Método de clusterização de programas de pós-graduação utilizando *k-means* e a algoritmos genéticos

Felipe Luzzardi
Centro de Desenvolvimento Tecnológico (CDTec)
Universidade Federal de Pelotas (UFPel)
Pelotas, Brasil
fldrosa@inf.ufpel.edu.br

Nícolas de Araujo
Centro de Desenvolvimento Tecnológico (CDTec)
Universidade Federal de Pelotas (UFPel)
Pelotas, Brasil
nodaraujo@inf.ufpel.edu.br

***Abstract*—No Brasil, os programas de pós-graduação são usualmente classificados de acordo com sua área de avaliação. Porém, devido a arbitrariedade de escolha diversos programas são classificados de forma genérica ou imprecisa quando considerado sua produção intelectual. Este trabalho tem por objetivo apresentar um método de clusterização de programas de pós-graduação com base nos 50 conceitos mais relevantes trabalhados, escolhidos através de uma derivação de TF-IDF. A clusterização foi realizada utilizando o algoritmo *k-menas* e algoritmos genéticos, clusterizando mais de 450 programas. Durante a execução do algoritmo, foi necessário reduzir drasticamente o número de exemplos, para conseguir executar o algoritmo em tempo hábil. Ao final da execução, os resultados obtidos foram inconclusivos, não tendo sido possível determinar se o mal desempenho da estratégia teve origem no método em si ou na baixa representatividade dos exemplos utilizados.**

Index Terms*—clusterização, algoritmos genéticos, pós-graduação, *k-means

I. Introdução

No Brasil, a criação, classificação, monitoramento e avaliação dos programas de pós-graduação é feita pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES). Cada programa de pós-graduação (PPG) possui, dentre outras características, uma área de avaliação atrelada ao seu cadastro, sendo esta área de conhecimento indicativa do campo de pesquisa com o qual o PPG se ocupa e pelo qual este será avaliado. Porém, com a criação de diversas novas áreas de pesquisa e abertura de diversos novos programas nestes tópicos, a CAPES tem dificuldade em acompanhar este ritmo, gerando uma certa defasagem pela ausência de áreas ou classificação incorreta de programas. Isto é bastante evidente, principalmente, nas áreas interdisciplinares, onde diversos dos programas presentes encaixam-se na definição de outras áreas, porém permanecem associados à esta área genérica.

A clusterização pode ser definida como um estudo formal de algoritmos e métodos para agrupar ou classificar objetos sem rótulos categóricos [1]. As partições resultantes do processo de clusterização debem possuir duas propriedades: (1) homogeneidade entre os *clusters*, significando que objetos pertencentes a um mesmo *cluster* devem ser o mais similar possível e (2) heterogeneidade entre *cluster*, significando que objetos que pertencem a diferentes *clusters* devem ser o mai diferente possível.

Dentre os diversos algoritmos utilizados para clusterização, o algoritmo de *k-means* ("k-médias", em tradução livre) é um dos mais utilizados. Ele é do tipo de subida de encosta iterativa, obtendo uma solução dependente da clusterização inicial. Embora o algoritmo de *k-means* tenha sido aplicado com sucesso para diversos problemas práticos de clusterização, ele pode acabar convergindo para um partição significativamente inferior a um ótimo global [2].

Para tentar mitigar tal problema, trabalhos como [1] e [3] utilizam-se de algoritmos genéticos em conjunto com o algoritmo de *k-means* ou de outras estratégias de clusterização. Algoritmos genéticos (GAs) foram inicialmente propostos em [4] e podem ser descritos como algoritmos de busca de propósito geral que utilizam-se de princípios inspirados pela genética de populações naturais para evoluir a solução para problemas. Nos GAs, uma população de indivíduos, cada um representado por seu "cromossomo", é mantida, representando as soluções candidatas à solução do problema em questão. Um cromossomo é composto de uma série de genes que representam variáveis de decisão ou parâmetros. Assim, a cada iteração, cada membro da "população" é avaliado e associado à um valor de *fitness*, representando o quão adequada esta solução é para o problema. A cada iteração, chamada no contexto de GAs de "geração", esta pontuação de *fitness* é utilizada para selecionar apenas os mais aptos para comporem a próxima população. Somam-se a estes indivíduos, novos gerados através do cruzamento entre os mais aptos (*crossover*) e da mutação.

Dito isto, este trabalho tem por objetivo clusterizar os programas de pós-graduação brasileiros em diferentes grupos, de acordo com pontuações de relevância para cada um de seus conceitos. Para isso, é utilizado o algoritmo de cluterização *k-means* somado a técnicas de algoritmos genéticos para buscar uma solução mais acurada possível.

O restante do trabalho está dividido como segue: no Capítulo II, é explicado como foi realizada a geração do

dataset utilizado no trabalho, apresentando a extração e pontuação de conceitos. No Capítulo III, é apresentada a execução do algoritmo em si, explicando os parâmetros utilizados e decisões de projeto tomadas. Por sua vez, o Capítulo IV apresenta os resultados obtidos após a execução do algoritmos, realizando também uma breve discussão sobre os mesmos. Por fim, o Capítulo V apresenta as considerações finais do trabalho, e as propostas de trabalhos futuros.

II. Geração do Dataset

Para desenvolvimento do trabalho, o primeiro passo foi a formação de um *dataset*, composto pelos programas de pós-graduação e a pontuação da relevância de sua produção intelectual para o conceito. Para isto, utilizamos a base de dados fornecida pela *startup* de ciência de dados Indeorum, onde estavam disponíveis todos os programas de pós-graduação e suas respectivas produções intelectuais – livros, artigos, patentes, apresentação e etc..

Para auxiliar na extração dos termos relevantes utilizando TF-IDF, foi utilizado o motor de busca ElasticSearch[1], que implementa nativamente o *score* TF-IDF na geração de seus *Term Vectors*. Assim, cada programa foi inserido como um documento com um *corpus* textual representado pelos títulos e resumos de suas produções associadas. Os *Term Vectors* do ElasticSearch nada mais são do que vetores que indicam estatísticas de termos de um documento indexado na base. Para este trabalho, todas as produções de 450 PPGs foram indexadas, possibilitando assim que a ferramenta gerasse os vetores dos 50 termos de maior pontuação para cada um dos programas.

Para formação dos *Term Vectors* foram considerados unigramas, bigramas e trigramas, extraídas as *stopwords* e utilizado o algoritmo Okapi BM-25, derivação do TF-IDF, para dar a pontuação de relevância. Este algoritmo de pontuação destaca-se por considerar também o tamanho do documento para pontuar e também por permitir saturar a pontuação [5]. Atualmente, o TF-IDF é uma das estratégias de ponderação de termo mais utilizadas, com pelo menos 83% dos recomendadores e indexadores baseados em texto utilizando-a [6]. Esta consulta, resultou em um arquivo no formato *JavaScript Object Notation* (JSON), com os dados do programa e seus 50 termos de maior pontuação. Parte do JSON resultante pode ser visualizado na Figura 1.

Por fim, devido ao fato de o algoritmo de *k-means* trabalhar de maneira muito mais eficiente com valores numéricos do que com valores textuais, o *dataset* foi convertido para um formato de matriz. Para isso, foram coletados todos os termos únicos dentro os vetores de cada programa, resultando em em torno de 11 mil termos, e então convertido para matriz, onde cada linha representou um programa e cada coluna representava um termo, com o valor armazenado sendo a pontuação do Okapi BM-25 para este programa com este termo. Com isso, obteve-se um *dataset* puramente numérico que foi posteriormente normalizado para que todos os valores

[1]https://www.elastic.co/

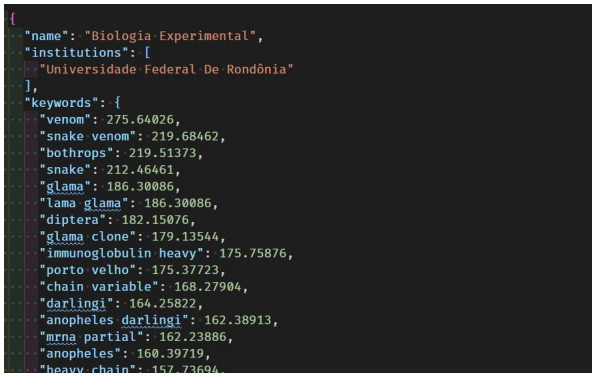

Fig. 1. Parte do arquivo JSON gerado

pertencessem ao intervalo de 0 a 1. Com o *dataset* gerado, passou-se para a próxima etapa do processo, a execução do algoritmo.

III. Execução do Algoritmo

Primeiramente, para execução de um algoritmo genético é importante definirmos os genes que compõem o cromossomo. Como no *k-means* trata-se da clusterização de pontos em K grupos, com cada ponto pertencendo ao *cluster* cujo centróide está menos distante de si, o cromossomo natural para este trabalho é composto pelos centróides de cada cluster. Assim, o cromossomo possui um comprimento de K pontos, onde cada ponto possui em torno de 11 mil dimensões – onde cada uma é um dos conceitos únicos do *dataset*.

Devido a razões de performance, foi necessário uma redução no número de programas analisados. Originalmente, o *dataset* continha os *scores* de 472 programas de pós-graduação o que, idealmente, com um número de clusters K representativo, seria capaz de representar todos os programas de pós-graduação. Para testes preliminares com este dataset foram utilizados um K de 10, com populações compostas por 20 indivíduos e uma iteração por 200 gerações. Porém, com estes parâmetros e este tamanho de dataset, a execução estava sendo muito custosa, com cada geração demorando mais de 2 horas para executar mesmo com paralelização das etapas. Assim, devido a ausência de infraestrutura necessária e o fator tempo para a espera, optou-se por uma redução no número de exemplos, reduzindo dos 472 programas originais para amostra de 30 programas.

Para a execução do algoritmo genético de clusterização, foi necessário definir seus parâmetros, bem como suas funções de *fitness*, de seleção, de *crossover* e de mutação. Para o número de grupos foi utilizado um valor K de 3, com uma população composta por 5 indivíduos e uma iteração por 200 gerações.

Para a função de *fitness* foi utilizado o índice de Davies-Bouldin. Este índice avalia o quão boa é uma clusterização a partir dos próprios atributos dos *clusters*, sem depender de fatores e labels externos. Assim, para cada par de *clusters* i e j da solução, é somado a distância média do centroide para os pontos de cada cluster e então dividida pela distância entre os centróides dos clusters i e j [7].

Para a função de seleção, optou-se por utilizar uma seleção de ranqueamento simples. Nesta estratégia, a lista de indivíduos de uma geração é ordenada de acordo com o *fitness* de cada indivíduo, com os melhores no topo da lista e os piores ao final. Após a ordenação, uma porcentagem de indivíduos do final da lista, ou seja, com *fitness* baixo, é substituída pelo mesmo número de indivíduos do topo da lista, ou seja, com *fitness* alto, duplicando estes indivíduos. A porcentagem de indivíduos a serem substituídos é definida pelo parâmetro de probabilidade de seleção, ou *Ps*, que é definido em um arquivo de configuração, e preenchido antes do início do algoritmo. Após testes, a Ps escolhida para uso neste trabalho foi 20%.

Já a respeito da estratégia de *crossover*, foi decidido utilizar uma estratégia de *single-point crossover*. Nesta estratégia, pontos dos cromossomos de ambos os pais são escolhidos de maneira aleatória e designados como "pontos de *crossover*". Após a designação, bits a direita dos pontos são trocados entre os dois cromossomos pais, resultando em dois descendentes, cada um carregando uma porção da informação genética de ambos os pais. Após a ocorrência do *crossover*, os dois indivíduos pais e seus dois descendentes são ordenados por seu *fitness*, e os dois indivíduos com maior *fitness* são selecionados para fazer parte da próxima geração. Com isso, se ambos os descendentes forem superiores aos seus pais, os dois farão parte da próxima geração, se apenas um descendente for superior a seus pais, este descendente irá para a próxima geração junto de um pai e, se nenhum descendente for superior, os indivíduos pais continuarão na próxima geração. Assim como em todo algoritmo genético, o *crossover* não ocorre para todos os indivíduos, sendo que, no presente algoritmo, a ocorrência ou não de *crossover* é definida por um parâmetro de probabilidade denominado probabilidade de *crossover*, ou *Pc*. para este trabalho, utilizou-se o valor de *Pc* como 0.8.

Por sua vez, a estratégia de mutação utilizada foi a mutação uniforme. Nesta estratégia, quando ocorre mutação, o valor do gene escolhido para sofrer mutação é substituído por um valor aleatório, gerado entre limites superiores e inferiores, que são especificados pelo usuário do algoritmo. Este tipo de mutação só pode ser utilizado para genes que codifiquem valores inteiros ou decimais e, como toda mutação, deve possuir uma baixa probabilidade de ocorrência. No presente trabalho, a probabilidade de mutação, ou Pm, também foi definida em um arquivo de configuração externo, lido no início da execução do algoritmo. Já quanto ao limite inferior e superior do valor aleatório gerado, optou-se por utilizar o valor 0.0 como inferior e 1.0 como superior, já que os dados de entrada do *dataset* são normalizados entre estes dois valores. Para as execuções cujos resultados são apresentados neste trabalho, o valor do Pm utilizado foi 2%

IV. Resultados Obtidos

A execução com um *dataset* com 30 exemplos, um valor *K* de 3 e 5 indivíduos por geração resultou em um fitness de apenas 0.14710447568578033 ao final da execução. O valor do *fitness* não se alterou conforme a passagem das gerações, um comportamento considerado inesperado. Assume-se que um dos possíveis motivos para este desempenho tão fraco seja a baixa quantidade de exemplos, com 30 não sendo um número muito significativo de exemplos. Também é possível que este comportamento tenha sido causado pelo baixo número de indivíduos nas gerações, ou por alguma inadequação dos pesos utilizados (Pc, Ps e Pm). Porém, até o fechamento deste artigo, não foi encontrado um conjunto de parâmetros que melhorasse de forma significativa os resultados obtidos.

Devido a problemas de performance e falta de infraestrutura, não foi possível realizar a execução completa do algoritmo com o *dataset* completo e com uma parametrização mais adequada. A impossibilidade de realizar os testes com o *dataset* completo prejudica enormemente a discussão dos resultados obtidos, pois impossibilita que se conclua se o método de clusterização utilizado é inapropriado para o problema apresentado, ou se ele é um método válido, mas apenas na presença de um número maior de exemplos. É importante ressaltar, porém, que um tempo de execução muito longo para a obtenção de um resultado para um problema de aproximação com muitos dados pode indicar que o problema não deveria ser resolvido desta forma, já que, a medida que o tempo passa, o número de dados do *dataset* só tende a aumentar, aumentando ainda mais o tempo de execução do algoritmo.

Por fim, conclue-se que os resultados obtidos não foram positivos mas, ao mesmo tempo, foram inconclusivos, não tendo sido possível chegar a uma resposta precisa sobre a validade da utilização de algoritmos genéticos para clusterização de conceitos de programas de pós-graduação.

V. Considerações Finais

Este trabalho teve como objetivo apresentar um método de clusterização de programas de pós-graduação utilizando algoritmos genéticos e o algoritmo *k-means*. A geração do *dataset* utilizado foi apresentada, explanado as diversas etapas percorridas até a obtenção do conjunto de dados final, em formato puramente numérico e normalizado.

Para que o algoritmo de clusterização encontra-se relação entre as pontuações de programas, foi necessária a extração de um alto número de termo para cada programa, de modo a garantir que houvesse uma boa representatividade de sua produção. Porém, com um alto número de termos únicos, temos um alto número de dimensões para os dados e, consequentemente, um alto número de genes em cada cromossomo. Este fator apresentou-se como um grande problema principalmente em termos de escala e performance do algoritmo, necessitando uma forte paralelização do mesmo e, mesmo assim, tornando-se um fator de dificuldade para a obtenção de resultados. Esse problema de performance resultou na necessidade de redução do *dataset* utilizado e limitação em alguns parâmetros, como o número de grupos, para viabilizar a execução.

Os resultados apresentados foram bastante insatisfatórios. Estima-se que foram fatores influenciadores a necessidade de redução do *dataset* para execução, de 472 para 30 amostras. Este fator reduziu drasticamente a amostra utilizada. Soma-se a este fator também a limitação na representação do escopo de

atuação de um programa, onde apenas 50 termos talvez não sejam suficientes para isto, porém também foram limitados pelo fator de performance.

Para trabalhos futuros apresenta-se claramente como principal desafio a viabilização da execução do algoritmo em questões de escalabilidade e performance. Também apresenta-se como desafio aumentar a representatividade de termos, de forma a obter vetores e representações de cada programa mais completas.

REFERENCES

[1] Yongguo Liu, Xindong Wu, and Yidong Shen. Automatic clustering using genetic algorithms. *Applied mathematics and computation*, 218(4):1267–1279, 2011.

[2] SZ Selim, MA Ismail, and K-means Type Algorithms. A generalized convergence theorem and characterization of local optimality. *IEEE Trans. Pattern Anal. Mach. Intell*, 6:81–87, 1984.

[3] Yu-Chiun Chiou and Lawrence W Lan. Genetic clustering algorithms. *European journal of operational research*, 135(2):413–427, 2001.

[4] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[5] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[6] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016.

[7] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.