Classificação com Modelos de Linguagem

Escola de Verão 2025

Alexandre Teles <alexandre.teles@inctdd.org>

Índice

Por que treinar LLMs?

- Customização para domínios específicos
  - Modelos podem ser adaptados para entender jargões e contextos específicos de uma área
  - Exemplo: um modelo especializado em textos médicos terá melhor desempenho em diagnósticos
- Melhoria de desempenho em tarefas especializadas
  - Fine-tuning permite otimizar o modelo para tarefas específicas
  - Exemplo: um modelo pode ser ajustado para ser especialmente bom em análise de sentimentos
- Controle sobre o comportamento do modelo
  - Permite definir o "tom de voz" e estilo de resposta do modelo
  - Importante para alinhar o modelo com políticas corporativas e requisitos éticos
- Redução de custos operacionais
  - Modelos menores e especializados podem ser mais eficientes que modelos grandes genéricos
  - Menor consumo de recursos computacionais no deploy

Desafios e Oportunidades

- Requisitos computacionais significativos
  - Necessidade de GPUs potentes para treinamento
  - Técnicas de otimização podem reduzir estes requisitos
- Necessidade de dados de qualidade
  - Dados ruins podem resultar em modelos com vieses ou comportamentos indesejados
  - A qualidade dos dados impacta diretamente a qualidade do modelo final
- Importância do formato correto dos dados
  - Dados mal formatados podem prejudicar o treinamento
  - Consistência no formato é crucial para bons resultados
- Novas técnicas tornando o processo mais acessível
  - Métodos como LoRA reduzem necessidade de recursos
  - Ferramentas como Unsloth democratizam o processo

Modelos Base vs Instruction Fine-tunes

Modelos Base

- Treinados em grandes corpus de texto
  - Aprendem padrões gerais da linguagem através de grandes volumes de texto
  - Base de conhecimento ampla, mas sem foco específico
- Capacidade de geração de texto mais genérica
  - Bons em tarefas criativas e geração livre
  - Podem produzir conteúdo inesperado ou fora do contexto
- Menos controle sobre o comportamento
  - Podem não seguir instruções de forma consistente
  - Respostas podem variar muito em estilo e formato
- Exemplos: LLaMA 2, Mistral, MPT
  - Modelos que servem como fundação para fine-tuning
  - Geralmente disponíveis em diferentes tamanhos

Modelos Instruction-tuned

- Adaptados para seguir instruções
  - Treinados especificamente para entender e executar comandos
  - Melhor compreensão do que o usuário deseja
- Comportamento mais controlado e previsível
  - Respostas mais consistentes e focadas
  - Menor probabilidade de gerar conteúdo inadequado
- Melhor alinhamento com intenções humanas
  - Capacidade de entender nuances em instruções
  - Melhor interpretação do contexto conversacional
- Exemplos: Alpaca, Vicuna, OpenOrca
  - Modelos que já passaram por processo de fine-tuning
  - Prontos para uso em aplicações práticas

Supervised Fine-tuning

Fundamentos

- Treinamento com pares de pergunta-resposta
  - Cada exemplo contém uma entrada e a resposta desejada
  - O modelo aprende a mapear entradas para saídas específicas
- Otimização direta da likelihood
  - O modelo é treinado para maximizar a probabilidade das respostas corretas
  - Processo similar ao treinamento original, mas com dados específicos
- Necessidade de dados de alta qualidade
  - Dados devem representar bem o comportamento desejado
  - Qualidade dos exemplos impacta diretamente o resultado
- Processo relativamente direto
  - Metodologia bem estabelecida e compreendida
  - Frameworks maduros disponíveis

Parameter-Efficient Fine-Tuning (PEFT)

O que é PEFT?

- Técnicas para reduzir custos de fine-tuning
  - Treina apenas uma pequena fração dos parâmetros
  - Mantém qualidade comparável ao full fine-tuning
- Benefícios principais
  - Redução dramática de memória necessária
  - Treinamento mais rápido e eficiente
  - Facilita experimentação e iteração
- Aplicações práticas
  - Permite fine-tuning em hardware mais modesto
  - Viabiliza adaptação de modelos muito grandes
  - Reduz custos operacionais significativamente

LoRA (Low-Rank Adaptation)

Fundamentos do LoRA

- Decomposição de matriz de baixo posto
  - Substitui matrizes densas por produtos de matrizes menores
  - Reduz significativamente número de parâmetros treináveis
- Funcionamento básico
  - Congela pesos originais do modelo
  - Adiciona matrizes de baixo posto paralelas
  - Treina apenas as matrizes adicionadas

Vantagens e Características

- Eficiência computacional
  - Redução de 10000x+ em parâmetros treináveis
  - Mantém qualidade próxima ao full fine-tuning
- Flexibilidade
  - Rank configurável (trade-off qualidade vs eficiência)
  - Múltiplos adaptadores podem ser combinados
  - Fácil reversão para modelo original

QLoRA (Quantized LoRA)

Características Principais

- Quantização do modelo base
  - Reduz precisão dos pesos para 4 ou 8 bits
  - Mantém gradientes em precisão completa
- Inovações técnicas
  - Quantização de paginação
  - Dequantização durante forward pass
  - Otimização de memória

Benefícios

- Uso extremamente eficiente de memória
  - Permite fine-tuning em GPUs com menos VRAM
  - Suporte a modelos maiores em hardware modesto
- Preservação de qualidade
  - Resultados próximos ao LoRA tradicional
  - Mínima perda de performance

RsLoRA (Rank-Stabilized LoRA)

Conceito e Inovações

- Estabilização de ranking
  - Melhora estabilidade durante treinamento
  - Reduz problemas de convergência
- Modificações técnicas
  - Normalização adaptativa
  - Controle dinâmico de gradientes
  - Melhor inicialização de pesos

Vantagens Específicas

- Maior estabilidade
  - Treinamento mais robusto
  - Menos sensível a hiperparâmetros
- Melhor convergência
  - Resultados mais consistentes
  - Menor necessidade de múltiplas tentativas

DoRA (Weight-Decomposed Low-Rank Adaptation)

Funcionamento

- Decomposição de pesos
  - Separa magnitude e direção dos pesos
  - Aplica LoRA para adaptação direcional
- Processo de adaptação
  - Fine-tuna componentes de magnitude e direção
  - Usa LoRA para atualizações direcionais eficientes
  - Permite merge com pesos originais após treinamento

Benefícios e Características

- Maior capacidade de aprendizado
  - Padrões de aprendizado mais próximos do full fine-tuning
  - Ajustes direcionais substanciais com mudanças mínimas de magnitude
- Vantagens práticas
  - Compatible com LoRA e variantes
  - Sem overhead adicional após merge
  - Performance superior ao LoRA em várias tarefas

Comparação de Métodos PEFT

Características Principais

- LoRA
  - Base para outros métodos
  - Equilíbrio entre simplicidade e eficácia
  - Amplamente testado e suportado
- QLoRA
  - Máxima eficiência de memória
  - Ideal para hardware limitado
  - Pequeno trade-off de performance
- RsLoRA
  - Foco em estabilidade
  - Menor sensibilidade a configurações
  - Bom para produção
- DoRA
  - Decomposição sofisticada
  - Melhor qualidade de adaptação
  - Sem custos adicionais de inferência

Escolhendo o Método

- Considerar recursos disponíveis
  - Hardware e memória
  - Tempo de treinamento
  - Necessidades de qualidade
- Avaliar complexidade de implementação
  - Suporte de frameworks
  - Necessidade de expertise técnica
  - Maturidade da solução

Considerações Práticas

- Seleção cuidadosa de learning rate
  - Taxa muito alta pode desestabilizar o conhecimento existente
  - Taxa muito baixa pode tornar o treinamento ineficiente
- Importância do tamanho do batch
  - Batches maiores proporcionam treinamento mais estável
  - Limitado pela memória disponível na GPU
- Estratégias de regularização
  - Previne overfitting aos dados de fine-tuning
  - Mantém conhecimentos úteis do pré-treinamento
- Monitoramento de métricas chave
  - Acompanhamento de loss, perplexity e métricas específicas
  - Importante para detectar problemas no treinamento

Métodos Modernos

- Direct Preference Optimization (DPO)
  - Otimiza diretamente a partir de comparações de preferência
  - Elimina necessidade de reward modeling explícito
- Simplified Policy Optimization (SIMPO)
  - Versão mais eficiente do processo DPO
  - Reduz complexidade computacional mantendo eficácia
- Vantagens sobre RLHF tradicional
  - Processo mais simples e direto
  - Menor necessidade de recursos computacionais
- Menor complexidade computacional
  - Treinamento mais rápido e eficiente
  - Mais acessível para equipes com recursos limitados

Direct Preference Optimization (DPO)

Fundamentos do DPO

- Alternativa mais simples ao RLHF tradicional
  - Elimina necessidade de reward modeling
  - Transforma problema de RL em classificação
  - Mais estável e fácil de implementar
- Motivação
  - RLHF é complexo e instável
  - Necessidade de método mais direto
  - Manter qualidade com menos complexidade

Vantagens Principais

- Simplicidade computacional
  - Não requer sampling durante fine-tuning
  - Menor necessidade de ajuste de hiperparâmetros
  - Processo de treinamento mais rápido
- Resultados superiores
  - Melhor controle sobre comportamento do modelo
  - Qualidade comparable ou superior ao RLHF
  - Maior estabilidade no treinamento

Como DPO Funciona

Teoria Básica

- Reformulação matemática
  - Transforma preferências em problema de classificação
  - Deriva política ótima em forma fechada
  - Simplifica processo de otimização
- Processo de treinamento
  - Usa pares de respostas (preferida vs não preferida)
  - Otimiza diretamente diferença de log-probabilidades
  - Mantém proximidade com modelo original

Implementação

- Componentes principais
  - Modelo de linguagem base
  - Dataset de preferências
  - Função de loss simplificada
- Fluxo de treinamento
  - Forward pass com ambas respostas
  - Cálculo de probabilidades relativas
  - Otimização direta da preferência

Preparação de Dados para DPO

Estrutura do Dataset

- Formato básico:

```
{
    "prompt": "Explique o que é gravidade",
    "chosen": "A gravidade é uma força fundamental que atrai todos os objetos com massa...",
    "rejected": "A gravidade é tipo quando as coisas caem no chão..."
}
```

- Componentes necessários
  - Prompt: contexto ou pergunta inicial
  - Chosen: resposta preferida pelos avaliadores
  - Rejected: resposta menos preferida

Boas Práticas

- Qualidade das preferências
  - Avaliações consistentes e bem definidas
  - Critérios claros de preferência
  - Diversidade de exemplos
- Considerações importantes
  - Balanceamento de tópicos
  - Consistência nas avaliações
  - Documentação de critérios

Kahneman-Tversky Optimization (KTO)

Fundamentos do KTO

- Baseado em Prospect Theory
  - Teoria sobre tomada de decisão humana
  - Considera vieses cognitivos naturais
  - Incorpora aversão a perdas
- Diferencial principal
  - Não requer dados de preferência
  - Usa apenas sinais binários (bom/ruim)
  - Mais próximo do comportamento humano real

Vantagens sobre outros métodos

- Coleta de dados simplificada
  - Dados mais fáceis de obter
  - Menor custo de anotação
  - Processo de avaliação mais rápido
- Eficiência e escalabilidade
  - Performance comparable ao DPO
  - Funciona bem de 1B a 30B parâmetros
  - Menor complexidade de implementação

Prospect Theory na Prática

- Conceitos fundamentais
  - Pessoas avaliam ganhos e perdas relativamente
  - Aversão a perdas é mais forte que atração por ganhos
  - Decisões são baseadas em utilidade percebida
- Aplicação em LLMs
  - Modelagem explícita de utilidade humana
  - Incorporação de vieses cognitivos
  - Otimização direta de utilidade percebida

Função de Utilidade

- Características principais
  - Assimetria entre ganhos e perdas
  - Não-linearidade na percepção de valor
  - Ponderação baseada em referências
- Implementação
  - Transformação de probabilidades
  - Incorporação de aversão a perdas
  - Calibração com comportamento humano real

Preparação de Dados para KTO

Estrutura do Dataset

- Formato básico:

```
{
    "prompt": "Explique o conceito de inflação",
    "completion": "Inflação é o aumento generalizado dos preços...",
    "label": true  # ou false para respostas inadequadas
}
```

- Simplicidade dos dados
  - Apenas três campos necessários
  - Label binário (true/false)
  - Sem necessidade de comparações

Exemplos Práticos

```
# Exemplo 1: Resposta Adequada
{
    "prompt": "Como economizar energia?",
    "completion": "Para economizar energia, você pode: 1) Desligar aparelhos não utilizados, 2) Usar lâmpadas LED, 3) Aproveitar a luz natural, 4) Manter aparelhos em bom estado.",
    "label": true
}

# Exemplo 2: Resposta Inadequada
{
    "prompt": "Como economizar energia?",
    "completion": "Sei lá, só deixa tudo ligado que não faz diferença.",
    "label": false
}
```

Simple Preference Optimization (SimPO)

Visão Geral

- Simplificação do DPO
  - Remove necessidade de modelo de referência
  - Processo mais direto e eficiente
  - Mantém ou supera performance do DPO
- Inovações principais
  - Log likelihood normalizado por comprimento
  - Margem de recompensa no objetivo de ranking
  - Função objetivo mais simples e estável

Vantagens Principais

- Eficiência computacional
  - Menor overhead computacional
  - Menos parâmetros para ajustar
  - Treinamento mais rápido
- Simplicidade de implementação
  - Código mais limpo e direto
  - Menos componentes móveis
  - Menor complexidade de debug

Aspectos Técnicos do SimPO

Componentes Chave

- Normalização por comprimento
  - Ajusta likelihood baseado no tamanho da resposta
  - Reduz viés para respostas mais curtas/longas
  - Melhor comparação entre respostas diferentes
- Margem de recompensa
  - Define diferença mínima desejada entre chosen/rejected
  - Melhora estabilidade do treinamento
  - Promove separação mais clara entre respostas

SimPO vs DPO

Diferenças Técnicas

- Modelo de referência
  - SimPO: Não necessita
  - DPO: Usa modelo de referência
- Normalização
  - SimPO: Normaliza por comprimento
  - DPO: Usa log probs brutos
- Margem de recompensa
  - SimPO: Incorpora margem explícita
  - DPO: Não usa margem

Vantagens Comparativas

- SimPO
  - Implementação mais simples
  - Menor overhead computacional
  - Menos hiperparâmetros
- DPO
  - Mais estabelecido
  - Maior base de pesquisa
  - Mais ferramentas disponíveis

Preparação de Dados

Formato dos Dados

- Estrutura de turnos (conversacional)
  - Organização clara de quem fala o quê
  - Facilita o treinamento para diálogos naturais
- Formato ShareGPT
  - Padrão estabelecido pela comunidade
  - Facilita compartilhamento e reuso de dados
- Consistência na formatação
  - Todos os exemplos seguem o mesmo padrão
  - Reduz erros durante o treinamento
- Validação de qualidade
  - Verificação de formato e conteúdo
  - Garante dados adequados para treinamento

Boas Práticas

- Limpeza e normalização
  - Remoção de ruídos e inconsistências
  - Padronização de formato e estilo
- Verificação de duplicatas
  - Evita viés por repetição excessiva
  - Otimiza uso de recursos de treinamento
- Balanceamento de tópicos
  - Evita viés para certos tipos de conteúdo
  - Garante cobertura adequada de diferentes áreas
- Validação de qualidade dos diálogos
  - Verificação de coerência e naturalidade
  - Remoção de exemplos problemáticos

O que são Prompt Templates?

- Definição básica
  - Estruturas predefinidas para formatar inputs
  - Padrões consistentes de comunicação
  - Guias para interação modelo-usuário
- Componentes típicos
  - Instruções do sistema
  - Delimitadores de contexto
  - Marcadores de papel/função
  - Formatação específica do modelo

Exemplos Práticos

```
# Template Alpaca
{
    "system": "Below is an instruction that describes a task. Write a response that appropriately
completes the request.",
    "prompt": "### Instruction:\n{instruction}\n\n### Response:"
}

# Template Llama 2
{
    "system": "You are a helpful, respectful and honest assistant.",
    "prompt": "<s>[INST] {instruction} [/INST]"
}

# Template Vicuna
{
    "system": "A chat between a curious user and an artificial intelligence assistant.",
    "prompt": "USER: {instruction}\nASSISTANT:"
}
```

Importância dos Templates

Benefícios Principais

- Consistência
  - Formato padronizado de inputs
  - Comportamento previsível do modelo
  - Facilita fine-tuning e avaliação
- Controle comportamental
  - Define tom e estilo das respostas
  - Estabelece restrições e limitações
  - Guia o modelo para outputs desejados

Considerações Práticas

- Escolha do template
  - Adequação ao modelo base
  - Compatibilidade com caso de uso
  - Facilidade de processamento
- Customização
  - Adaptação para necessidades específicas
  - Balanceamento entre rigidez e flexibilidade
  - Manutenção da consistência

Turnos em Datasets Conversacionais

Fundamentos

- O que são turnos?
  - Unidades de diálogo alternadas
  - Sequência de interações modelo-usuário
  - Estrutura natural de conversação
- Importância
  - Mantém contexto conversacional
  - Permite aprendizado de dinâmica dialógica
  - Crucial para comportamento natural

Por que Turnos Importam?

- Contexto sequencial
  - Modelo aprende a manter coerência
  - Entende referências anteriores
  - Desenvolve "memória" conversacional
- Dinâmica natural
  - Simula conversas reais
  - Permite respostas contextualizadas
  - Melhora qualidade da interação

Estrutura de Turnos

Anatomia de uma Conversa

```
# Exemplo de estrutura de turnos
conversation = [
    {"role": "system", "content": "Você é um assistente prestativo."},
    {"role": "user", "content": "Como funciona fotossíntese?"},
    {"role": "assistant", "content": "A fotossíntese é um processo..."},
    {"role": "user", "content": "E qual é o papel do CO2?"},
    {"role": "assistant", "content": "O CO2 é fundamental pois..."}
]
```

Elementos Importantes

- Sequência lógica
  - Progressão natural do diálogo
  - Conexão entre turnos
  - Manutenção de contexto
- Papéis claros
  - Sistema (instruções gerais)
  - Usuário (queries/instruções)
  - Assistente (respostas)

Formato ShareGPT

Origem e Contexto

- História
  - Desenvolvido pela comunidade ShareGPT
  - Adotado amplamente para fine-tuning
  - Padrão de facto para dados conversacionais
- Propósito
  - Padronização de formato
  - Facilitar compartilhamento de dados
  - Simplificar processo de treino

Estrutura Básica

```
{
    "conversations": [
        {
            "from": "human",
            "value": "Explique o que é energia nuclear."
        },
        {
            "from": "assistant",
            "value": "Energia nuclear é a energia liberada..."
        },
        {
            "from": "human",
            "value": "E quais são os riscos?"
        },
        {
            "from": "assistant",
            "value": "Os principais riscos incluem..."
        }
    ]
}
```

Treinamento com Unsloth

Vantagens do Unsloth

- Otimização de memória
  - Uso eficiente de recursos disponíveis
  - Permite treinar modelos maiores com menos hardware
- Aceleração do treinamento
  - Otimizações específicas para LLMs
  - Redução significativa no tempo de treinamento
- Suporte a diferentes arquiteturas
  - Compatível com diversos modelos populares
  - Flexibilidade na escolha do modelo base
- Facilidade de uso
  - API intuitiva e bem documentada
  - Menor curva de aprendizado

Implementação Prática

- Configuração do ambiente
  - Setup simplificado do ambiente de desenvolvimento
  - Instalação e configuração otimizadas
- Preparação do modelo
  - Carregamento e configuração inicial
  - Definição de parâmetros de treinamento
- Ajuste de hiperparâmetros
  - Otimização para caso de uso específico
  - Balanceamento entre velocidade e qualidade
- Monitoramento e avaliação
  - Acompanhamento em tempo real do treinamento
  - Métricas relevantes para avaliação
- Exportação e deploy
  - Salvamento eficiente do modelo treinado
  - Preparação para uso em produção