Projetos em Ciência de Dados
Status Update — A1

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

Escola de Matemática Aplicada
FGV EMAp

Rio de Janeiro
2024.1

Sumário

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.

2 Desenvolvimento

2.1 Tratamento de dados

Como nosso objetivo era utilizar o texto completo dos episódios, nossa etapa de tratamento e limpeza foi bastante superficial. Adicionamos o título e a descrição do podcast à transcrição completa, para maximizar as informações disponíveis. Além disso, removemos as flags da transcrição que determinavam quem está falando, já que identificamos que isso poderia ser um problema para a detecção de palavras (numa query sobre a cerimônia do Oscar, por exemplo, um episódio com algum convidado chamado Oscar seria indevidamente recomendado).

2.2 Modelos implementados

Para o recurso de busca e ranking de podcasts, utilizamos três diferentes modelos disponibilizados pelo pyVespa. Em todas as abordagens, para fazer a busca, consideramos toda a informação que temos do podcast: o título, a descrição e a transcrição completa. Nosso primeiro modelo é baseado no **BM25**, e calcula a relevância de um podcast relacionando os termos da sua transcrição e da consulta segundo uma fórmula fechada. O segundo é um **modelo Semântico**, onde criamos um tensor que representa cada um dos podcasts, atualmente com dimensão 384, e ao fazer uma consulta, o modelo transforma o texto da consulta em um tensor de mesma dimensão e utiliza um algoritmo de vizinhos próximos aproximado para criar o ranking. Nosso último modelo para busca e ranking

é um **modelo híbrido**, que chamamos de Fusion, que realiza uma combinação das duas abordagens comentadas anteriormente, a fórmula fechada do BM25 e a distância entre tensores do modelo semântico.

Para nosso segundo desafio, que seria a recomendação de podcasts partindo de um podcast inicial, tentamos utilizar os três modelos comentados anteriormente, passando a transcrição completa do podcast inicial como uma consulta (query) para achar o ranking dos mais relevantes a essa consulta, mas tivemos problemas com os modelos baseados na fórmula fechada utilizando palavras (BM25 e Fusion, que utiliza o BM25). A query para consulta era grande demais para o cálcula em fórmula fechada, ultrapassando o limite de 2000 caracteres. Com o modelo semântico, no entanto, obtivemos o resultado esperado, pois ele depende apenas da vetorização do podcast uma vez, para então comparar o resultado com a vetorização dos outros podcasts, e o número de palavras não se torna impeditivo.

3 Front-end

A aplicação web está estruturado como uma aplicação dentro de um projeto Django. Para tanto, foram criadas 3 views principais:

- uma página inicial;
- uma página de recomendação, que roda o modelos de recomendação de outros episódio com base na transcrição de um episódio;
- uma página de busca por assunto, com campo aberto, que roda o modelo de recomendação de episódios com base em uma expressão de query.

Em cada página de busca, você pode também pode optar entre os modelos disponíveis para ser usados e assim, ver resultados diferentes conforme o modelo escolhido. Para comunicação com banco de dados, os dados foram convertidos de .csv para um banco de dados SQLite, pois o Django já possui integração nativa com esse banco de dados.

4 Avaliação de performance

Como estamos tratando de um problema de aprendizado não-supervisionado, metrificar os resultados obtidos depende de técnicas menos clássicas, já que não contamos com *true labels* para comparar o desempenho real com uma performance ideal.

Por isso, definimos 2 scores para os modelos: um **score de pontuação** e um **score binário**. Na prática, criamos uma lista de 11 queries sobre assuntos diversos, e avaliamos o retorno dos cinco primeiros podcasts ranqueados por cada query e modelo, atribuindo notas entre 0 e 5 para os retornos de acordo com uma avaliação qualitativa

da relação entre cada query e o conteúdo dos podcasts retornados (score de pontuação). Para minimizar os efeitos da subjetividade, o score de relevância final é o resultado de uma média entre as avaliações feitas por dois integrantes do grupo. O score binário é 1 para pontuações maiores ou iguais a 3, e 0 para as pontuações menores que 3.

4.1 Avaliação do modelo de pesquisa por assunto

Munidos de avaliações qualitativas e sabendo que o retorno dos modelos é um ranking de k podcasts ordenados pela importância atribuída a eles pelo modelo, uma das métricas que utilizamos é a Discounted Cumulative Gain (DCG), que mede a qualidade de uma lista ranqueada da seguinte forma:

$$\text{DCG@k} = \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{\log_2(i+i)},$$

onde rel_i é a relevância do podcast na posição i do ranking. O objetivo da métrica é atribuir um score ao modelo a partir dos scores de pontuação de seus retornos, atribuindo mais peso aos scores nas primeiras posições do ranking. A métrica final é resultado da média do DCG de todas as queries e subsequente normalização, dividindo o resultado pelo Ideal Discounted Cumulative Gain (IDCG), isto é, o valor da métrica caso o indicador de relevância fosse máximo para todos os podcasts retornados. Com isso, reportamos o Normalized Discounted Cumulative Gain (NDCG).

Utilizando o score binário avaliado, foi possível calcular a Precisão dos modelos. Como discutiremos na seção 5, no entanto, a metodologia utilizada para avaliação — em que damos notas ao retorno dos modelos — não nos permite o cálculo de métricas como Recall e F1-Score. Trata-se de um possível ponto de ajuste para os próximos passos do trabalho.

A terceira métrica utilizada para avaliação foi a Ranking Stability. Para pequenas modificações em uma query, espera-se que os resultados da busca sejam semelhantes. Dessa maneira, ela reflete justamente a frequência com que os outputs do modelo se repetem de acordo com pequenas variações nas queries — isto é, quanto maior o valor reportado, mais estável é o modelo em relação a essas variações. Para calculá-la, fixamos os dez primeiros podcasts retornados por cada modelo a partir de uma query, fazemos pequenas alterações nessa query e quantificamos quantos dos podcasts retornados se mantiveram em relação à query original. Fazemos isso para todas as 11 queries definidas, calculamos a frequência de repetição para 4 versões modificadas de cada uma delas e calculamos uma média dos resultados finais.

Os resultados de cada um dos modelos de acordo com as métricas definidas estão a seguir:

Modelo	NDGC	Precision	Ranking Stability
BM25	67.3%	59.5%	66.0%
Semantic	60.0%	51.2%	52.5%
Fusion	83.6%	66.3%	62.5%

Tabela 1 – Métricas de Performance: NDGC, Precision e Ranking Stability

Observamos uma dominância do modelo Fusion sobre os outros, conforme esperávamos, contando com NDGC e Precisão consideravelmente mais altos. Dentre os outros dois modelos, a performance foi dependente da métrica. Apesar da relevância do Ranking Stability, essa métrica favorece o modelo BM25 em relação aos outros, já que as variações de queries contavam com muitas palavras repetidas da query original, reduzindo a variabilidade dos retornos.

Ainda assim, podemos dizer, de modo geral, que surpreendentemnte o modelo BM25 teve uma performance superior ao Semantic. Acreditamos que esse resultado se dá principalmente pela natureza dos nossos dados. Os podcasts trazem uma informalidade ao texto que torna possível o desvio completo do tema principal do podcast com muita frequência. Isto é, é possível que os *hosts* se distraiam e passem a comentar sobre como o estúdio está quente, por exemplo, já que o ar-condicionado está quebrado. Com as menções frequentes ao calor, o podcast pode parecer relevante a uma query sobre aquecimento global. Enquanto isso, no BM25, como essas palavras exatas não foram pronunciadas, o episódio é considerado.

4.2 Avaliação do modelo de recomendação

Este modelo, na prática, pode ser avaliado de forma bastante análoga à outra funcionalidade. Isto é, se atribuirmos **scores de pontuação** e **scores binários**, podemos calcular NDCG e Precision dos modelos para essa *task*. A tarefa de atribuição de scores é bastante lenta pelo caráter dos dados, já que é necessário que dois membros do grupo passem por todo o episódio, que conta com cerca de 1 hora de transcrição, para avaliar um único retorno. Assim, temos intenção de reportar essas métricas também para a tarefa de recomendação na a entrega final do projeto.

A terceira métrica para a tarefa de recomendação será a Reciprocidade do modelo. Isto é, se o podcast B é recomendado a partir do podcast A, é esperado que o podcast A também seja recomendado a partir do podcast B. Assim, a métrica se baseia na taxa de reciprocidade de recomendação entre episódios.

5 Problemas e próximos passos

Ao longo do desenvolvimento enfrentamos algumas questões que pretendemos, após discutir mais a fundo em aula, solucionar para a entrega final do projeto. Atualmente,

contamos com apenas um modelo para recomendação de podcasts, conforme discutido na seção 2.2, e desejamos encontrar uma alternativa de redução do texto completo do podcast a uma query que possa ser utilizada pelo modelo BM25 e o Fusion nessa tarefa.

Além disso, encontramos problemas na etapa de desenvolvimento de métricas de avaliação. Como os podcasts são muito longos, e temos muitos episódios no conjunto de dados, avaliar todos os 825 podcasts atribuindo scores para cada query significa uma quantidade de trabalho manual exorbitante, que não poderia ser executada em tempo hábil. Por isso, optamos por realizar a avaliação dos primeiros k resultados de cada query ou recomendação.

Apesar de ser uma avaliação satisfatória, já que podemos avaliar os resultados relevantes de cada modelo e comparar sua performance com algumas métricas, não somos capazes de produzir algumas métricas clássicas como o Recall, que dependem da separação das samples negativas em *true negatives* e *false negatives*. Na prática, todas os resultados posteriores a k são considerados negativos para nós por não terem recebido relevância suficiente do modelo. Mas observe que é impossível determinar quais são verdadeiros/-falsos negativos se não atribuirmos um *true label* a todos os podcasts para essa consulta. Esse problema foi contornado pelo cálculo de outras métricas que não dependem de uma avaliação manual tão extensiva, mas acreditamos que é importante de ser destacado.

Ainda no tópico de metrificação, obtivemos resultados insatisfatórios para a métrica de Reciprocidade na tarefa de recomendação. Por exemplo, a taxa ficou em cerca de 20% de reciprocidade apenas para o modelo Semântico, e nesse momento ainda não é claro para nós se isso demonstra uma falha do modelo na aplicação a nosso conjunto de dados ou se a métrica, por alguma razão, não é informativa conforme esperávamos.

Em relação a nossos próximos passos, esperamos implementar novas funcionalidades de front, como a possibilidade de comparar dois resultados lado a lado e selecional qual modelo performa melhor, possivelmente recebendo feedback de usuários, passar a exibir na lista de resultados o próprio podcast invés de apenas o título. Além disso, temos a intenção de ampliar a gama de modelos, principalmente para a tarefa de recomendação, e revisar nossas métricas, atribuindo scores e realizando cálculos para a tarefa de recomendação.