Avaliação de Modelos de Recuperação de Informação Utilizando Biblioteca de Busca por Similaridade FAISS

José Carlos Ferreira Neto[1]

[1]Programa de Pós-Graduação em Engenharia de Sistemas e Automação – Universidade Federal de Lavras (UFLA)

Abstract. *Information Retrieval (IR) is an area of computer science that explores techniques of representation, storage, organization and access to information. RI models seek to satisfy user needs by retrieving documents that meet their request. This work presents a comparison of IR modeling techniques, combining vector representation generated through TF-IDF and BERT. These strategies are evaluated with the Cystic Fibrosis (CF) collection.*

Resumo. *A Recuperação de Informação (RI) é uma área da computação que explora técnicas de representação, armazenamento, organização e acesso a informação. Os modelos de RI buscam satisfazer as necessidades do usuário, recuperando documentos que atendem a sua solicitação. Este trabalho apresenta uma comparação de técnicas de modelagem de RI, combinando representação vetorial gerada através do TF-IDF e do BERT. Estas estratégias são avaliadas com a coleção Cystic Fibrosis.*

1. Introdução

bla bla bla.

2. Representação Vetorial

As máquinas de busca podem ser definidas, segundo [Croft et al. 2010] como sendo a utilização de técnicas de recuperação de informação (RI) para coleções de texto.

Para o desenvolvimento de um sistema de máquina de busca é necessário definir o tipo de representação vetorial a ser aplicado aos documentos da coleção. Várias são as formas de se gerar estas representações, desde estratégias relacionado a frequência dos termos em um documento até estratégias baseadas em aprendizado de máquina.

O *Term Frequency – Inverse Document Frequency* (TF-IDF) é uma técnica dentro da modelagem vetorial, que consiste em produzir uma representação vetorial de um texto, e esta estratégia é baseada em pesos. O componente TF baseia-se na frequência de cada palavra em cada documento da coleção, e o componente IDF está relacionado com a quantidade de documentos que uma palavra aparece [Croft et al. 2010].

O documento pode ser representado pelo vetor $\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{i,j})$, sendo que o componente tf do termo k_i no documento d_j pode ser obtido por:

$$tf_{i,j} = \begin{cases} 1 + log f_{i,j} & se\ f_{i,j} > 0 \\ 0 & caso\ contrário \end{cases} \quad (1)$$

Onde $f_{i,j}$ representa a frequência de ocorrência de um termo k_i no documento d_j. Já o IDF pode ser obtido por:

$$idf_i = log\frac{N}{n_i} \tag{2}$$

Onde N é o total de documentos em uma coleção e n_i o número de documentos que o termo k_i aparece. Portanto, a combinação dos componentes TF e IDF produz pesos conforme a equação a seguir:

$$w_{i,j} = \begin{cases} (1 + log f_{i,j}) \times log\frac{N}{n_i} & se\ f_{i,j} > 0 \\ 0 & caso\ contrário \end{cases} \tag{3}$$

Outra forma de modelagem é a modelagem semântica. Uma das técnicas desse tipo de modelagem, sendo esta mais recente, é utilizar o modelo BERT (*Bidirectional Encoder Representations from Transformers*) para produzir vetores para as palavras captando o seu sentido no contexto a qual esta está inserida.

O BERT é um modelo pré-treinado desenvolvido pela Google utilizado para tarefas de Processamento de Linguagem Natural (PLN). Os dados de pré-treinamento utilizados extraídos do *BooksCorpus* (800 milhões de palavras) e do *Wikipedia* (2.500 milhões de palavras) [Devlin et al. 2018].

Entretanto, o BERT é restrito a vetorização de palavras, por conta disso, pensando em escalar o mesmo conceito e gerar vetores (*embeddings*) para toda uma sentença, em [Reimers and Gurevych 2019] é proposto uma modificação deste modelo, dando origem ao *sentece*-BERT. Os autores explicam que, o *sentece*-BERT utiliza um codificador que pode converter passagens mais longas de texto em vetores. Este sistema é bastante útil para busca semântica de documentos, uma vez que sentenças semelhantes semanticamente estão próximas umas das outras no espaço vetorial.

3. Estratégias Adotadas

Para construção dos classificadores, foram utilizados dois algoritmos, kNN e SVM. Para definir a melhor configuração de cada um destes algoritmos, utilizou-se a técnica de *Grid Search* com validação cruzada, conforme a Figura 1.

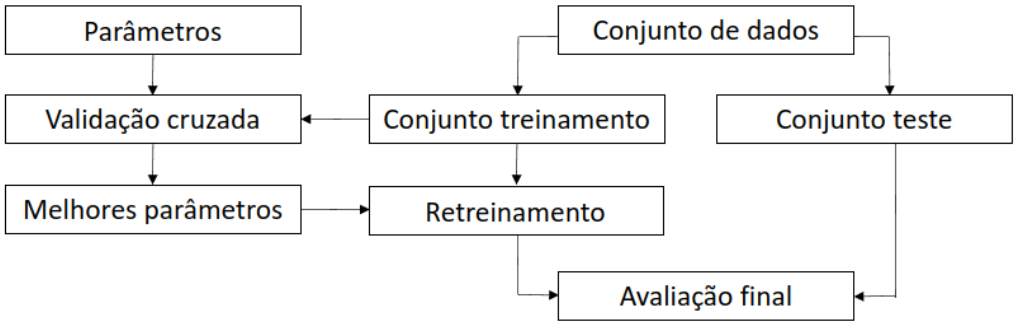

Figura 1. Fluxo Adotado de Treinamento e Avaliação dos Algoritmos

Os tópicos a seguir abordarão conceitualmente os algoritmos e as técnicas utilizadas.

3.1. *Grid Search*

Nesta técnica, faz-se uma busca exaustiva pelo melhor valor de um parâmetro específico do algoritmo. Para isso, defini-se uma série de valores com que cada um destes parâmetros podem assumir.

A busca exaustiva se dá pelo treinando de diversos modelos, combinando os valores definidos para cada um dos parâmetros a serem otimizados, e pela avaliação de cada um destes modelos. A melhor configuração, ou seja, a melhor combinação resultará no modelo com as melhores métricas e por consequência este será o modelo selecionado para o treinamento final, conforme demonstra o fluxo da Figura 1.

3.2. Validação Cruzada

A validação cruzada é uma técnica de amostragem que utiliza diferentes partes dos dados para treinar e testar um modelo em diferentes iterações. Utiliza-se esta técnica principalmente para avaliar o modelo sob diferentes amostragens, estimando assim a sua precisão para prever novos valores sobre dados não vistos durante a etapa de treinamento.

Existem várias abordagens para aplicar validação cruzada, são elas: *leave-p-out*, *leave-one-out*, *holdout*, *k-fold* entre outras. Neste trabalho abordaremos apenas a *k-fold*.

O *k-fold* consiste em dividir os dados disponíveis em k partições (*folds*), instanciar k modelos idênticos e treiná-los em $k-1$ partições enquanto os avalia na partição restante. A validação do modelo é feita utilizando a média da(s) métrica(s) obtida(s) nas k iterações [Chollet 2021].

3.3. kNN

A classificação por método de vizinhança não exige uma etapa de treinamento, como normalmente outros classificadores exigem. Isto é, no kNN esta etapa serve para armazenar as instâncias dos dados de treinamento. Para que um novo registro de dado seja classificado, os seus k vizinhos mais próximos são recuperados, formando a vizinhança deste registro. É atribuído a este registro a classe de dados que tem o maior número de representantes dentre a vizinhança, esta última etapa é denominada de votação majoritária [Guo et al. 2003].

Para produzir a vizinhança, necessita-se definir o tipo de cálculo de distância utilizado. A implementação clássica do algoritmo é a utilização da distância euclidiana, conforme a equação a seguir:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4}$$

Onde, $\mathbf{x}$ e $\mathbf{y}$ são os vetores, cada um representando um registro, x_i e y_i são valores da coordenada i e n o número de coordenadas, isto é, o comprimento do vetor.

Para além da distância euclidiana, pode-se utilizar a similaridade do cosseno, conforme equação a seguir:

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{5}$$

A votação para a definição da classe a ser atribuída a um novo registro pode ser por voto majoritário simples, isto é, todos os k vizinhos mais próximos possuem o mesmo peso de voto, quanto por voto majoritário ponderado pela distância, isso significa que, vizinhos mais próximos possuem maior influência para definir a classe do novo registro do que vizinhos mais distantes.

3.4. SVM

A ideia principal de uma máquina de vetor de suporte é construir um hiperplano como superfície de decisão de tal forma que a margem de separação entre exemplos de classes diferentes seja máxima [Haykin 2001].

Para conjuntos de dados que não são linearmente separáveis, existem algumas complicações, visto que o objetivo é separar as classes sem violação da margem.

Para contornar este problema, podemos adotar classificação de margem suave, isto é, permitir que pontos de dados violem a margem. Com isto, produz-se um modelo flexível, capaz de equilibrar a largura da margem para que seja máxima e permitir que alguns pontos violem a margem [Géron 2022].

Outra abordagem é aumentar o número de dimensões do conjunto de dados, de tal forma que, em uma alta dimensão os dados possam ser separados. Porém, ao aumentar a dimensionalidade do problema, o modelo se torna mais lento computacionalmente falando. Podemos mapear estes dados em alta dimensão sem a necessidade de adicionar mais características utilizando o troque do *kernel*. Este mapeamento é feito por meio de uma função, os *kernels* podem ser: linear, polinomial, RBF e sigmoide [Géron 2022].

4. Experimentos

Todos as bibliotecas mencionadas na seção 3 estão disponíveis para linguagem Python, e portanto, todos os experimentos foram executados utilizando esta linguagem. A biblioteca FAISS, que disponibiliza as opções de indexação e busca está disponível para ser executada via processamento de CPU ou GPU, neste trabalho utilizou-se o processamento via CPU. Assim sendo, a configuração da máquina utilizada é: processador (CPU) Ryzen 5 3600, 6-CORE, 12-THREADS, velocidade de 3.6GHz, GPU GTX 1660TI com 6Gb de memória, e duas memórias (RAM) DDR4 de 16GB e velocidade de 2666MHz.

Todos os códigos utilizados para executar os experimentos mencionados neste trabalho podem ser acessados no GitHub[1].

5. Resultados

6. Conclusão

Referências

Chollet, F. (2021). *Deep learning with Python*. Simon and Schuster.

Croft, W. B., Metzler, D., and Strohman, T. (2010). *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading.

[1]Códigos disponíveis em: https://github.com/jcfneto/ir

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. "O'Reilly Media, Inc.".

Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. (2003). Knn model-based approach in classification. In *OTM Confederated International Conferences"On the Move to Meaningful Internet Systems*
, pages 986–996. Springer.

Haykin, S. (2001). *Redes neurais: princípios e prática*. Bookman Editora.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.