Lec 12 (Reinforcement) Learning w Human Feedback

Getting human feedback (for generation task)

So far: maximum likelihood training

Problems:
1. Some mistakes are worse than others
   a ↔ the — maybe less bad
   Tokyo ↔ Pittsburgh — bad
   ] but to likelihood, these are equal
2. Gold standard can be bad
   e.g. Google translate posted online
3. Overexposure bias
   previously appeared word more likely to appear again
   model repeats itself
   model not seen bad text

Measure output quality

- Objective assessment
  - annotated data
- Subjective human eval
  - generate hypotheses, human rate them
  - trait-based assessments
    - → fluency
    - → adequacy (is it semantically correct)
    - → factuality (in general or w.r.t. input)
    - → coherence
  - preference rating — which is better?
    - good for relative cmp but not globally how good
  - hybrid: preference rating + score rating
  - use ELO ranking
    - pairwise battel to rank many models
  - from MT: multi-dim quality metric
    - annotate spans in seq, categorise them, and rate their severity (minor / major)
    - \+ consistent, fine-grained
    - − time consuming
- Machine predict human preference "reward model", "automatic eval"
  - reference human feedback for some outputs
  - → Embedding based - (BERTScore)
    unsupervised, calc embedding similarity btwn output & ref
    embed each tkn, pairwise cos sim btwn two seqs, mk sim matrix, take max along a dim, ...
  - → Regression-based (COMET)
    supervised training of emb-based regressor
  - → QA-based
    ask LM how good it is wrt reference (GEMBA)
    - e.g. GPT

    ask LM what the mistakes are (AutoMQM)
    + more consistent results
- Downstream task
  - Intrinsic: quality of downstream output itself
  - Extrinsic: by utility

Human feedback postproc

→ Normalise per annotator
→ Average
→ Drop examples human disagree on

Meta evaluation

- Are metrics associated?
- .. human / auto eval correlated?

→ Can use large shared dsets to do this

Error and Risk

Generate output $\hat{Y}$
Calculate the badness $error(Y, \hat{Y}) = 1 - eval(Y, \hat{Y})$
Try to minimise error, usually can't do gradient-based opt

$$risk(X, Y, \theta) = \sum_{\tilde{Y}} P(\tilde{Y} \mid X; \theta)\, error(Y, \tilde{Y})$$

← but too many things in possible outputs. intractable

Try minimise risk instead

Sampling (5-50 sentences) (to make tractable)

$$risk(X, Y, \theta) = \sum_{\tilde{Y} \in S} \frac{P(\tilde{Y} \mid X; \theta)}{Z}\, error(Y, \tilde{Y})$$

→ Sampling, n-best search, etc.

Reinforcement Learning, Policy Grad

Given: env X
action A
delayed reward R

Generation: X — conversation history
A — next tkn
R — human pressing buttons, call center outcome

Coode: X — coode, env
A — tkn
R — unit tests, speed, readability, etc.

Why reinforcement in NLP
- When having delayed reward info
- Have a latent variable
- Have sequence-level evaluation

Supervised MLE loss for imitation learning $\quad l_{super}(X, Y) = -\log P(Y \mid X)$

Self training $\quad l_{self}(X) = -\log P(\hat{Y} \mid X)$
↳ optimises toward both good and bad output but actually works in some situation

→ Co-training
→ ...

▷ Policy Grad / REINFORCE

scale loss by reward $\quad l_R(X, \hat{Y}) = -\underbrace{R(Y, \hat{Y})}_{\text{when reward is 1}} \log(Y \mid X)$

Problem: how do we know which action led to reward?
- best: immediate reward
- worse: eventually reward

→ Just not deal with it ← usually ppl just do this
→ Decay reward for future event (viz. scale by time)

Reinforcement Learning Stabalisation

Unstable reasons
- Sample individual output
- Very large output space

▷ Pre-train with MLE, then switch to RL
just always do this

▷ Regularise to existing LM
- not let the model diverge too far away

→ KL regularisation
$l_{regularisation}$ = reward stuff − β KL with other model

→ Proximal policy opt
use clipping to not reward large jumps
push model back if model goes too bad

▷ Adding a baseline
account for hard vs easy examples by having expectation on reward

opt with reward − baseline

baseline can be arbitrary
1. use model to predict reward
   - sentence level
   - decoder state level
2. calc mean reward of batch

▷ Contrasting Pairwise Examples
- learn directly from pairwise human judgement

→ DPO — upweight better output, downweight the other

▷ Increase batch size
- RL samples high variance so higher batch size helps