Implementação da versão distribuída do algoritmo DCDistance usando a ferramenta Apache Spark

Patrícia Dias dos Santos
RA 23201810211
patricia.santos@ufabc.edu.br

Universidade Federal do ABC
Programa de Pós-graduação em Ciência da Computação
Disciplina de Inteligência na Web e Big Data

Abstract

No projeto foi feita a implementação da versão distribuída do algoritmo *DC-Distance: A Supervised Text Document Feature extraction based on class labels*, um algoritmo de extração e redução de atributos supervisionados, utilizando a ferramenta Apache Spark, e mais especificamente a biblioteca *pyspark*.

Keywords: `DCDistance`, Apache Spark, programação funcional

1. Breve Explicação do DCDistance

Este projeto trata da implementação da versão distribuída do algoritmo *DC-Distance: A Supervised Text Document Feature extraction based on class labels* [1], utilizando a biblioteca *pyspark* do Python. Este algoritmo faz a extração e redução de atributos supervisionados, e cria recursos baseados na distância entre um documento e um representante de cada etiqueta de classe. A sua principal vantagem é reduzir a dimensionalidade do conjunto de dados, ao mesmo tempo em que mantém a sua discriminalidade para as tarefas de classificação.

Document-Class Distance. Esse algoritmo pode ser descrito da seguinte maneira: basicamente, um vetor do mesmo tamanho do número de rótulos distintos representa cada documento de texto. Assim, o elemento $d_{i,j}$ representa a distância do i-ésimo texto para um vetor representativo da j-ésima classe.

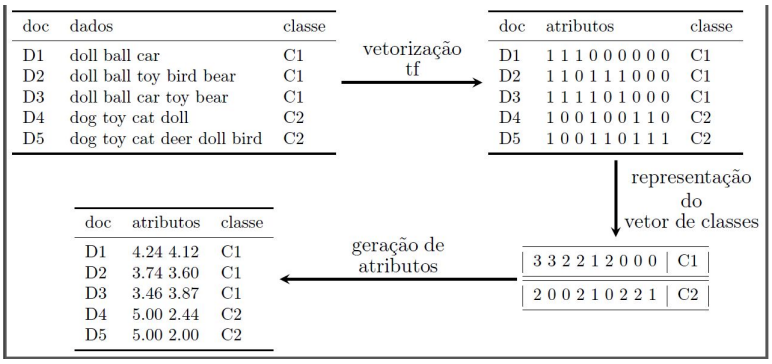

Figura 1: Exemplo de aplicação do DCDistance.

O algoritmo começa com a aplicação de qualquer algoritmo de vetorização nos dados do corpus. Por exemplo, um Bag-of-Words com ponderação TF-IDF. Depois disso, os vetores dos documentos correspondentes a cada etiqueta são somados gerando um vetor representativo desta classe particular. Finalmente, a nova representação vetorial para cada documento é criada calculando a distância entre este documento e cada vetor representativo [1].

Um exemplo de aplicação do DCDistance pode ser visto na Figura 1.

2. Implementação da versão distribuída do algoritmo DCDistance

Base de Dados. Foi utilizada a base *DOHMH New York City Restaurant Inspection Results*[1], com dados em formato CSV, com um tamanho de aproximadamente de 133 megabytes e mais de 370 mil registros. Essa base contém o resultado da inspeção sanitária em restaurantes da cidade de Nova Iorque entre os anos de 2014 e 2018. Para aplicar o DC Distance foram selecionadas as colunas 'VIOLATION DESCRIPTION' e 'CRITICAL FLAG' que continha duas classificações 'critical' e 'not critical'. A ideia era treinar a base para ler a descrição da violação e decidir se ela era crítica ou não.

[1]Disponível em: https://data.cityofnewyork.us/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/43nn-pn8j/data

A implementação da versão distribuída do DCDistance foi feita em 4 etapas: (i) ingestão e extração dos dados da base, (ii) geração de um Bag-of-Words com ponderação TF-IDF, (iii) soma dos vetores dos documentos correspondentes às classes 'critical' e 'not critical' de forma a gerar um vetor representativo de cada uma, e finalmente a etapa (iv) que consistiu em criar uma nova representação vetorial para cada documento calculando a distância euclidiana entre este documento e cada vetor representativo.

Cada uma das etapas será descrita de forma mais detalhada nas subseções seguintes seguintes.

Ingestão e extração dos Dados. Essa etapa consistiu dos seguintes passos: (i) Carregamento do arquivo CSV contendo os dados da base com os pacotes Spark csv e (ii) Remoção das colunas que não foram utilizadas e das linhas que continham dados não rotulados. Para isso foi utilizado o componente Spark SQl que permitiu a manipulação dos dados através de comandos na linguagem SQL, após a transformação dos dados para a estrutra Dataframe.

Geração do Bag-of-Words e cálculo do TF-IDF. Essa etapa consistiu dos seguintes passos: (i) Uso da API Spark Machine Learning Pipeline para as seguintes funções: tokenização dos termos dos documentos através da biblioteca *regexTokenizer*, remoção das stop-words através da biblioteca *stopwordsRemover* e contagem dos "document-term vectors" através da biblioteca *countVectors* (geração do Bag-of-Words). (ii) Codificação da coluna de string dos rótulos para uma coluna de índices (0.0 para 'critical' e 1.0 para 'not critical') através da biblioteca *StringIndexer*. (iii) Cálculo do TF-IDF dos documentos através das bibliotecas *HashingTF, IDF*

Geração dos vetores representativos das classes. Essa etapa consistiu em gerar o vetor representativo das classes que é a soma de valores de todos os vetores para cada classe. Isso foi feito através da transformação do *reduceByKey* com o operador de soma de acordo com a posição de cada vetor e da transformação

mapValues que mapeou os valores nas entradas para os valores na saída, transformando o sparse array em array.

Criação da nova representação vetorial das classes para cada documento. Essa etapa consistiu em criar a nova representação vetorial das classes que é o cálculo da distancia euclidiana de cada entrada para o vetor de classe da etapa anterior. Isso foi feito através da transformação do *mapValues* que mapeou os valores nas entradas para os valores na saída, transformando os vetores representativos anteriores em vetores contendo as distâncias euclidianas entre cada documento e vetor representativo.

O código está disponível em: `https://github.com/patyDSantos/BIGDATA2018/blob/master/Projeto`.

3. Resultados e Discussão

A Tabela 1 apresenta a comparação do tempo de execução entre as versões distribuída e serial. A versão serial do algoritmo foi totalmente executada em pouco mais de 23 minutos enquanto que a versão distribuída utilizando 4 cores foi executada totalmente em pouco mais de 1 minuto. Em relação ao CPU time não houve diferença muito significativa de tempo de execução entre os dois algoritmos.

Tabela 1: Comparação do tempo de execução entre as versões distribuída e serial.

Tempo	**Serial**	**Distribuída**
CPU time	$0,556s$	$0,465s$
Wall Time	$23m$	$1,84m$

Logo, conforme o esperado, verificou-se que a versão distribuída do algoritmo obteve um ganho substancial de tempo de processamento quando comparada com a sua versão serial.

References

[1] C. H. P. Ferreira, D. M. R. de Medeiros, F. O. de França, Dcdistance: A supervised text document feature extraction based on class labels, arXiv preprint arXiv:1801.04554.