

# **Projetos em Ciência de Dados**

## **Versão Final — A2**

**Autores:** Ana Carolina Erthal, Cristiano Larréa, Felipe Lamarca, Guilherme de Melo, Paloma Borges, Raul Lomonte  
**Docente:** Thiago Guerrero

Escola de Matemática Aplicada  
FGV EMAp

Rio de Janeiro  
2024.1

# Sumário

<b>Sumário</b>	<b>I</b>	
<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Status na A1</b>	<b>1</b>
<b>3</b>	<b>Shortcomings e progresso</b>	<b>2</b>
3.1	Multi-vector indexing	2
3.2	Large Language Model (LLM)	2
3.3	Avaliações	3
<b>4</b>	<b>Avaliação de performance</b>	<b>3</b>
<b>5</b>	<b>Conclusões</b>	<b>6</b>

# 1 Introdução


O objetivo deste projeto é o desenvolvimento do Podflix, um sistema de recomendação de episódios do podcast *This American Life*. Buscamos tornar possível, através de Word Search:

- A partir de um episódio, recomendar outros episódios semelhantes
- Dada uma sequência de tokens (query), retornar episódios mais relevantes

Nosso banco de dados para a execução do projeto conta com a transcrição completa de 825 podcasts de aproximadamente 1 hora de duração, além de metadados como título, descrição curta, data de lançamento e áudio do podcast.

O objetivo principal, além da obtenção de uma aplicação funcional para recomendação de podcasts, é explorar as etapas para a construção de um projeto end-to-end em Ciência de Dados, explorando modelos de Text Search e suas aplicações, metrificação de resultados e a própria biblioteca PyVespa, muito útil a esse contexto.

O relatório está subdividido nas seguintes seções: na próxima seção, comentados brevemente sobre o status do trabalho na ocasião da entrega da A1; em seguida, na seção de *Shortcomings* e progresso, indicamos os problemas da entrega da A1 e de que maneira eles foram endereçados na entrega final. A penúltima seção traz os resultados de avaliação de performance, feitas as devidas modificações, e a seção final conclui o trabalho.

O código fonte e instruções para executar a aplicação estão disponíveis no Github 

## 2 Status na A1

Nesta seção, retomamos brevemente os principais pontos do trabalho na primeira entrega, em particular: o tratamento e segmentação dos dados, os modelos utilizados, métricas de avaliação e principais resultados.

**Tratamento e segmentação dos dados** No desenvolvimento da A1, o tratamento dos dados foi simples e envolveu apenas a inclusão do título e da descrição dos podcasts à transcrição completa, maximizando as informações disponíveis. Na prática, utilizávamos todo o longo conteúdo dos podcasts para a realização das buscas, o que implicava o mapeamento das informações em espaços latentes sub-otimizados.

**Modelos implementados** Utilizando o pyVespa, utilizamos três modelos de busca: (i) **BM25**, um modelo determinístico que calcula a relevância de um podcast de acordo com a frequência relativa dos termos; (ii) **modelo semântico**, que mapeava todo o conteúdo dos podcasts em um espaço latente de dimensão 384 e comparava o vetor resultante com o vetor da query, de mesma dimensão, para fazer o ranqueamento; e (iii), um **modelo híbrido** (ou Fusion), que combina as duas abordagens.

**Métricas de avaliação** Para avaliar os modelos, na ocasião da entrega da A1, foi necessário atribuir manualmente *scores* aos podcasts de acordo com sua relevância em relação à query. Criamos uma lista de queries sobre assuntos diversos e avaliamos qualitativamente o retorno dos primeiros podcasts ranqueados, atribuindo *scores* entre 0 e 5<sup>1</sup>.

As métricas utilizadas foram o Normalized Discounted Cumulative Gain (NDCG), Precisão e Ranking Stability. Mais detalhes sobre cada uma das métricas estão descritos no relatório da A1.

**Principais resultados** O modelo Fusion foi superior aos outros dois modelos na maioria das métricas. No entanto, o BM25 performou melhor que o modelo semântico em todas as métricas.

## 3 Shortcomings e progresso

No desenvolvimento da versão final desse trabalho para a A2, distribuímos nossos esforços em resolver alguns shortcomings da versão parcial entregue na A1, com foco nos feedbacks recebidos.

### 3.1 Multi-vector indexing

Entre os pontos de destaque no feedback está a necessidade de alterar o uso de um podcast inteiro para realização de buscas, já que temos um contexto grande demais, dificultando a realização da busca. Para resolver esse problema, passamos a utilizar multi-vector indexing, realizando quebras no texto a fim de utilizar contextos menores para a busca.

Nos podcasts temos separações bastante claras nas falas de cada indivíduo, e portanto utilizamos cada uma dessas falas como uma quebra no texto. Assim, obtivemos uma média de 260 tokens por vetor, com máximo de 470, bastante satisfatórios para os modelos utilizados. No paper que apresenta o modelo utilizado para o semantic search (E5-small, padrão do Vespa) não temos a informação explícita de média de tokens por janela de contexto nos dados de treinamento, mas temos a janela máxima de 512 tokens.

Essa mudança tornou a aplicação mais pesada pela quantidade de vetores de busca, gerando necessidade de migrarmos a aplicação do ambiente local para o Vespa Cloud.

### 3.2 Large Language Model (LLM)

Na primeira entrega, realizamos muitas tarefas manualmente, como a definição de queries de busca e avaliação de resultados. Essa escolha, além de lenta e dispendiosa (sem muito ganho de aprendizado), nos levou a problemas como escolhas não otimizadas de queries de busca. Por exemplo, poderíamos escolher uma query que não representasse bem

---

<sup>1</sup> Para minimizar os efeitos da subjetividade, o score de relevância final era o resultado de uma média entre as avaliações feitas por dois integrantes do grupo.

nenhum podcast, e consequentemente atribuiríamos notas baixas aos retornos, penalizando os modelos de search indevidamente.

Para corrigir esse problema, passamos a empregar uma LLM na tarefa de criar resumos de podcasts em uma frase simples para uso como query.

O modelo de LLM escolhido foi o Llama 3 8b 4bit pré-treinado da biblioteca Unsloth com fine-tuning para instruções. Utilizar a Unsloth também nos trouxe um ganho de velocidade e memória, já que a biblioteca traz otimizações para que o modelo tenha uma execução 2.5x mais rápida utilizando 70% menos memória.

### 3.3 Avaliações

A decisão de substituir as tarefas manuais por automatizações também teve grande impacto nas métricas utilizadas para avaliação dos modelos. Como deixamos de atribuir *scores* manuais, passamos a empregar métricas também automatizadas.

Estendemos também a cobertura dos testes realizados, passando a realizar as seguintes comparações:

**Testes com diferentes textos do lado da query:** ao invés de testarmos utilizar queries definidas manualmente para busca, passamos a realizar comparações com versões utilizando queries automatizadas pela LLM, título e descrição.

**Testes com diferentes textos do lado da busca:** ao invés de testarmos utilizar apenas o texto completo para busca (título + descrição + corpo), passamos a realizar comparações com versões utilizando apenas título e título + descrição também.

**Testes de multi-vector indexing:** mantivemos ainda os modelos que não utilizavam multi-vector indexing para comparação de métricas de resultado utilizando ou não essa quebra em vetores de contexto.

A nova proposta é, utilizando os novos **testes com diferentes textos do lado da query**, automatizar a checagem do podcast retornado, já que a partir do uso do resumo da LLM, título e descrição, sabemos qual o podcast que deveria ser retornado pela busca, e podemos medir de forma automatizada esse retorno. As métricas serão melhor definidas na próxima seção.

## 4 Avaliação de performance

Utilizamos três métricas para avaliação dos modelos concebidos, todas elas realizando avaliações a partir do retorno obtido quando realizamos uma query a partir do título, descrição ou resumo da LLM de um podcast específico. São elas o HitRate, HitRate@10 e MRR:

1. **HitRate:** avalia se o primeiro retorno de uma query (rank 1) é o podcast buscado:

$$\frac{\text{Número de queries em que o rank 1 é o podcast buscado}}{\text{Total de queries}}$$

2. **HitRate@10:** avalia se o podcast buscado está entre os 10 primeiros retornos de uma query:

$$\frac{\text{Número de queries em que o podcast buscado está nos top 10 ranks}}{\text{Total de queries}}$$

3. **Mean Reciprocal Rank (MRR):** Avalia a média dos ranks do podcast buscado em um conjunto de queries (considerando  $Q$  queries):

$$\frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{Rank do podcast buscado pela query } i}$$

Essas métricas foram escolhidas pela possibilidade de automatizar completamente a avaliação de modelos. Utilizar algumas outras possibilidades, como Precision@k e Recall@k, geraria a necessidade de avaliar, para os primeiros  $k$  ranks obtidos a partir de uma query, quais são bons retornos (ao invés de levarmos em conta apenas o podcast que gerou o resumo/possui o título ou descrição).

Os resultados de acordo com as métricas definidas estão a seguir:

Vamos iniciar avaliando se a utilização de MVI nos trouxe ganho de performance. Para isso, avaliaremos os modelos Semantic (Sem.) e Fusion (Fus.), já que não utilizamos MVI no BM25, utilizando o podcast completo para busca (título+descrição+corpo), e faremos variações na query:

Modelo	HR	HR@10	MRR	Modelo	HR	HR@10	MRR
<b>Sem.</b>	47.7%	66.5%	56.8%	<b>Sem.</b>	57.9%	80.9%	67.3%
<b>Sem. MVI</b>	78.8%	96.2%	86.0%	<b>Sem. MVI</b>	19.6%	46.5%	32.0%
<b>Fus.</b>	64.2%	86.9%	72.8%	<b>Fus.</b>	71.9%	94.0%	80.7%
<b>Fus. MVI</b>	88.2%	99.3%	93.2%	<b>Fus. MVI</b>	44.3%	78.3%	56.4%

Tabela 1 – Métricas para modelos utilizando ou não MVI, utilizando resumos da LLM como query

Tabela 2 – Métricas para modelos utilizando ou não MVI, utilizando títulos como query

Os resultados foram bastante interessantes! Observe que o uso de Multi-Vector Indexing nos trouxe resultados muito positivos quando utilizamos resumos da LLM como queries, aumentando o desempenho de acordo com todas as métricas. Por outro lado, quando utilizamos o título como query, utilizar MVI trouxe piora à performance! Acreditamos que esse fenômeno se deve aos resumos da LLM serem ricos em palavras-chave e conceitos que estão presentes em várias partes do texto completo. Como resultado, o MVI, que funciona bem com consultas mais ricas em contexto, pode aproveitar a diversidade de informações para encontrar correspondências mais relevantes.

É interessante observar, também, que obtivemos um resultado melhor utilizando o modelo Fusion do que o Semantic, assim como na versão para a A1, independentemente da query usada ou da variação de MVI. Vamos verificar mais a fundo a influência da escolha da query em cada modelo, mas agora avaliando apenas para os modelos BM25 e Fusion, de melhor desempenho:

Variações BM25	HR	HR@10	MRR
Query Resumo LLM	91.2%	98.7%	94.5%
Query Descrição	73.4%	93.8%	81.8%
Query Título	88.8%	97.8%	92.8%

Tabela 3 – Métricas de Performance para BM25 variando a query

Variações Fusion	HR	HR@10	MRR
Query Resumo LLM	64.2%	86.9%	72.8%
Query Descrição	88.8%	97.8%	92.8%
Query Título	71.9%	94.0%	80.7%
Query Resumo LLM + MVI	88.2%	99.3%	93.2%
Query Descrição + MVI	72.1%	94.5%	80.4%
Query Título + MVI	44.3%	78.3%	56.4%

Tabela 4 – Métricas de Performance para Fusion variando a query e uso de MVI

Observe que, novamente, o BM25 se mostrou um modelo com métricas de maior destaque, obtendo consistentemente boas avaliações. Na versão utilizando resumos a partir da LLM, obtivemos os valores muito altos nas três métricas utilizadas. Esse resultado parece bastante razoável, sendo uma causa bastante provável a quantidade de nomes próprios utilizados nos resumos da LLM, que facilitam a busca do BM25 pelo podcast exato.

Esse modelo (BM25+Resumo LLM) compete apenas com a versão Fusion+Resumo LLM+MVI, em que também obtemos métricas bastante altas. Por fim, façamos uma avaliação rápida alterando os textos em que realizamos a busca. Avaliaremos aqui ao uso de (título + descrição) e (título + descrição + corpo) para busca, mantendo fixa a query como resumo da LLM:

Modelos	HR	HR@10	MRR
BM25 título+descrição	27.4%	48.4%	38.6%
BM25 título+descrição+corpo	91.2%	98.7%	94.5%
Fusion título+descrição	29.1%	48.7%	40.1%
Fusion título+descrição+corpo	64.2%	86.9%	72.8%

Tabela 5 – Métricas de Performance variando o texto em que a busca é realizada, utilizando como queries resumos da LLM

Esses resultados, por fim, nos mostram como o melhor texto para realizar a busca de fato é a versão completa, utilizando todas as informações do podcast (título+descrição+corpo).

Possuir um texto mais completo possibilita buscas melhores, enquanto textos reduzidos, apesar de acelerarem a busca, podem dificultar a busca pela falta de contexto suficiente.

## 5 Conclusões

Os resultados deste trabalho nos levam à conclusão que diferentes combinações de modelos de busca e tipos de queries geram variações significativas no desempenho dos sistemas de text search. Em particular, observamos que o uso de Multi-Vector Indexing (MVI) melhora substancialmente a performance quando utilizamos resumos gerados por Large Language Models (LLMs) como queries, pois esses resumos capturam uma variedade de palavras-chave e conceitos que aumentam a relevância das correspondências. Por outro lado, ao utilizar títulos dos podcasts como queries, por exemplo, a aplicação de MVI resultou em piora de performance devido à natureza mais específica e menos diversificada dos títulos.

Portanto, a partir desses comportamentos durante a avaliação de modelos, destaca-se a importância de analisar cuidadosamente as escolhas feitas na implementação de um modelo de text search. A decisão de utilizar MVI, por exemplo, deve ser ponderada em função do tipo de queries que serão mais frequentes no uso do sistema. Em nossa aplicação, como o objetivo principal era o aprendizado, mantivemos todas as possibilidades na aplicação. Assim, concluímos que a combinação adequada de técnicas e modelos, ajustada ao tipo de dado e ao objetivo da busca, pode otimizar significativamente os resultados obtidos.