Classificação com Modelos de Linguagem

Escola de Verão 2025

Alexandre Teles <alexandre.teles@inctdd.org>

Índice

Introdução

Por que estudar classificação com LLMs?

Os Modelos de Linguagem Large (LLMs) revolucionaram o processamento de linguagem natural por sua capacidade única de compreender e processar texto de forma mais natural e eficiente que métodos tradicionais. Eles permitem classificação de texto com maior precisão e menor necessidade de engenharia manual de características.

Objetivos do Curso

- Compreender como LLMs processam e classificam texto: Você aprenderá os mecanismos internos que permitem aos LLMs entender e categorizar texto de forma eficiente
- Entender a codificação de informações: Exploraremos como o conhecimento é armazenado nos pesos e estruturas do modelo
- Explorar arquiteturas especializadas: Veremos como diferentes arquiteturas se adaptam a tarefas específicas

Eficácia dos LLMs na Classificação

Compreensão Contextual

- Capturam relações semânticas complexas: Os modelos entendem conexões sutis entre palavras e conceitos que vão além da simples co-ocorrência
- Entendem nuances e contexto mais amplo: Capacidade de interpretar significados baseados no contexto completo do texto
- Processamento similar ao humano: Análise que se aproxima da forma como humanos interpretam linguagem natural

Pré-treinamento Extensivo

- Treinados em vastos conjuntos de dados: Exposição a bilhões de textos que formam uma base robusta de conhecimento
- Exposição a diversos estilos e domínios: Capacidade de entender diferentes tipos de escrita e áreas de conhecimento
- Representações ricas de linguagem: Desenvolvimento de compreensão profunda de estruturas linguísticas e semânticas

Mecanismos Internos

Arquitetura Avançada

- Mecanismo de atenção eficiente: Permite ao modelo focar em partes relevantes do texto, similar à atenção humana
- Processamento paralelo: Capacidade de analisar múltiplos aspectos do texto simultaneamente
- Capacidade para sequências longas: Habilidade de manter contexto em textos extensos

Robustez

- Tolerância a ruído nos dados: Capacidade de lidar com erros de digitação, variações gramaticais e ambiguidades
- Alta capacidade de generalização: Aplicação efetiva do conhecimento em situações novas
- Desempenho consistente: Manutenção da qualidade em diferentes cenários e domínios

Codificação de Informações

Embeddings e Representações

- Conversão de tokens em vetores densos: Transformação de palavras em representações numéricas multidimensionais
- Dimensões capturam aspectos semânticos: Cada dimensão do vetor representa diferentes características do significado
- Relações codificadas como distâncias: Palavras semanticamente similares ficam próximas no espaço vetorial

Padrões Implícitos

- Correlações estatísticas nos pesos: O modelo aprende automaticamente padrões recorrentes na linguagem
- Codificação de estruturas linguísticas: Captura de regras gramaticais e construções sintáticas comuns
- Emergência de padrões naturais: Desenvolvimento orgânico de compreensão linguística durante o treinamento

Espaço Latente e Representações

Compreendendo o Espaço Latente

O espaço latente de um LLM pode ser imaginado como um vasto universo multidimensional onde cada ponto representa um conceito ou significado. Por exemplo, a palavra "banco" pode estar próxima tanto de "instituição financeira" quanto de "assento", dependendo do contexto.

Organização Semântica

Imagine um mapa onde palavras similares estão próximas umas das outras:

- "Cachorro", "gato", "hamster" formam um cluster de animais de estimação
- "Correr", "saltar", "pular" agrupam-se como verbos de movimento
- "Feliz", "alegre", "contente" compartilham região do espaço emocional positivo

Contexto e Disambiguação

O modelo aprende a navegar este espaço baseado no contexto:

- "O banco fechou minha conta" → ativa região financeira
- "O banco da praça quebrou" → ativa região de mobiliário
- Os vetores se ajustam dinamicamente conforme o contexto da frase

Estruturas Hierárquicas

Os modelos desenvolvem uma hierarquia de compreensão:

```
Nível 1: Caracteres e subpalavras
├── Nível 2: Palavras completas
    ├── Nível 3: Frases e expressões
        ├── Nível 4: Conceitos abstratos
            └── Nível 5: Relações complexas
```

Exemplo de Codificação Progressiva

Considere a frase "O gato pulou o muro":

1. Tokens individuais: [O] [gato] [pul] [ou] [o] [muro]
2. Composição sintática: [sujeito: O gato] [verbo: pulou] [objeto: o muro]
3. Semântica: [agente animado] [ação de movimento] [obstáculo físico]
4. Inferências: [capacidade de saltar] [altura significativa] [movimento intencional]

•deRelações contextuais: gato → casa → estimação → veterinário

Mecanismo de Atenção em Detalhes

Atenção Multi-Cabeça

Imagine várias pessoas lendo o mesmo texto, cada uma focando em aspectos diferentes:

- Cabeça 1: Foco em relações gramaticais
- Cabeça 2: Foco em entidades e seus atributos
- Cabeça 3: Foco em relações causais
- Cabeça 4: Foco em coerência temporal

Exemplo de Processamento Paralelo

Para a frase "O cientista publicou sua pesquisa revolucionária":

Cabeça 1 (Gramática):

```
[Determinante: O] → [Substantivo: cientista]
[Verbo: publicou] → [Pronome: sua] → [Substantivo: pesquisa]
[Adjetivo: revolucionária]
```

Cabeça 3 (Causalid

Sentence Transformers

O que são?

- Modelos especializados em embeddings de sentenças: Arquiteturas otimizadas para criar representações vetoriais de frases completas
- Otimizados para comparação semântica: Projetados especificamente para medir similaridade entre textos
- Arquitetura baseada em transformers: Utilizam a mesma base tecnológica dos LLMs, mas com foco específico

Funcionamento

- Processamento de sentenças completas: Análise holística do significado de frases inteiras
- Geração de embeddings densos fixos: Criação de vetores de tamanho constante que representam o significado completo
- Otimização para similaridade semântica: Treinamento específico para maximizar a precisão em comparações de significado

Comparação: Sentence vs. Traditional Transformers

Sentence Transformers

- Embeddings de tamanho fixo: Produzem vetores com dimensões consistentes, independente do tamanho do texto
- Otimizados para similaridade: Especialmente eficientes em tarefas de comparação e busca
- Processamento mais rápido: Menor complexidade computacional por terem objetivo específico
- Menor consumo de memória: Requisitos reduzidos por não manterem estados intermediários complexos

Transformers Tradicionais

- Embeddings de tamanho variável: Geram representações que variam com o tamanho da entrada
- Foco em compreensão geral: Projetados para entender e gerar linguagem de forma mais ampla
- Processamento token a token: Análise detalhada de cada elemento do texto
- Maior flexibilidade de uso: Adaptáveis a uma variedade maior de tarefas

Arquitetura Especializada

Diferente dos LLMs tradicionais, Sentence Transformers focam em criar uma "fotografia semântica" da frase:

```
Entrada: "O café está quente"
↓
Processamento: Média ponderada dos tokens contextualizados
↓
Saída: Vetor único [0.2, -0.5, 0.8, ...] (ex: 384 dimensões)
```

Otimização de Similaridade

O treinamento otimiza distâncias no espaço vetorial:

- Frases similares: "O café está quente" ≈ "A bebida está em alta temperatura"
- Distância vetorial pequena: cos(vec1, vec2) ≈ 0.92

Frases diferentes: "O café está quente" ≠ "O dia está frio"

- Distância vetorial grande: cos(vec1, vec3) ≈ 0.15

Anatomia de um Embedding

Sentence Transformer (384d):

```
Frase: "O café está quente"
→ [0.2, -0.5, 0.8, ..., 0.3]
   ↑     ↑     ↑        ↑
   |     |     |        |
Temperatura  Estado  Objeto  ...
```

Token Embeddings (LLM tradicional):

```
"O": [0.1, 0.2, ...]
"café": [0.3, -0.4, ...]
"está": [-0.1, 0.5, ...]
"quente": [0.8, 0.2, ...]
```


diferentes.rtilham componentes do campo semântico de temperatura, mas com intensidades

Aplicações Práticas de Embeddings

Busca Semântica

Exemplo de busca em uma base de documentos:

```
query = "impacto do aquecimento global"
doc1 = "mudanças climáticas afetam ecossistemas"
doc2 = "receita de bolo de chocolate"

similarity(query, doc1) = 0.82  # Alta similaridade
similarity(query, doc2) = 0.12  # Baixa similaridade
```

Agrupamento Temático

Documentos se agrupam naturalmente no espaço vetorial:

```
Cluster 1: Meio Ambiente
- mudanças climáticas
- preservação ambiental
```

O espaço vetorial

Aplicações Práticas

Sentence Transformers em Ação

- Busca de documentos similares: Encontrar textos semanticamente relacionados em grandes bases de dados
- Agrupamento de textos: Organização automática de documentos por similaridade de conteúdo
- Detecção de duplicatas: Identificação de conteúdo similar mesmo com diferenças na formulação

Vantagens Práticas

- Implementação mais simples: APIs mais diretas e menor necessidade de configuração
- Menor custo computacional: Recursos de hardware mais modestos para operação
- Resultados comparáveis: Desempenho próximo aos transformers completos em tarefas específicas

Considerações Finais

- Menos flexíveis que LLMs completos: Limitados a tarefas específicas de embedding e similaridade
- específicos ou técnicosing: Podem requerer ajustes para domínios muitopreensão