Explicação do Pipeline de Classificação de Texto com Embeddings

Pipeline simplificada

1. Combinação de campos (para todos os casos):
   'titulo': ['Titulo'],
   'texto': ['Noticia'],
   'subtitulo': ['Subtitulo'],
   'titulo_subtitulo': ['Titulo', 'Subtitulo'],
   'completo': ['Titulo', 'Subtitulo', 'Noticia']
2. Extração de features (embeddings): transformação de texto em vetores densos usando modelos pré-treinados
3. Divisão dos dados: 80/20 estratificado
4. Classificação: Treinamento de modelos tradicionais de Machine learning com os embeddings baseados na classe do Fake Recogna

Modelos de embedding usados

Modelos Leves (<300M de parâmetros)

all-MiniLM-L6-v2 (Sentence-Transformers)

- Parâmetros: 22M
- Dimensão: 384
- Arquitetura: MiniLM (6 layers)
- Características: Modelo mais popular (17M+ downloads), treinado em 1B+ pares de sentenças

nomic-embed-text-v1.5 (Nomic AI)

- Parâmetros: 137M
- Dimensão: 768 (Matryoshka flexível: 768→512→256→128)
- Arquitetura: Modern BERT com long context
- Características: Contexto de 8192 tokens, task-specific prefixes, open source auditável

gte-modernbert-base (Alibaba NLP)

- Parâmetros: 149M
- Dimensão: 768
- Arquitetura: ModernBERT (primeira arquitetura BERT moderna de 2024)
- Características: RoPE, local-global alternating attention, Flash Attention, contexto de 8192 tokens, treinado em 2T tokens

Modelos Médios (300M - 600M de parâmetros)

granite-embedding-278m-multilingual (IBM Research)

- Parâmetros: 278M
- Dimensão: 768
- Arquitetura: XLM-RoBERTa-like (12 layers)
- Características: 12 línguas incluindo português, Apache 2.0 license, enterprise-ready

KaLM-embedding-v2.5 (HIT-SCIR)

- Parâmetros: 500M
- Dimensão: 896 (Matryoshka: 896→512→256→128→64)
- Arquitetura: Qwen2-0.5B com atenção bidirecional
- Características: Suporta até 32k tokens, focal-style reweighting, SOTA em chinês <1B

Jina-embeddings-v3 (Jina AI)

- Parâmetros: 570M
- Dimensão: 1024 (Matryoshka até 32 dim)
- Arquitetura: XLM-RoBERTa com task-specific LoRA
- Características: Task adapters, late chunking, 89 línguas, contexto de 8192

multilingual-E5-Large (Microsoft)

- Parâmetros: 560M
- Dimensão: 1024
- Arquitetura: XLM-RoBERTa-large (24 layers)
- Características: 100 línguas, prefixos query:/passage:, MTEB benchmark leader

persian-embeddings (HeydariAI)

- Parâmetros: 560M

- Dimensão: 1024
- Arquitetura: XLM-RoBERTa fine-tuned para Persian
- Características: Bilíngue Persian↔English, treinado em corpus massivo de notícias persas

Modelos Grandes (>600M de parâmetros)

SFR-Embedding-Mistral (Salesforce Research)

- Parâmetros: 7B
- Dimensão: 4096
- Arquitetura: Mistral-7B adaptado para embeddings
- Características: #1 no MTEB geral, instruções específicas, maior dimensionalidade

SERAFIM-900M-Portuguese-PT (PORTULAN/CLARIN.PT)

- Organização: Universidade de Lisboa - PORTULAN CLARIN
- Parâmetros: 900M
- Dimensão: 1536
- Arquitetura: Foundation model da família Albertina, encoder-only
- Base: Treinamento from-scratch em corpus português massivo

Outros modelos classicos também, Word2Vec, BERT e TF-IDF

Modelos via API

OpenAI text-embedding-3-small

- Dimensão: 1536 (configurável via Matryoshka)
- Características: Modelo proprietário, estado da arte em benchmarks

Google Embedding-001 (Vertex AI)

- Dimensão: 768
- Características: Otimizado para retrieval e clustering

Processo de Geração de Embeddings

Processo para gerar embedding: Texto Original → Pré-processamento (Base já esta pre-processado) → Tokenização → Modelo → Embedding (vetor denso)
Pre processamento no mesmo banco de dados dividido em 2:

- 1 com lemmatização + remoção de stopwords
- 2 com lemmatização + remoção apenas de caracteres . , / - !

Tokenização com truncamento automático para MAX_SEQ_LENGTH configurado. Cada modelo usa seu tokenizer específico (WordPiece, SentencePiece, BPE)

Normalização L2

Classificadores: Todos os embeddings com as classes foram treinados sobre as mesmas circunstâncias

1. SVM SVC(kernel='linear', random_state=42)
2. Random Forests RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
3. Logistic Regression LogisticRegression(max_iter=1000, random_state=42, n_jobs=-1)
4. Metricas por útlimo, basicamente eu usei F1 e accuracy como resumo, porém calculei as outras tbm

Resumo gerado pelo claude do processo

1. Carregamento dos Dados
   ↓
2. Para cada MODELO:
   ↓
   2.1. Carrega modelo pré-treinado na GPU
   ↓
   2.2. Para cada DATASET:
   ↓
   2.2.1. Para cada COMBINAÇÃO DE CAMPOS:
   ↓
   a) Combina campos de texto
   b) Gera embeddings (ou carrega do cache)
   c) Split treino/teste (80/20 estratificado)
   d) Normaliza embeddings
   ↓
   e) Para cada CLASSIFICADOR:
   - Treina modelo
   - Prediz no conjunto de teste
   - Calcula métricas (Acc, Prec, Rec, F1)

- Salva resultados

↓

f) Salva checkpoint

↓

3. Agrega todos os resultados

↓

4. Gera relatórios comparativos (CSV)