vokser eksponentielt med setningslengden. Med andre ord er exhaustive search intractable.

8 Interaktive systemer (10 poeng)

8.1 Grounding (3 poeng)

"Grounding" handler om kommunikative signaler som deltakere i en samtale sender hverandre for å forsikre seg at de er "på samme linje". Blant annet inneholder menneskelige samtaler en rekke signaler for å formidle at et bidrag har blitt oppfattet eller forstått. Disse signalene kan være eksplisitte (slik som backchannels) eller implisitte (gjentakelse av hverandres ord eller utrykk). Grounding-signaler inkluderer også strategier for å formidle at noe ikke har blitt forstått (hæ?) og for å komme tilbake til en samtalesituasjon med gjensidig forståelse (repair strategies). Ved hjelp av disse signalene fører en samtale til en økning av deltakeres "common ground", altså kunnskapen som er delt av alle deltakere.

Grounding er viktige for dialogsystemer siden:

1. menneskelige brukere kommer spontant til å produsere slike signaler (spesielt for talebaserte systemer), og det er derfor viktig at deres innhold blir forstått av systemet
2. dialogsystemet trenger også å produsere slike signaler for å formidle til brukeren når systemet har forstått eller ikke forstått hva som blir sagt. Det er kanskje enda viktigere for dialogsystemer siden det kan være vanskelig for brukeren å få oversikt over hva systemet faktisk klare å forstå og hva som ligger utenfor systemets samtalekompetanse.

8.2 Målsetting (4 poeng)

"Velykkede" samtaler må ofte avveie ulike hensyn. For eksempel må oppgaveorienterte dialogsystemer både oppnå et langsiktig mål (f.eks. fullføre en bestilling eller besvare en henvendelse), minimere samtalelengden, og gi en positiv brukeropplevelse ved utgangen av samtalen. Disse målene kan komme i konflikt med hverandre. For eksempel vil et system som fokuserer kun på måloppnåelse føre til lengre samtaler hvor alt må bekreftes flere ganger for å være 100% sikker på at systemet ikke har misforstått noe. Hensynet til effektivisering peker i midlertid i motsatt retning og føre til kortere samtaler, selv om dette kan øke risiko for misforståelser (og dermed redusere sansynnligheten for å oppnå målet).

Det er vanskelig å modellere disse konkurrende hensynene med tradisjonelle, symbolske dialogsystemer. Med reinforcement learning kan vi representere disse hensynene i en reward-funksjon. Funksjonen gir en numerisk "belønning" (som kan være positiv eller negativ) til hver mulig tilstand. Målet for systemet er da å finne adferden som

maksimerer den forventede kumulative belønningen. Funksjonen kan for eksempel gi en liten negativ belønning for hver ny samtaletur (for å utrykke at å forlenge samtalen kan irritere brukeren) og en stor positiv belønning når målet er oppnådd. Reward-funksjonen er en elegant måte å representere og veie disse hensynene opp mot hverandre. I stedet for å måtte programmere akkurat hva systemet skal gjøre i hver mulige tilstand kan systemutvikleren bare spesifisere i reward-funksjonen hva som er ønskede (positiv belønning) og uønskede (negativ belønning) tilstander, og man kan da bruke reinforcement learning til å automatisk lære en adferd som maksimerer den forventede kumulative belønning over hele samtalen.

8.3 Etikk (3 poeng)

Svaret er selvsagt ja. Noen problemstillinger:

1. IR-baserte chatbots bare "plukker opp" eksisterende ytringer fra samtalekorpora, ofte ekstrahert fra ulike nettfora. Disse ytringene kan inneholde bl.a. hatytringer, fordommer, krenkende språkbruk, osv.
2. I tillegg inneholder disse korpora ofte personopplysninger (navn på privatpersoner osv.), uten at disse personene har samtykket for at deres ytringer brukes til å utvikle en chatbot
3. En sist punkt er at IR-basert chatbots ofte kommer med svar som ser "troverdig" ut (de har jo plukket opp fra korpora av menneskelige ytringer), selv om IR-baserte systemet egentlig har en veldig overfladisk forståelse av samtalesituasjonen, og ikke har evnen til å produsere nye ytringer som ikke allerede finnes i det opprinnelige korpuset. Det kan være med til å gi brukere en feil oppfatning at systemet er smartere enn det egentlig er.