

Exercício Prático – Kauan

Análise comparativa de embeddings e clustering em bases de notícias (PT-6 e 20NG-6)

1. Contexto

Atualmente existem diversas formas de representar textos como vetores numéricos: desde representações lexicais clássicas (como TF-IDF) até embeddings semânticos pré-treinados baseados em Transformers (SBERT, GTE, BGE, entre outros), além de embeddings derivados de LLMs via API.

Este exercício tem como objetivo comparar, de forma sistemática, diferentes tipos de embeddings aplicados a duas bases de notícias com 6 classes cada (PT-6 em português e 20NG-6 em inglês), avaliando como essas representações influenciam o desempenho de algoritmos de clustering, a qualidade das visualizações em 2D e algumas análises adicionais.

2. Objetivos do exercício

Ao final deste trabalho, o(a) aluno(a) deverá ser capaz de:

- Construir e comparar várias representações vetoriais para textos (lexicais e semânticas).
- Aplicar 4 ou 5 algoritmos de clustering sobre esses vetores, em duas bases (PT-6 e 20NG-6).
- Avaliar os resultados usando métricas internas e externas, comparando: (a) efeito do embedding, (b) efeito do algoritmo de clustering, (c) diferenças entre português (PT-6) e inglês (20NG-6).
- Produzir visualizações 2D (PCA, t-SNE, UMAP) que ajudem a interpretar os clusters e a separação entre classes.
- Elaborar um relatório organizado com tabelas e figuras, discutindo os achados mais importantes.

3. Bases de dados (PT-6 e 20NG-6)

Serão utilizadas duas bases de notícias, cada uma com 6 classes:

a) PT-6 (Português): conjunto de notícias em português com 6 classes. A base será fornecida em formato CSV, com campos como, por exemplo, texto e rótulo de classe (coluna 'classe' ou similar).

b) 20NG-6 (Inglês): subconjunto da base clássica 20 Newsgroups contendo 6 categorias em inglês (por exemplo, temas de tecnologia, esportes, ciência etc.). Pode ser obtida via scikit-learn ou script auxiliar.

As classes verdadeiras serão utilizadas apenas para avaliação (métricas externas como ARI, NMI e pureza). Não haverá treinamento supervisionado; o foco é clustering não supervisionado.

4. Embeddings a serem comparados

O exercício deve incluir, no mínimo, quatro tipos de embeddings (pode-se adicionar outros, se houver tempo):

- 1) TF-IDF + SVD (baseline lexical):
- Usar TfidfVectorizer (scikit-learn) com n-grams (1-2 ou 1-3),
- Limitar número de características (por exemplo, 50.000 termos),
- Reduzir dimensionalidade com TruncatedSVD (por exemplo, 300 dimensões).
- 2) SBERT (Sentence Transformer):
- Modelo multilíngue, por exemplo: 'sentence-transformers/paraphrase-multilingual-mpnet-base-v2'.
- Gerar um embedding por texto usando a biblioteca sentence-transformers.

3) GTE (General Text Embeddings):

- Modelo open-source multilíngue, por exemplo: 'thenlper/gte-multilingual-base'.
- Utilizar via Hugging Face Transformers ou sentence-transformers, configurando batch size adequado.

4) BGE (BAAI General Embedding):

- Modelo BGE-m3 ou similar (multilingual), também disponível no Hugging Face.
- Tipicamente otimizado para tarefas de retrieval, mas aplicável a clustering.

5) (Opcional) Embedding via LLM/API:

- Se houver acesso a alguma API (OpenAI, Gemini, etc.), gerar embeddings para pelo menos uma das bases.
- Este item é opcional e pode ser tratado como experimento extra.

5. Algoritmos de clustering

Para cada combinação (dataset, tipo de embedding), aplicar 4 ou 5 algoritmos de clustering, por exemplo:

- k-means (scikit-learn): definir k=6 (número de classes), inicialização k-means++ e múltiplas inicializações.
- Gaussian Mixture Models (GMM): usar 6 componentes e diferentes inicializações, avaliando convergência.
- Agglomerative clustering: usar linkage Ward ou complete, com número de clusters fixado em 6.
- DBSCAN: experimentar diferentes valores de eps e min_samples; o número de clusters é determinado pelos dados.
- HDBSCAN ou Spectral Clustering: escolher pelo menos um desses para explorar diferenças em relação aos métodos anteriores.

A ideia é aplicar sistematicamente os mesmos algoritmos em todos os embeddings e bases, permitindo comparação cruzada.

6. Métricas de avaliação

Para cada combinação (dataset, embedding, algoritmo de clustering), calcular as seguintes métricas:

- Métricas externas (usando as classes verdadeiras):
- ARI (Adjusted Rand Index);
- NMI (Normalized Mutual Information);
- Pureza (Purity).
- Métrica interna:
- Silhouette (quando fizer sentido para o número de clusters obtido).

Sugestão: implementar uma função em Python que receba os rótulos verdadeiros (y_{true}), os rótulos de cluster (y_{pred}) e, quando necessário, os vetores de embedding X, retornando todas essas métricas organizadas em um dicionário ou DataFrame.

Organizar os resultados em tabelas, separando PT-6 e 20NG-6, por exemplo:

- Tabela para PT-6: linhas = embeddings, colunas = métricas por algoritmo.
- Tabela para 20NG-6: mesma estrutura.

7. Visualizações em 2D e análise qualitativa

Para interpretar melhor a estrutura dos dados e dos clusters, gerar visualizações 2D dos embeddings, utilizando métodos como:

- PCA (Principal Component Analysis);

- t-SNE (t-distributed Stochastic Neighbor Embedding);
- UMAP (Uniform Manifold Approximation and Projection).

Para cada embedding (ou pelo menos para os principais, como TF-IDF+SVD e BGE), produzir dois tipos de gráficos em 2D:

- a) Pontos coloridos pela classe verdadeira (rótulos reais da base).
- b) Pontos coloridos pelo rótulo de cluster (por exemplo, saída do k-means).

Essas visualizações permitem comparar qualitativamente a separação entre classes e clusters, bem como as diferenças entre PT-6 e 20NG-6.

8. Etapas sugeridas de implementação (roteiro de trabalho)

Sugestão de roteiro para organizar o desenvolvimento do exercício:

- 1) Preparar o ambiente:
- Criar repositório ou pasta do projeto (por exemplo, com subpastas 'data', 'notebooks', 'results', 'figures').
- Instalar pacotes necessários: scikit-learn, sentence-transformers, umap-learn, hdbscan, etc.
- 2) Carregar e inspecionar as bases PT-6 e 20NG-6:
- Verificar número de amostras, distribuição de classes, exemplos de textos.

3) Implementar pipeline de embeddings:

- Funções separadas para: TF-IDF+SVD, SBERT, GTE, BGE, e (opcionalmente) LLM via API.
- Salvar embeddings em arquivos (por exemplo, .npy ou .pkl) para evitar recálculo.

4) Rodar algoritmos de clustering:

- Para cada dataset e embedding, rodar k-means, GMM, Agglomerative, DBSCAN, etc.
- Armazenar os rótulos de cluster obtidos.

5) Calcular métricas:

- Implementar uma função para calcular ARI, NMI, pureza e silhouette.
- Gerar DataFrames ou tabelas com resultados consolidados.

6) Gerar visualizações 2D:

- Projeções PCA, t-SNE e UMAP para alguns embeddings, com cores por classe e por cluster.

7) Escrever o relatório:

- Sintetizar metodologia, principais resultados, tabelas, figuras e conclusões.

9. Dicas de código e bibliotecas úteis

Algumas referências e bibliotecas úteis para o desenvolvimento do exercício:

- Scikit-learn (clustering, métricas, PCA, t-SNE, acesso a 20 Newsgroups): <https://scikit-learn.org/>
- Sentence-Transformers (SBERT e modelos relacionados): <https://www.sbert.net/>
- Modelos GTE e BGE (Hugging Face Hub): <https://huggingface.co/>
- UMAP (umap-learn): <https://umap-learn.readthedocs.io/>
- HDBSCAN: <https://hdbscan.readthedocs.io/>

Sugestão de organização em Python:

- Criar um módulo ou notebook com funções do tipo:
- `build_tfidf_svd_embeddings(texts) -> ndarray;`
- `build_sbert_embeddings(texts, model_name) -> ndarray;`
- `run_clustering_kmeans(X, k) -> labels;`
- `compute_metrics(y_true, y_pred, X) -> dict com ARI, NMI, pureza, silhouette.`

Manter o código bem comentado, explicando decisões de parâmetros (número de dimensões, eps do DBSCAN, etc.).

10. Entregáveis

Ao final do exercício, espera-se a entrega de:

- Código-fonte (notebooks ou scripts) bem organizados e comentados, contendo a implementação do pipeline de embeddings, clustering e avaliação.
- Relatório em PDF ou Word (4 a 8 páginas) com:
- Introdução ao problema e aos embeddings utilizados;
- Seção de metodologia (dados, embeddings, algoritmos, métricas);
- Seção de resultados (tabelas e figuras principais);
- Discussão e conclusões (comparações entre embeddings, algoritmos e bases PT-6/20NG-6).
- Pasta com resultados: tabelas (CSV) de métricas e imagens das visualizações 2D.

11. Perguntas-guia para reflexão

As questões abaixo devem orientar a análise e aparecer, pelo menos em parte, discutidas no relatório final:

- Qual fator impacta mais o desempenho de clustering: o tipo de embedding ou o algoritmo de clustering?

- Embeddings semânticos (SBERT, GTE, BGE) melhoram de forma consistente o desempenho em relação ao TF-IDF+SVD nas duas bases?
- Há diferenças claras de comportamento entre PT-6 e 20NG-6 em termos de separação de classes e clusters?
- Qual combinação (dataset, embedding, algoritmo) parece mais adequada para uma aplicação real de agrupamento de notícias?

Prof. José Alfredo Costa