# Løsningsforslag til eksamen i IN2110, vår 2022

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

## 1 Vektorrom (38 poeng)

### 1.1 K-means vs Rocchio (8 poeng)

- Mens Rocchio er et eksempel på en algoritme for klassifikasjon basert på veiledet læring fra annoterte data, så er K-means et eksempel på klyngeanalyse og basert på ikke-veiledet læring fra rå/ikke-annoterte data .
- 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.
- Siden metodene deler samme centroide-baserte representasjon av grupper i dataene delere de også de underliggende antakelsene for hva grupper av datapunkter kan være, nemlig linært separerbare sammenhengende sfæriske regioner med ca like stor radius.
- 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 altså si at den gjør Rocchio-klassifikasjon i hver iterasjon.

### 1.2 Distribusjonell semantikk (5 poeng)

- Den distribusjonelle hypotesen antar at ord som har lik betydning også forekommer i like kontekster.
- Det betyr at vi kan måle hvor like to gitte ord er semantisk ved å sammenlikne hvilke kontekster de forekommer i over et stort korpus (tekstsamling).
- Antonymi er det motsatte av synonymi og betegner relasjonen mellom to ord som har motstridende betydning. Eksempler på antonymer kan være f.eks. varm/kald, slem/snill, død/levende, osv.
- Antonymer utforder den distribusjonelle hypotesen ved at de typisk forekommer i like kontekster, men likevel betyr det motsatte av hverandre. Antomymer vil typisk fremstå som svært like distribusjonelt sett, selv om de er semantiske motsatser.

### 1.3 Vektortyper (5 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.
- Mens tradisjonelle vektorer gjerne er basert på opptellinger av manuelt definerte trekk-typer, så er embeddings typisk representasjoner som har blitt lært som del av en annen (typisk nevral) modell, f.eks. en nevral språkmodell for å predikere ord i kontekst.

### 1.4 BoW + ordfrekvens (8 poeng)

- En fordel er rett og slett at representasjonen er enkel — den er konseptuelt enkel å forstå og også enkel å implementere.
- Siden den er basert på diskré trekk, og verdier som enkelt kan inspiseres, er det også enkelt å forstå (tolke) modellen.

- En ulempe knyttet til å ha diskré trekk er at det ikke er noen informasjonsdeling mellom ulike trekk — modellen vet ikke at forekomster av ordet 'pizza' er et trekk som likner på forekomster av ordet 'margherita'.
- Den største ulempen er at modellen ikke tar hensyn til komposisjonalitet, altså at måten ord kombineres på har betydning for semantikken. For eksempel betyr "veldig dårlig, ikke bra" noe ganske annet enn "ikke dårlig, veldig bra", men i en BoW-representasjon ville disse sekvensene få samme representasjon siden hverken rekkefølge eller ordrelasjoner tas høyde for. Hvor stor ulempe dette er kommer an på den spesifikke anvendelsen – dersom man f.eks. skal klassifisere tematikk (f.eks. sport, underholdning, utenriks, osv.) for lange dokumenter (f.eks. nyhets-artikler), så har kanskje ikke komposisjonalitet så mye å si, men dersom vi skal klassifisere polariteten til setninger vil det potensielt være svært viktig.
- Nok en annen ulempe er knyttet til å bruke rå frekvenser. Mange svært vanlige ord bidrar kun i liten grad til meningsinnholdet i en tekst, men på grunn av deres høye frekvens vil de likevel bli tillagt mye vekt i representasjonen. To mulige grep for å endre dette vil være å foreta vekting av trekkverdiene (f.eks. TF-IDF), i tillegg til å bruke en liste av såkalte stopp-ord som vi filtrerer ut.

### 1.5 Pre-prosessering (5 poeng)

- Det første relevante trinnet er tokenisering, altså hvordan sekvensen av løpende tekst deles opp i enkeltord. Typisk skilles da tegnsetting (parenteser, komma, punktum, anførselstegn, osv.) fra resten av ordet.
- Det er videre en mulighet å bruke full-formene av ordene direkte, altså slik de står skrevet i teksten, med eller uten normalisering av store/små bokstaver, osv.
- Men det er også utbredt å benytte andre former for normalisering av ordene, som lemmatisering eller stemming. Lemmatisering prøver å identifisere rotformen av ordet f.eks 'retter' – 'rette' (noe som også implisit involerer å finne riktig ordklasse 'retter' som substantiv skal lemmatiseres til 'rett', i stedet for verb-formen 'rette'). Ved stemming sikter man gjerne på å identifisere felles prefikser av ordene, uten at disse nødvendig er lingvistisk meningsfulle.
- Det er heller ikke uvanlig å filtrere ut såkalte stopp-ord, typisk høyfrekvente funksjonsord fra lukkete ordklasser, som preposisjoner osv.
- I en BoW-modell vil hvert trekk/dimensjon tilsvare en ordtype (altså et unikt ord i vokabulæret). Ved å benytte ulike former for normalisering som stemming, lemmatisering, stoppliste, osv., får vi færre unike ordtyper, og dermed færre trekk i modellen.

### 1.6 P og R (7 poeng)

- Vi antar her at vi jobber med binær klassifikasjon med positive og negative eksempler for klassen vi er interessert i å identifisere.
- $P = \frac{TP}{TP+FP}$
- Precision måler andelen positivt klassifiserte eksempler som faktisk er korrekte (**1**).
- $R = \frac{TP}{TP+FN}$
- Recall måler andelen av faktisk positive eksempler som har blitt korrekt identifisert.
- Det er fullt mulig å oppnå høye (eller lave) verdier for både P og R samtidig. Men det er også mulig å få høy verdi for kun en av dem, og det er i praksis ofte en avveining mellom å optimalisere for det ene eller andre målet. Dersom vi f.eks. klassifiserer alt som medlem av den positive klassen vil vi oppnå 100% recall, men typisk få svært lav precision.

## 2 Logistisk regresjon (8 poeng)

### Spørsmål 1

Modellen inneholder (30 000 vekter og 1 skjæringspunkt) for hver klasse, altså **150 005** parametre totalt.

### Spørsmål 2

Ja, trekk som forekommer svært sjeldent kan føre til **overtrening**, hvor modellen fungerer overraskende bra på treningsett, men generaliserer dårlig til nye datapunkter.

### Spørsmål 3

Ja, regularisering (for eksempel L2-regularisering) kan brukes for å sikre at ingen trekk får for mye "vekt", og dermed unngå å stole for mye på enkelte sjeldne trekk.

### Spørsmål 4

En mulig løsning er å endre **tapsfunksjonen** slik at noen klassifiseringsfeiler (knyttet til klassen vi ønsker å prioritere) får større vekt. Andre løsninger er mulig, for eksempel ved å endre tersklene som skiller klassene.

## 3 HMMs (12 poeng)

### Spørsmål 1

Transisjonsmodell:

| | | Tilstand $t+1$: | | |
|---|---|---|---|---|
| | | Humoristisk (H) | Alvorlig (A) | Snakkesalig (S) |
| Tilstand $t$: | <start> | 0.4 | 0.1 | 0.5 |
| | Humoristisk (H) | 0 | 0.5 | 0.5 |
| | Alvorlig (A) | 0.5 | 0 | 0.5 |
| | Snakkesalig (S) | 0.5 | 0.5 | 0 |

Emisjonsmodell:

| | | Observasjon $t$: | |
|---|---|---|---|
| | | Glad (G) | Bekymret (B) |
| Tilstand $t$: | Humoristisk (H) | 0.9 | 0.1 |
| | Alvorlig (A) | 0.4 | 0.6 |
| | Snakkesalig (S) | 0.7 | 0.3 |

### Spørsmål 2

Ved $t = 0$ er vi per definisjon på tilstand <start>.

Viterbi-verdier for $t = 1$, gitt at $o_{t=1} = G$:

$$\begin{aligned} v_1(H) &= P(s_{t=1} = H \mid s_{t=0} = \langle \text{start} \rangle)\ P(o_{t=1} = G \mid s_{t=1} = H) \\ &= 0.4 * 0.9 = 0.36 \end{aligned}$$

$$v_1(A) = 0.1 * 0.4 = 0.04$$

$$v_1(S) = 0.5 * 0.7 = 0.35$$

Viterbi-verdier for $t = 2$, gitt at $o_{t=2} = B$:

$$\begin{aligned} v_2(H) &= \max_{x \in \{H,A,S\}} \left(P(s_{t=2} = H \mid s_{t=1} = x)\ v_1(x)\right)\ \ P(o_{t=2} = B \mid s_{t=2} = H) \\ &= \max([0, 0.5 * 0.04, 0.5 * 0.35]) * 0.1 = 0.0175 \end{aligned}$$

$$v_2(A) = \max_{x \in \{H,A,S\}} (P(s_{t=2} = A \mid s_{t=1} = x)\ v_1(x))\ \ P(o_{t=2} = B \mid s_{t=2} = A)$$
$$= \max([0.5 * 0.36, 0, 0.5 * 0.35]) * 0.6 = 0.108$$

$$v_2(S) = \max_{x \in \{H,A,S\}} (P(s_{t=2} = S \mid s_{t=1} = x)\ v_1(x))\ \ P(o_{t=2} = B \mid s_{t=2} = S)$$
$$= \max([0.5 * 0.36, 0.5 * 0.04, 0]) * 0.3 = 0.054$$

Vi kan dermed si at den mest sannsynlige tilstanden for $t = 1$ er den humoristiske adferden, og den meste sannsynlige tilstanden for $t = 2$ er den alvorlige adferden.

For å beregne den endelige sannsynligheten bør vi i tillegg normalisere, slik at

$$P(s_{t=2} = A \mid o_{t=1} = G, o_{t=2} = B) = \frac{0.108}{0.0175 + 0.108 + 0.054} = 0.602$$

# 4 Dependenssyntaks og parsing (20 poeng)

## 4.1 Dependenssyntaks (10 poeng)

1. Dependensgrammatikk kjennetegnes ved rettede relasjoner (bileksikale") mellom to ord: et hode og en dependent. Relasjonene er merket med funksjoner, som feks subjekt og adverbial og beskriver derfor syntaktisk funksjon snarere enn form. DG skiller seg som sådann fra frasestrukturgrammatikk i det ikke forekommer fraser i dependensgrafen og heller ingen strukturelle kategorier, feks NP, VP, som beskriver frasene.

2. De to kriteriene som her står i konflikt er semantiske og syntaktiske kriterier, der det første vil kreve at hodet angir det semantiske innholdet til konstruksjonen mens det syntaktiske kriteriet angir de syntaktiske egenkapene til konstruksjonen og at hodet ofte kan opptre alene. Når det gjelder *har pekt* så uttrykker *pekt* det meste av det semantiske innholdet, mens *har* kun opererer som hjelpeverb. Dette vil kunne indikere at *pekt* bør være hode. Samtidig er *har* syntaktisk sett mer sentral da det er det finitte verbet som kan opptre alene i en setning og som bøyes for tid. I analysen her har man lagt vekt på det semantiske kriteriet og valgt *pekt* som hode og *har* som dependent.

## 4.2 Dependensparsing (10 poeng)

| | | |
|---|---|---|
| [ root ] | [ Mange har pekt på EU som løsning på Europas problemer ] | SHIFT |
| [ root Mange ] | [ har pekt på EU som løsning på Europas problemer ] | SHIFT |
| [ root Mange har ] | [ pekt på EU som løsning på Europas problemer ] | LEFT-ARC_aux |
| [ root Mange ] | [ pekt på EU som løsning på Europas problemer ] | LEFT-ARC_nsubj |
| [ root ] | [ pekt på EU som løsning på Europas problemer ] | RIGHT-ARC_root |
| [ root pekt ] | [ på EU som løsning på Europas problemer ] | SHIFT |
| [ root pekt på ] | [ EU som løsning på Europas problemer ] | LEFT-ARC_case |
| [ root pekt ] | [ EU som løsning på Europas problemer ] | RIGHT-ARC_obl |
| [ root pekt EU ] | [ som løsning på Europas problemer ] | SHIFT |
| [ root pekt EU som ] | [ løsning på Europas problemer ] | LEFT-ARC_mark |
| [ root pekt EU ] | [ løsning på Europas problemer ] | REDUCE |
| [ root pekt ] | [ løsning på Europas problemer ] | RIGHT-ARC_xcomp |
| [ root pekt løsning ] | [ på Europas problemer ] | SHIFT |
| [ root pekt løsning på ] | [ Europas problemer ] | SHIFT |
| [ root pekt løsning på Europas ] | [ problemer ] | LEFT-ARC_nmod |
| [ root pekt løsning på ] | [ problemer ] | LEFT-ARC_case |
| [ root pekt løsning ] | [ problemer ] | RIGHT-ARC_nmod |
| [ root pekt løsning problemer ] | [ ] | REDUCE |
| [ root pekt løsning ] | [ ] | REDUCE |
| [ root pekt ] | [ ] | REDUCE |
| [ root ] | [ ] | REDUCE |

# 5 Maskinoversettelse (12 poeng)

## Spørsmål 1

Et pro-drop eller nullanaforaspråk er et språk som har noen pronomer som ikke trenges å uttrykke eksplitt i setningen. Slike språk kan skape utfordringer i maskinoversettelse, når man oversetter fra et slikt språk til et språk hvor pronomen er obligatorisk, da modellen vil da måtte finne ut hva slags pronomen (f.eks. kvinnelig eller mannlig) bør brukes avhengig av konteksten.

## Spørsmål 2

Vi starter med å beregne presisjonene for $n = 1, 2, 3$ eller $4$:

For unigrams har vi 5 ord som forekommer både i systemoutput og i fasit for den første setningen (ut fra 7 ord som ble produsert, inkludert tegnsetting), og 4 ord for den andre setningen (ut fra 6 ord som ble produsert). Dermed:

$$p_{n=1} = \frac{5+4}{7+6} = 0.692$$

Vi gjentar for bigrams, trigrams, og quadrigrams:

$$p_{n=2} = \frac{3+3}{6+5} = 0.545$$

$$p_{n=3} = \frac{2+2}{5+4} = 0.444$$

$$p_{n=4} = \frac{1+1}{4+3} = 0.286$$

Vi må også beregne brevity penalty: $min([1, \frac{7+6}{8+6}]) = 0.929$

BLEU blir dermed: $0.929 * (0.692 * 0.545 * 0.444 * 0.286)^{1/4} = 0.437$.

# 6 Dialogsystemer (10 poeng)

## Spørsmål 1

Her er det rom for ulike forklaringer, men her er et forslag:

- Regelbaserte systemer krever i utgangspunktet ingen innsamlede dialogdata, selv om et korpus av eksisterende samtaler kan være nyttig for å skrive gode regler som er tilpasset domenen.
- IR-baserte systemer krever et korpus av samtaler, slik at systemet kan finne "lignende" kontekster/forespørsler, og plukke opp hensiksmessige responser blant ytringene i korpuset.
- Sequence-to-sequence systemer krever også et korpus av eksisterende samtaler for å trene den nevrale modellen som skal enkodere input og dekodere svaret
- NLU-baserte modeller (med intentgjenkjenning) krevet et treningskorpus av inputs som er annotert med kategorier (intents), og kanskje også noen slot-verdier.

## Spørsmål 2

- Regelbaserte systemer kan utvikles uten å ha tilgang til treningssdata, noe som kan være en stor fordel i noen anvendelser. I tillegg er de forholdsvis enkle å styre, da man kan lese reglene og endre disse ved behov. Men det er også en god del utfordringer, blant annet at de er vanskelig å utvikle, vanskelige å skalere til mer kompliserte samtaler, og kan virke ganske rigide / mekaniske.

- IR-baserte systemer er forholdsvis lett å trene, og kan produsere overraskende gode svar i mange tilfeller. Men de er begrenset til å velge ytringer som allerede finnes i korpuset, og kan dermed ikke produsere “nye” svar. En god modell krever også et fordholdsvis stort korpus (av god kvalitet).
- Sequence-to-sequence systemer kan generere helt nye setninger som aldri har blitt observert i treningskorpuset, og er en elegant og fleksibel tilnærming. Men å trene slike modeller kan være vanskelig, spesielt når man ønsker å styre"modellen i enn viss retning, for eksempel slik at modellen foretrekker visse formuleringer eller svar.
- NLU-baserte modeller fungerer bra for anvendelser hvor man vet på forhånd hva systemet skal snakke om, og trenger å få kontroll på hva systemet faktisk sier til brukere. Men slike modeller krever annoterte eksempler (hvor brukerytringer er klassifiserte i ulike intents, og muligens også slotverdier"). Også vanskelig å bruke for mer uformelle samtaler.