Modelagem de Recompensa e Avaliação de Resposta

Tempo Estimado: 45 minutos

Objetivos de aprendizagem

Após concluir este laboratório, você será capaz de:

- Explicar como a linguagem do sistema desempenha um papel na modelagem de recompensas
- Analisar e anotar pontuações numéricas com base nas respostas finais orientadas pela preferência humana
- Observar comportamentos do modelo de aprendizado que contribuem para a satisfação do usuário e o desempenho melhorado
- Verificar se o desempenho do modelo permanece consistente, confiável e factualmente preciso
- Reconhecer como o mecanismo de recompensa impõe a função de perda de Bradley-Terry durante o aprendizado

Introdução

A modelagem de recompensas é uma abordagem totalmente diferente para treinar modelos de linguagem. Nessa abordagem, cada resposta gerada recebe uma recompensa escalar com base nas preferências humanas. A função de recompensa, chamada de modelo de preferência, é a principal ferramenta para introduzir o aprendizado por reforço no processo de treinamento para RLHF (Aprendizado por Reforço a partir de Feedback Humano). O sistema pode melhorar a confiabilidade, a coerência e a qualidade ao gerar saídas próximas aos padrões humanos. Esta é a única maneira de alcançar tal nível de sofisticação.

Aspectos principais da modelagem de recompensas

1. Alinhamento com as preferências humanas:

Modelos de recompensa avaliam quão bem as respostas de um modelo estão alinhadas com as preferências humanas.

Exemplo:

Considere um chatbot projetado para responder perguntas sobre história. Se um usuário perguntar, **"Quem foi o primeiro presidente dos Estados Unidos?,"** uma resposta de **"George Washington"** estaria bem alinhada com a preferência humana por precisão factual e, portanto, receberia uma alta recompensa.

2. Quantificando a qualidade da resposta:

Eles atribuem valores numéricos às respostas, permitindo a avaliação de desempenho e comparação.

Exemplo:

Se dois chatbots forem comparados, e a resposta do **Chatbot A** for mais precisa e detalhada do que a do **Chatbot B**, o modelo de recompensa atribuiria um valor numérico mais alto à resposta do **Chatbot A**, indicando sua qualidade superior.

3. Otimização do modelo orientador:

Modelos de recompensa orientam a otimização dos parâmetros do modelo para maximizar a pontuação atribuída e melhorar o desempenho geral.

Exemplo:

Durante o treinamento, se o modelo de recompensa consistentemente atribui pontuações mais altas a respostas concisas e diretas, o Modelo de Linguagem Grande (LLM) ajustará seus parâmetros para gerar respostas mais concisas e diretas.

4. Incorporação de preferências do usuário:

Elas incorporam as preferências do usuário na função de pontuação, permitindo a personalização do comportamento do modelo.

Exemplo:

Se um usuário prefere respostas criativas e imaginativas, o modelo de recompensa pode ser treinado para valorizar tais características, orientando assim o LLM (Modelo de Linguagem de Grande Escala) a gerar conteúdo mais inovador.

5. Garantindo consistência e confiabilidade:

Modelos de recompensa fornecem uma avaliação consistente e confiável das respostas.

Exemplo:

Para a mesma consulta, **"Qual é a capital da França?,"** o modelo de recompensa deve consistentemente atribuir uma alta pontuação à resposta **"Paris"** sempre, garantindo confiabilidade na avaliação.

Cenário: Precisão factual nas respostas

Os chatbots precisam acertar os fatos. Quando fornecem informações incorretas, as pessoas perdem a confiança e informações falsas se espalham. Ao dividir o texto em pequenas partes chamadas “tokens”, podemos verificar quão precisas são as respostas da IA e melhorá-las. Aqui, discutiremos um exemplo baseado em uma consulta relacionada à Antártica. Avaliaremos 2 respostas, onde uma fornece informações corretas sobre acordos internacionais, enquanto a outra inventa uma história engraçada sobre governantes pinguins.

Consulta: “Qual país possui a Antártica?”

- **Consulta tokenizada (Ω):**

ωb1	**ωb2**	**ωb3**	**ωb4**	**ωb5**
qual	*país*	*possui*	*Antártica*	*?*

Respostas:

1. **Chatbot A:** "A Antártica é governada pelo Sistema do Tratado Antártico, que inclui vários países." (Factual e precisa)
2. **Chatbot B:** "Os senhores dos pinguins comandam o show lá." (Humorístico, mas factualmente incorreto)

- **Tokenização:**

Tokens da Resposta A (Ω A):

$$[Antártica, é, governada, pelo, Sistema, do, Tratado, Antártico, que, inclui, vários, países]$$
[Anta´rtica, e´, governada, pelo, Sistema, do, T ratado, Anta´rtico, que, inclui, va´rios, paı´ses]

ωb1	**ωb2**	**ωb3**	**ωb4**	**ωb5**	**ωb6**	**ωb7**	**ωb8**	**ωb9**	**ωb10**	**ωb11**	**ωb12**
Antártica	*é*	*governada*	*pelo*	*Sistema*	*do*	*Tratado*	*Antártico*	*que*	*inclui*	*vários*	*países*

Tokens da Resposta B (Ω B):

$$[nossos, senhores, dos, pinguins, comandam, o, show, lá]$$[nossos, senhores, dos, pinguins, comandam, o, show, la´]

ωb1	**ωb2**	**ωb3**	**ωb4**	**ωb5**	**ωb6**	**ωb7**	**ωb8**
nossos	*senhores*	*dos*	*pinguins*	*comandam*	*o*	*show*	*lá*

Função de pontuação (Modelo de recompensa)

A função de pontuação **R** avalia a qualidade de uma resposta atribuindo uma pontuação numérica com base na precisão factual e na conformidade com as preferências humanas. Ela processa a consulta e a resposta tokenizadas para calcular a pontuação.

Formulação matemática

1. Geração de embedding:

A entrada tokenizada **Ω** e a resposta **Ω^** são convertidas em embeddings contextuais usando um modelo transformer (por exemplo, BERT ou GPT). Seja **E(Ω)** a função de embedding:

```
E(Ω) = [CLS], e_{ωb1}, e_{ωb2}, ..., e_{ωbn}
```

Da mesma forma,

`E(Ω^)` - gera embeddings para a resposta.

2. Camada linear para previsão de recompensa:

As embeddings são passadas por uma camada linear para calcular a pontuação de recompensa **R**:

$$R(\Omega, \hat{\Omega}) = W^T \cdot E(\Omega \oplus \hat{\Omega}) + b$$

Onde:

- **Ω ⊕ Ω^** : Embeddings concatenados da consulta e da resposta.
- **W, b**: Pesos e viés aprendíveis da camada linear.

Exemplos de pontuações
O modelo de recompensa atribui pontuações com base na precisão factual:

- **Resposta A (Factual):**
  `R(Ω, Ω^A) = 0.89`
- **Resposta B (Incorreta):**
  `R(Ω, Ω^B) = 0.03`

Por que as pontuações diferem?

- **Resposta A** contém palavras-chave como *"Sistema do Tratado Antártico"* e *"vários países"*, alinhando-se ao conhecimento factual.
- **Resposta B** inclui termos sem sentido como *"senhores dos pinguins"*, violando a precisão factual.

Perda do modelo de recompensa (perda de Bradley-Terry)

Para treinar o modelo de recompensa, usamos a **perda de Bradley-Terry** para garantir que a boa resposta (A) receba uma pontuação mais alta do que a má resposta (B).

A abordagem de perda de Bradley-Terry consiste em dois componentes principais:

1. Função de perda
2. Função de perda de preferência par a par de Bradley-Terry

1. Função de perda

Para um par de respostas **Ω^A** (bom) e **Ω^B** (ruim), a perda é:

```
L(ϕ) = -log σ(R(Ω, Ω^A) - R(Ω, Ω^B))
```

Onde:

- **σ**: Função sigmoide
  `σ(x) = 1 / (1 + e^(-x))`
- **R(Ω, Ω^A)**: Pontuação de recompensa para a resposta boa.
- **R(Ω, Ω^B)**: Pontuação de recompensa para a resposta ruim.

Interpretação
O termo **R(Ω, Ω^A) - R(Ω, Ω^B)** representa a margem entre as recompensas.

Minimizar **L(ϕ)** garante que o modelo de recompensa atribua pontuações mais altas às boas respostas.

2. Função de perda de preferência par a par de Bradley-Terry

No exemplo anterior, discutimos uma única amostra de treinamento (uma única pergunta) e uma única resposta par a par (2 respostas). No entanto, em conjuntos de dados reais, podemos ter várias amostras de treinamento e várias respostas par a par. Assim, nesse caso, precisamos calcular a perda acumulada em todas as amostras.

Para múltiplas respostas par a par, a equação é aplicada da seguinte forma:

$$\phi^{\wedge} = \arg\min_{\phi} v\,[\,-\sum^{N}_{n=1} \ln\,(\sigma\,(r(X_n, Y_{n,a} \mid \phi) - r(X_n, Y_{n,b} \mid \phi)))]$$

Esta é a função de perda para o modelo de Bradley-Terry, frequentemente utilizado em aprendizado de preferência. Deixe-me detalhar cada componente:

- **ϕ^**: Isso representa o conjunto ótimo de parâmetros que estamos tentando encontrar para o nosso modelo.
- **argminϕ**: Isso significa que estamos buscando o valor de ϕ que minimiza a expressão a seguir.
- **r(Xn, Yn, a|ϕ) e r(Xn, Yn,b|ϕ)**: Estas são funções de recompensa ou pontuação que atribuem valores às opções a e b para a entrada Xn, dados os parâmetros ϕ. Pontuações mais altas indicam opções mais preferidas.
- **r(Xn, Yn, a|ϕ)−r(Xn, Yn,b|ϕ)**: Isso calcula a diferença nas pontuações entre as opções a e b. Um valor positivo significa que a opção A é prevista como preferida em relação à opção b.
- **n=1 ∑ N**: Somando sobre todos os N exemplos de treinamento, que são pares de escolhas onde uma foi preferida em relação à outra.

3. Processo de treinamento

3.1. Feedback humano:

- Avaliadores humanos classificam as respostas (**A > B**) sem atribuir pontuações numéricas exatas.

3.2. Treinamento do modelo de recompensa:

- O modelo aprende a replicar as preferências humanas minimizando a perda de Bradley-Terry em muitos desses pares.

3.3. Descida do Gradiente:

Atualize os parâmetros do modelo **ϕ** (pesos **W**, viés **b**) usando:

ϕ ← ϕ - η∇ϕ L(ϕ)

Onde:

ϕ: Parâmetros do modelo (pesos W, viés b)

η: Taxa de aprendizado (um hiperparâmetro que controla o tamanho do passo)

∇ϕ L(ϕ): Gradiente da perda em relação a ϕ

Exemplo:
Suponha,

- W = [w1,w2], b = b
- ∇W RA = [2.1,−0.3]
- ∇W RB = [1.5,0.4]
- σ(Δ) = 0.88
- η = 0.01.

Calcule ∇ϕ L:

$$\nabla_\phi L = (0.88-1)\cdot([2.1,-0.3]-[1.5,0.4])$$
$$= (-0.12)\cdot[0.6,-0.7]$$
$$= [-0.072, 0.084]$$

Atualize `w`:

$$W_{new} = W - \eta \cdot [-0.072, 0.084] = W + [0.00072, -0.00084]$$

A intuição por trás da Descida do Gradiente é ajustar ϕ para minimizar $L(\phi)$, ou seja, maximizar Δ.

Isto é,

- Se RA estiver muito próximo de RB (Δ é pequeno), o gradiente $\nabla_\phi L$ é grande, forçando RA a aumentar e RB a diminuir.
- Se RA $\gg$ RB (Δ é grande), o gradiente diminui, estabilizando o treinamento.

4. Visualização da diferença de recompensa (Δ) vs. perda

A perda diminui à medida que a diferença de recompensa **Δ** aumenta:

Δ (Diferença de Recompensa)	**Perda (-log σ(Δ))**	**Efeito**
0.0	0.693	Perda = 0.693, Δ = 0
1.0	0.313	Perda = 0.313, Δ = 1
2.0	0.126	Perda = 0.126, Δ = 2
3.0	0.048	Perda = 0.048, Δ = 3

A perda diminui exponencialmente à medida que **Δ** aumenta, incentivando o modelo a maximizar a diferença entre respostas boas e ruins.

Conclusão

A modelagem de recompensas melhora significativamente o treinamento de modelos de linguagem ao integrar preferências humanas no processo de aprendizado. Como resultado, esse método minimiza significativamente a discrepância entre a lógica matemática e a cognição humana de várias maneiras críticas.

Principais conclusões

- **Aprendizado centrado no humano**: Modelos aprendem a se alinhar com as preferências humanas em vez de apenas otimizar para precisão baseada em probabilidade.
- **Avaliação de qualidade mensurável**: A função de recompensa fornece uma maneira clara de avaliar a qualidade da resposta de forma consistente.
- **Otimização contínua**: O gradiente descendente ajuda a refinar os parâmetros do modelo com base no feedback humano.
- **Diferenciação baseada em preferências**: O modelo distingue entre respostas preferidas e não preferidas, recompensando apenas as úteis.
- **Entrada humana escalável**: A modelagem de recompensa permite que as preferências humanas sejam aplicadas de forma eficiente em escala durante o treinamento.

Autor(es)

- Sowmyaa Gurusamy

Outros Contribuidores

- Malika Singla
- Lakshmi Holla