UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ESPECIALIZAÇÃO EM FRONTEIRAS DE TI

JOSÉ EVALDO GONÇALVES LOPES FILHO

O uso de Machine Learning (t-SNE) como técnica para visualização de padrões no Programa Bolsa Família com base nos dados do Cadastro Único.

MONOGRAFIA

Brasília
2019

JOSÉ EVALDO GONÇALVES LOPES FILHO

O uso de Machine Learning (t-SNE) como técnica para visualização de padrões no Programa Bolsa Família com base nos dados do Cadastro Único.

Monografia apresentada ao Curso de Especialização em Fronteiras de TI como parte dos requisitos necessários à obtenção do título de Especialista em Tecnologia da Informação.

Orientador: Adriano Alonso Veloso

Brasília
2019

UNIVERSIDADE FEDERAL DE MINAS GERAIS

INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ESPECIALIZAÇÃO EM INFORMÁTICA: ÁREA DE CONCENTRAÇÃO
GESTÃO EM TECNOLOGIA DA INFORMAÇÃO

O uso do t-SNE como técnica para identificação de padrões em benefícios do Bolsa Família a partir do Cadastro Único

JOSÉ EVALDO GONÇALVES LOPES FILHO

Monografia apresentada aos Senhores:

Prof. Adriano Alonso Veloso
Orientador
DCC - ICEx – UFMG

Prof. José Nagib Cotrim Arabe
DCC - ICEx - UFMG

Prof. José Marcos Silva Nogueira
DCC - ICEx - UFMG

Belo Horizonte, 14 de março de 2019

Lopes Filho, José Evaldo

L864U O uso de Machine Learning (t-SNE) como técnica para visualização de

padrões no Programa Bolsa Família com base nos dados do Cadastro Único /José Evaldo Lopes Filho. – 2019.
33 f.

Monografia (especialização em informática) – Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Prof. Dr. Adriano Alonso Veloso.

1. Informática . 2. Machine Learning. 3 Bolsa família. 4.Cadastro Único. 5. Visualização. 6. Redução de dimensionalidade. I. Orientador. II. Título.

CDU 519.6*

José Evaldo Gonçalves Lopes Filho

O uso de Machine Learning (t-SNE) como técnica para visualização de padrões no Programa Bolsa Família com base nos dados do Cadastro Único.

Adriano Alonso Veloso
Orientador

José Nagib Cotrim Árabe

José Marcos Silva Nogueira

Brasília
2019

Dedico esse trabalho à minha amada família.

Agradecimentos

Agradeço a Deus, que tem sido minha torre forte. Ele tem me guiado em águas tranquilas e me sustentado em meio às dificuldades da vida. Graças a Ele, consegui realizar o sonho de executar esse trabalho.

À minha família, à qual amo profundamente, que me apoiou quando precisei me dedicar grande parte do tempo a esse projeto. Especialmente à minha esposa, que como auxiliadora, tem me apoiado e acompanhado cada passo da minha vida. Aos meus filhos, que entenderam muitas vezes que precisei me ausentar para realizar esse sonho.

À minha mãe, que nos deixou durante a elaboração dessa monografia. Seu cuidado foi fundamental na minha vida. Não tenho como compensar toda a dedicação e amor que recebi. A sua lembrança deixará sempre um tom de perseverança e alegria.

Ao meu pai, que da mesma forma que a minha mãe, tem me amparado sempre que preciso. A minha gratidão dedico também a ele, que tem passado por momentos difíceis após a ausência da sua companheira, mas que sempre renunciou a sua vida para o bem de todos nós.

A todos os meus professores, que fizeram parte dessa especialização. Ao meu orientador, o professor Adriano Alonso Veloso, que tem me ajudado a escrever e enriquecer esse trabalho.

Aos meus amigos, que durante esse tempo todo estiveram comigo nos momentos de descontração e nos momentos de dor.

“Aqueles que semeiam com lágrimas, com cantos de alegria colherão. Aquele que sai chorando enquanto lança a semente, voltará com cantos de alegria, trazendo os seus feixes.“

(Salmos 126:5,6)

Resumo

Este projeto se encaixa num contexto econômico bem peculiar para o país, onde existe grande preocupação na eficiência dos gastos públicos, especialmente no emprego útil do dinheiro público para a assistência social aos que realmente dela necessitam. Nesse contexto, este projeto objetiva avaliar o uso de *machine learning* no monitoramento e avaliação de benefícios do Programa Bolsa Família, de forma a identificar anormalidades ou desvios em relação a um padrão comum. Para atingir os objetivos pretendidos, foram realizadas pesquisas em relatórios governamentais sobre esta política pública, bem como em artigos científicos sobre o algoritmo t-SNE, que permite a visualização dos dados através da redução de dimensionalidade. Com este método, foi possível visualizar benefícios sociais em gráficos 2D e 3D, o que permite aos gestores e aos órgãos de controle um mecanismo inteligente para gerar amostragens utilizadas em suas avaliações e fiscalizações, bem como focar identificação de indícios de fraudes em determinados benefícios considerados *outliers*. Os resultados mostraram que foi possível visualizar determinados grupos onde é possível atuar com foco específico no âmbito dos controles da política assistencial analisada. No entanto, diante do escopo limitado deste trabalho, podem ser realizados testes mais amplos, com incremento do poder computacional empregado e com uma amplitude maior de benefícios, incluindo novas dimensões como dados de entrada.

Palavras-chave: machine learning. t-SNE. bolsa família. cadastro único. visualização. redução de dimensionalidade.

Abstract

This work fits into a very peculiar economic context for the country, where there is great concern about the efficiency of public spending, especially on the useful use of public money for social assistance to those who really need it. In this context, this project aims to evaluate the use of machine learning in the monitoring and evaluation of benefits of the Bolsa Família Program, in order to identify abnormalities or deviations from a common standard. In order to achieve the intended objectives, research was done in government reports on this public policy, as well as in scientific articles on the t-SNE algorithm, which allows the visualization of the data through the reduction of dimensionality. With this method, it was possible to visualize social benefits in 2D and 3D graphs, which allows managers and control bodies an intelligent mechanism to generate samples used in their evaluations and inspections, as well as to focus on identifying evidence of fraud in certain benefits considered outliers. The results showed that it was possible to visualize certain groups where it is possible to act with a specific focus within the scope of the controls of the care policy analyzed. However, given the limited scope of this work, larger tests can be performed, with an increase in computational power employed and with a greater range of benefits, including new dimensions as input data.

Keywords: machine learning. t-SNE. bolsa família. data visualisation. dimensionality reduction.

Lista de ilustrações

Lista de abreviaturas e siglas

2D	Duas dimensões
3D	Três Dimensões
CadUnico	Cadastro Único
NIS	Número de Informações Sociais
OBS	Observação
PBF	Bolsa Família
SQL	Server Query Language

Sumário

Introdução

A Assistência Social compõe um conjunto de ações que integram a Seguridade Social. Seus objetivos principais são a proteção à família, à maternidade, à infância, à adolescência, o amparo a crianças e adolescentes carentes e à pessoa com deficiência. Nesse contexto, o Programa Bolsa Família – (PBF) - tem como objetivo combater a fome, a pobreza e outras formas de privação das famílias. Para que a família possa ingressar no PBF, é necessária a sua inscrição no Cadastro Único, ou CadUnico, que é o cadastro para Programas Sociais do Governo Federal. Esse Cadastro é um instrumento que identifica e caracteriza as famílias de baixa renda, permitindo que o governo conheça melhor a realidade socioeconômica dessa população. Nele são registradas informações como: características da residência, identificação de cada pessoa, escolaridade, situação de trabalho e renda, entre outras. Atualmente, há cerca de 13,6 milhões de famílias recebendo os benefícios do Programa Bolsa Família, com o valor médio mensal aproximado de R$ 185,00.

Nesse contexto, esse trabalho objetiva buscar a melhoria na tarefa governamental de avaliar e monitorar seus beneficiários. Essa tarefa é feita periodicamente pelos próprios gestores e pelos órgãos de controle do governo, o que envolve cruzamentos de dados entre bases de dados governamentais como renda e benefícios. Isso torna possível avaliar se determinada família ou beneficiário(a) permanecerá no Programa. Por ser um benefício de caráter familiar, também pode ser necessário que sejam revisadas as composições familiares do cadastro, de forma a identificar se uma composição familiar inscrita no CadUnico foi informada de boa-fé pelo membro responsável. De forma geral, o processo de avaliação termina com a apuração dos casos mais críticos, seja pela incoerência detectada na composição familiar, seja pela renda apurada incompatível com os critérios exigidos pela política pública.

A tarefa de avaliar benefícios pode não gerar os resultados esperados, pois depende geralmente de cruzamentos entre bases de dados governamentais, que são métodos conhecidamente determinísticos. Além disso, é importante lembrar que a base de beneficiários do PBF, o CadUnico, é preenchida de maneira

autodeclarada. Isso quer dizer que os dados são colhidos com base em entrevista feita ao responsável familiar, e nem sempre refletem com precisão a realidade daquela família. No entanto, os dados documentais geralmente são confiáveis, e as informações adicionais podem ser obtidas através de cruzamentos entre os demais registros cadastrais governamentais, *facilitando* a complementação de informações até então incompletas.

Diante desse desafio, a visualização dos dados em agrupamentos ou *clusters* pode ser um meio de auxiliar os órgãos governamentais que cuidam da avaliação do PBF em suas atividades de monitoramento e avaliação. Este trabalho tem o propósito de avaliar a viabilidade de utilizar o algoritmo *t-SNE (T-distributed Stochastic Neighbor Embedding*) (MAATEN; HINTON, 2008b), como técnica baseada em *machine learning* para visualizar padrões, isto é, pontos “fora da curva” na base de benefícios do PBF. (MAATEN; HINTON, 2008a) realizou testes em bases como a MNIST (base com vários dígitos manuscritos de 0 a 9), e a base de dados de filmes da Netflix, colhidos a partir de avaliações de cerca de 17 mil avaliações, de um total de cerca de 400 mil espectadores. Quanto à experiência da Netflix, produziu uma visualização dos 500 filmes mais populares usando a técnica, mostrando que o algoritmo conseguiu agrupar satisfatoriamente filmes com características semelhantes. O mesmo resultado é pretendido por esta experiência com o CadUnico, de forma a agrupar visualmente famílias ou beneficiários semelhantes, agregando entre eixos bi ou tridimensionais os indivíduos semelhantes e expondo os grupos considerados destoantes.

Para atingir esse objetivo, esse trabalho coletou amostra de dados do Cadastro Único, com posição referente a agosto de 2018, e da folha de pagamento do PBF, com competência para o mês de janeiro de 2019. Os dados dos beneficiários foram obtidos através de um cruzamento simples usando como chave o Número de Identificação Social (NIS) do beneficiário. Por questões de privacidade, foram descartados os dados pessoais alfanuméricos, como nome, endereço e demais documentos pessoais. Dessa forma, ficou preservado o sigilo das informações, de forma a garantir que não fossem exibidas quaisquer informações que permitissem identificar um beneficiário ou família.

A proposta é executar o algoritmo para a amostra de 500 benefícios, selecionados aleatoriamente, e visualizar o gráfico resultante dessa etapa. Serão geradas 6 visualizações:

1) Ao final de 1000 iterações com *perplexidade igual a 5*

2) Ao final de 1000 iterações com *perplexidade igual a 25*

3) Ao final de 1000 iterações com *perplexidade igual a 50*

4) 4) Ao final de 3000 iterações com *perplexidade igual a 5*

5) Ao final de 3000 iterações com *perplexidade igual a 25*

6) Ao final de 3000 iterações com *perplexidade igual a 50*

7) Visualização utilizando o site dinâmico (http://projector.tensorflow.org/)

Na sequência, a revisão bibliográfica está descrita no Capítulo 2. O *t-SNE,* suas especificidades e os passos necessários para executar esse experimento foram descritos no Capítulo 3. O Capítulo 4 trata da coleta dos dados e dos resultados obtidos. As considerações finais são feitas no Capítulo 5.

Revisão da Literatura

Maaten; Hinton ,(2008b) apresentaram o t-SNE, técnica de visualização de dados multidimensionais. O trabalho consistiu em criar uma matriz de similaridades de pontos do universo, permitindo a visualização de agrupamentos nos dados originais. A técnica se baseia em converter distâncias Euclidianas do espaço multidimensional em probabilidades condicionais que representam as similaridades. Tal estrutura de saída permite, num plano 2D ou 3D, visualizar a existência ou não, dependendo dos dados de entrada, a existência de *clusters*, o que possibilita a análise e o tratamento dos dados nos diversos aspectos da área de negócio avaliada. A técnica implementa o parâmetro “perplexidade”, que permite balancear os aspectos locais e globais dos dados. Em linhas gerais, é um palpite acerca de quantos vizinhos no *cluster* cada ponto tem. Os experimentos de Maaten e Hinton foram executados em várias bases de dados de imagens, filmes e palavras, onde foi possível testar a capacidade do algoritmo de “juntar” os pontos similares em grupos bem definidos no mapa visualizado. O efeito *crowd*, indesejado para os pesquisadores de modelos de redução de dimensionalidade, foi tratado pelo t-SNE a partir da regulagem dos seus hiperparâmetros e do ajuste do gradiente, explicado no próximo capítulo.

Wattenberg; Viégas; Johnson (2016), explicam que as visualizações geradas pelo t-SNE podem ser enganosas, e que o método deve ser utilizado de forma racional. O trabalho de Wattenberg explora o t-SNE e alerta a interpretar os resultados evitando ler os resultados de forma equivocada. Em suma, o teste reforça os cuidados que se deve ter ao buscar o t-SNE como forma de tomar conclusões acerca dos grupos e formatos exibidos nos gráficos gerados. Alguns pontos do artigo de Wattenberg sobre o t-SNE são importantes destacar, a saber:

1) Os parâmetros são determinantes para o resultado.

2) Os tamanhos dos *clusters* não são significativos.

3) As distâncias entre os *clusters* podem (ou não!) significar alguma coisa.

4) O ruído aleatório nem sempre parece aleatório.

5) Podem existir várias formas diferentes com os mesmos dados de entrada.

6) Pode ser necessário visualizar mais de um gráfico para interpretar os agrupamentos.

Materiais e Métodos

Uma síntese do t-SNE

O algoritmo t-SNE fornece um método eficaz para visualizar um conjunto complexo de dados. Ele descobre com sucesso estruturas ocultas nos dados, expondo *clusters* naturais e suavizando variações não-lineares ao longo das dimensões. Mas como podemos reduzir a alta dimensionalidade de um conjunto de dados para duas ou três dimensões, que é o que estamos fazendo quando visualizamos dados em uma tela? O fato é que muitos conjuntos de dados do mundo real têm uma baixa dimensionalidade intrínseca, apesar de estarem inseridos em um espaço de alta dimensão. Esse espaço de baixa dimensão está embutido no espaço de alta dimensão de uma maneira complexa e não linear. Escondido nos dados, esta estrutura só pode ser recuperada através de métodos matemáticos específicos (ROSSANT, 2015).

O método, em síntese, converte as distâncias entre os pontos no espaço multidimensional em probabilidades que representam as similaridades. No espaço dimensional reduzido, as distâncias são também calculadas, sendo posteriormente ajustadas conforme o cálculo do gradiente, que representa a similaridade posicional dos pontos em relação a ambos os espaços dimensionais. A síntese do algoritmo segundo MAATEN; HINTON (2008 b), pode ser vista como segue:

Figura 1 – t-SNE - Algoritmo simplificado

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $X = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta y}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta y} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Fonte: Maaten e Hinton - Visualizing data using t-SNE (2008)

O algoritmo t-SNE adaptado para este trabalho

Em python, o algoritmo de Maaten e Hinton, com pequenas adaptações para este trabalho, assumiu a seguinte forma:

Código 3.1 – Código-fonte do algoritmo t-SNE com adaptações para esse trabalho

```
#
# tsne.py
#
# Implementation of t-SNE in Python. The implementation was tested on Python
#   2.7.10, and it requires a working The              installation          of
      implementation                                                        NumPy.
#   comes with an example on the MNIST                 dataset. In           order to
      plot the
#   results of this example, a working                 installation          of
      matplotlib is required.
#
#   The example can be run by
    executing: 'ipython tsne.py'
#
#
```

```
# Created by Laurens van der Maaten on 20-12-08.
# Copyright (c) 2008 Tilburg University. All rights reserved
    .
import numpy as np import
pylab
from sklearn.preprocessing import normalize

# This import registers the 3D projection, but is otherwise unused.
from mpl_toolkits.mplot3d import Axes3D # noqa: F401 unused import import
matplotlib.pyplot as plt

# Fixing random state for reproducibility np.random.seed(19680801)

def randrange(n, vmin, vmax):
    '''
    Helper function to make an array of random numbers having shape (n, )
    with each number distributed Uniform(vmin, vmax).
    '''
    return (vmax - vmin)*np.random.rand(n) + vmin

def Hbeta(D=np.array([]), beta=1.0):
    """
        Compute the perplexity and the P-row for a specific value of the
        precision of a Gaussian distribution. """

    # Compute P-row and corresponding perplexity P = np.exp(-
    D.copy() * beta) sumP = sum(P)
    H = np.log(sumP) + beta * np.sum(D * P) / sumP P = P / sumP
    return H, P

def x2p(X=np.array([]), tol=1e-5, perplexity=30.0):
    """
        Performs a binary search to get P-values in such a way that each
        conditional Gaussian has the same perplexity. """

    # Initialize some variables
    print("Computing pairwise distances...")
    (n, d) = X.shape
    sum_X = np.sum(np.square(X), 1)
    D = np.add(np.add(-2 * np.dot(X, X.T), sum_X).T, sum_X) P = np.zeros((n,
    n)) beta = np.ones((n, 1)) logU = np.log(perplexity)

    # Loop over all datapoints
    for i in range(n):

        # Print progress if i %
        500 == 0:
```

```
                print("Computing P-values for point %d of %d..."
                    % (i, n))

            # Compute the Gaussian kernel and entropy for the current precision
            betamin = -np.inf betamax =
            np.inf
            Di = D[i, np.concatenate((np.r_[0:i], np.r_[i+1:n]))] (H, thisP) = Hbeta(Di,
            beta[i])

            # Evaluate whether the perplexity is within tolerance Hdiff = H - logU
            tries = 0 while np.abs(Hdiff) > tol and tries < 50:

                # If not, increase or decrease precision if Hdiff > 0:
                    betamin = beta[i].copy() if betamax == np.inf or betamax
                == -np.inf: beta[i] = beta[i] * 2. else: beta[i] = (beta[i] + betamax) /
                2. else:
                    betamax = beta[i].copy() if betamin == np.inf or betamin ==
                    -np.inf: beta[i] = beta[i] / 2. else: beta[i] = (beta[i] +
                    betamin) / 2.

                # Recompute the values
                (H, thisP) = Hbeta(Di, beta[i]) Hdiff = H -
                logU tries += 1

            # Set the final row of P
            P[i, np.concatenate((np.r_[0:i], np.r_[i+1:n]))] = thisP

        # Return final P-matrix
        print("Mean value of sigma: %f" % np.mean(np.sqrt(1 / beta)))
        return P

def pca(X=np.array([]), no_dims=50):
        """
            Runs PCA on the NxD array X in order to reduce its dimensionality to
            no_dims dimensions. """
        print("Preprocessing the data using PCA...")
        (n, d) = X.shape
        X = X - np.tile(np.mean(X, 0), (n, 1)) (l, M) =
        np.linalg.eig(np.dot(X.T, X)) Y = np.dot(X, M[:,
        0:no_dims]) return Y

#alteração sugerida por Evaldo
#def tsne(X=np.array([]), no_dims=2, initial_dims=50, perplexity=30.0):
def tsne(X=np.array([]), no_dims=2, initial_dims=50, perplexity=30.0,
    max_iter=1000):
        """
            Runs t-SNE on the dataset in the NxD array X to reduce its
```

```
        dimensionality to no_dims dimensions. The syntaxis of the function is
        'Y = tsne.tsne(X, no_dims, perplexity), where X is an NxD NumPy array.
    """

    # Check inputs if isinstance(no_dims,
    float):
        print("Error: array X should have type float.") return -1
    if round(no_dims) != no_dims:
        print("Error: number of dimensions should be an integer.")
        return -1

    # Initialize variables
X  = pca(X, initial_dims).real
    (n, d) = X.shape
    #comentado por Evaldo
    #max_iter = 5000
    #max_iter = 1000
    initial_momentum = 0.5
    final_momentum = 0.8 eta =
    500 min_gain = 0.01
Y  = np.random.randn(n, no_dims)dY = np.zeros((n, no_dims)) iY =
   np.zeros((n, no_dims)) gains = np.ones((n, no_dims))

    # Compute P-values
    P = x2p(X, 1e-5, perplexity)
    P = P + np.transpose(P)
P  = P / np.sum(P) P = P * 4.
                                    # early exaggeration P =
    np.maximum(P, 1e-12)

    # Run iterations
    for iter in range(max_iter):

        # Compute pairwise affinities sum_Y =
        np.sum(np.square(Y), 1) num = -2. *
        np.dot(Y, Y.T)
        num = 1. / (1. + np.add(np.add(num, sum_Y).T, sum_Y)) num[range(n),
        range(n)] = 0. Q = num / np.sum(num)
Q  = np.maximum(Q, 1e-12)

        # Compute gradient
        PQ = P - Q
        for i in range(n):
   dY[i, :] = np.sum(np.tile(PQ[:, i] * num[:, i], (
                    no_dims, 1)).T * (Y[i, :] - Y), 0)

        # Perform the update if iter <
        20:
```

```
            momentum = initial_momentum
        else:
            momentum = final_momentum
        gains = (gains + 0.2) * ((dY > 0.) != (iY > 0.)) + \
                (gains *
                0.8) * ((dY
                > 0.) == (iY
                > 0.))
        gains[gains < min_gain] = min_gain
        iY = momentum * iY - eta * (gains * dY)
        Y = Y + iY
        Y = Y - np.tile(np.mean(Y, 0), (n, 1))

        # Compute current value of cost function if (iter + 1) %
        10 == 0:
            C = np.sum(P * np.log(P / Q))
                                                        output   pa
            #trecho inserido por Evaldo, só exibe iteracoes      ra
            superiores if (iter > max_iter/1.1):
                print("Itera
                    tion
                    %d:
                    error is
                    %f" %
                    (iter +
                    1, C))
        # Stop lying about P-values if iter ==
        100:
            P = P / 4.

    # Return solution return Y if
__name__ == "__main__":

    print("Running example on CadUnico...") #X =
    np.loadtxt("mnist2500_X.txt")

    X = np.loadtxt("CadUnico_tSNE_numerico_500.txt", delimiter=",")
    labels = np.loadtxt("CadUnico_tSNE_numerico_labels_500.
        txt")
    X = normalize(X, axis=0, norm='max')
    #print(X)
    #np.savetxt('CadUnico_tSNE_numerico_1000_normal.txt', (X) )

    print("Iteração 1000, Perplexidade = 5") Y = tsne(X, 2, 57,
    5.0, 1000) pylab.scatter(Y[:, 0], Y[:, 1], 20, labels)
    pylab.scatter(Y[:, 0], Y[:, 1], 20) pylab.show()
```

```
print("Iteração 1000, Perplexidade = 25") Y = tsne(X, 2, 57,
20.0, 1000) pylab.scatter(Y[:, 0], Y[:, 1], 20, labels)
pylab.scatter(Y[:, 0], Y[:, 1], 20) pylab.show()

print("Iteração 1000, Perplexidade = 50") Y = tsne(X, 2, 57,
50.0, 1000) pylab.scatter(Y[:, 0], Y[:, 1], 20, labels)
pylab.scatter(Y[:, 0], Y[:, 1], 20) pylab.show()


print("Iteração 3000, Perplexidade = 5") Y = tsne(X, 2, 57,
5.0, 3000) pylab.scatter(Y[:, 0], Y[:, 1], 20, labels)
pylab.scatter(Y[:, 0], Y[:, 1], 20) pylab.show()

print("Iteração 3000, Perplexidade = 25") Y = tsne(X, 2, 57,
20.0, 3000) pylab.scatter(Y[:, 0], Y[:, 1], 20, labels)
pylab.scatter(Y[:, 0], Y[:, 1], 20) pylab.show()

print("Iteração 3000, Perplexidade = 50") Y = tsne(X, 2, 57,
50.0, 3000) pylab.scatter(Y[:, 0], Y[:, 1], 20, labels)
pylab.scatter(Y[:, 0], Y[:, 1], 20) pylab.show()
```

A coleta e a preparação dos dados do Cadastro Único e Bolsa Família

A coleta dos dados para serem visualizados pelo t-SNE foi realizada através de uma consulta usando linguagem SQL na base (folha) de pagamento do Programa Bolsa Família com referência em janeiro de 2019. Em seguida, no Cadastro Único (agosto/2018), obteve-se os dados cadastrais daqueles beneficiários contidos na folha. Por questões de privacidade dos dados dos beneficiários e da estrutura do banco de dados consultado, os *scripts* executados nessa etapa não serão exibidos. Nessa tarefa foram descartados dados pessoais e documentais que pudessem permitir a identificação de cada membro ou família, preservando assim o sigilo pessoal dos indivíduos contemplados na amostra.

Por fim, foram selecionados aleatoriamente 500 indivíduos para comporem a amostra que comporá o arquivo de entrada do algoritmo de redução de dimensionalidade. Como resultado do refinamento, o arquivo resultante contém 57 dimensões.

Os dados alfanuméricos da base extraída também foram excluídos do conjunto de dados de entrada, restando apenas os dados numéricos, que serviram de características para cada indivíduo.

O passo seguinte compreendeu a inclusão, no próprio script do algoritmo disponibilizado por Maaten e Hinton, a normalização dos dados, visto que não existe um padrão de valor mínimo e máximo ao final da fase anterior. Esse passo transformou os números em números decimais (dentro do intervalo 0 a 1) e preparou o arquivo de entrada para ser executado pelo t-SNE.

A preparação da infraestrutura necessária

A execução desse trabalho foi inicialmente realizada com o Anaconda Navigator, versão 1.6.9, usando o ambiente Jupyter Notebook, versão 5.0.0. No entanto, a máquina utilizada possui configuração mediana, com processador intel Core i5 e 8 Gb de memória RAM, sendo considerada incapaz de produzir um resultado satisfatório com grande quantidade de dados. No entanto, foi utilizado posteriormente o ambiente Google Colaboratory, que permite o uso de GPUs/TPUs, tornando possível a continuidade desse trabalho. Para a realização dos testes, foi considerada uma amostra aleatória de 500 registros do total de benefícios.

Resultados e Discussão

Resultados

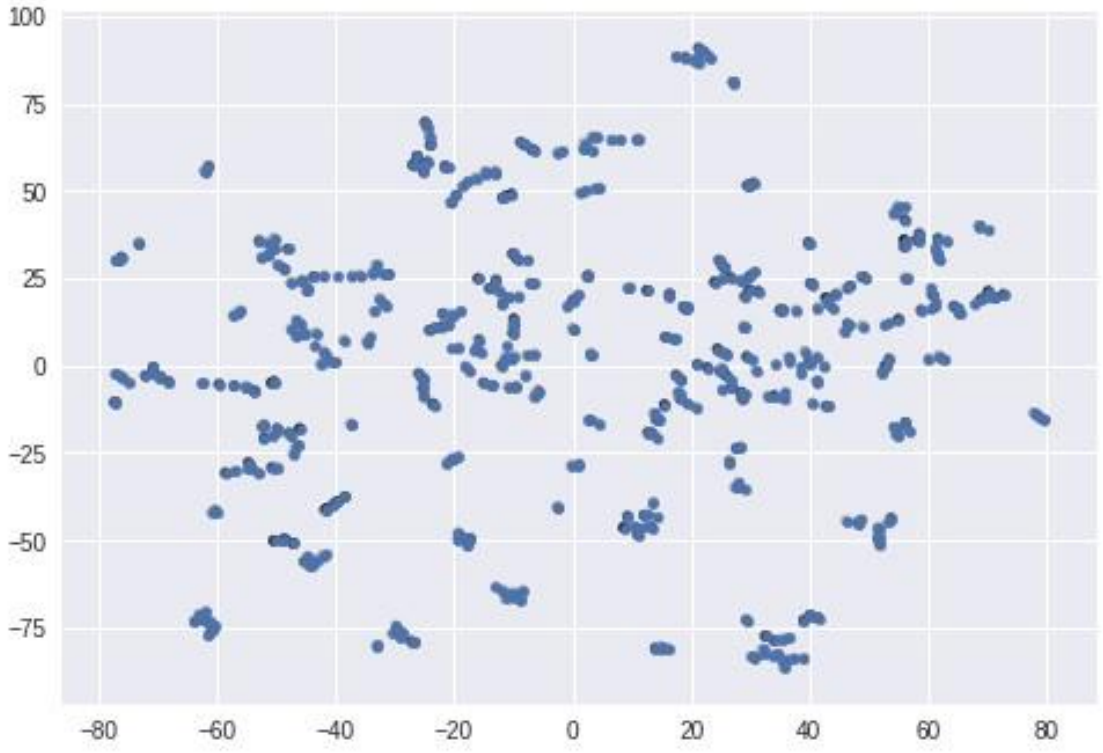
Os resultados obtidos estão a seguir, após sucessivas execuções com diferentes parâmetros de perplexidades e número máximo de iterações:

Visualização nº 1

Perplexidade = 5, Número de iterações: 1000

Iteração 1000, Perplexidade = 5
Preprocessing the data using PCA...
Computing pairwise distances...
Computing P-values for point 0 of 500...
Mean value of sigma: 0.414684
Iteration 1000: error is 0.585783

Figura 2 t-SNE para 500 beneficiários com perplexidade 5 e após 1000 iterações



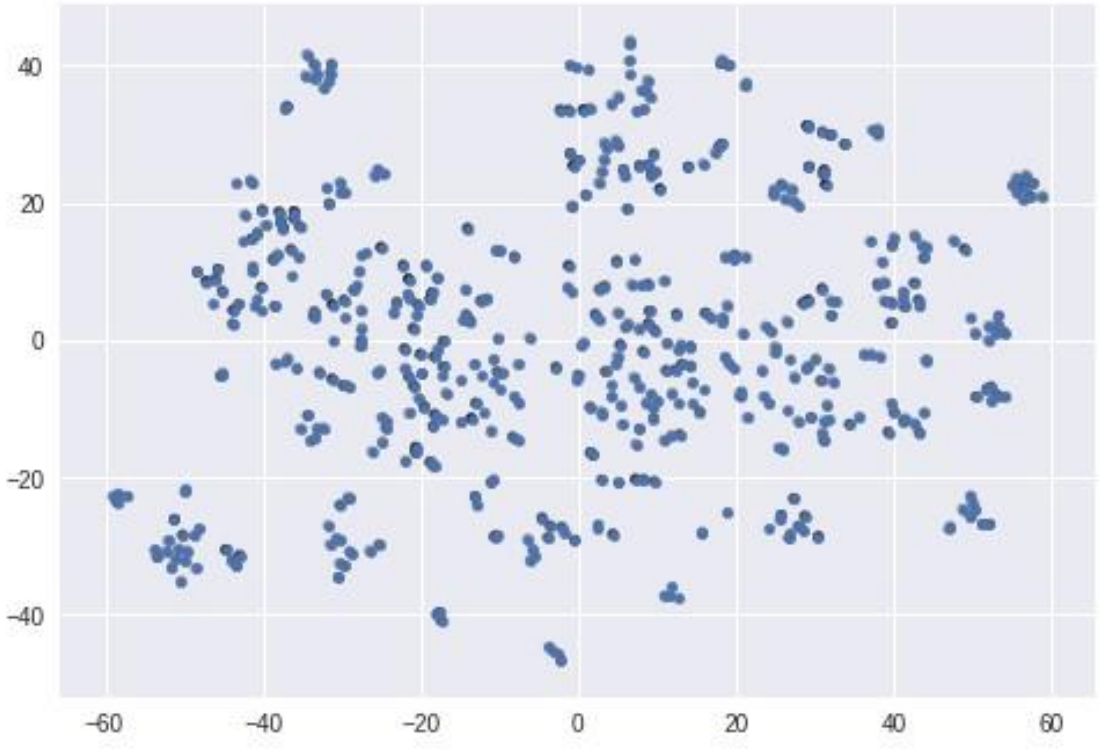
Fonte: Elaborado pelo autor

Visualização nº 2

Perplexidade = 25, Número de iterações: 1000

Iteração 1000, Perplexidade = 25
Preprocessing the data using PCA...
Computing pairwise distances...
Computing P-values for point 0 of 500...
Mean value of sigma: 0.531219
Iteration 1000: error is 0.611099

Figura 3 – t-SNE para 500 beneficiários com perplexidade 25 e após 1000 iterações



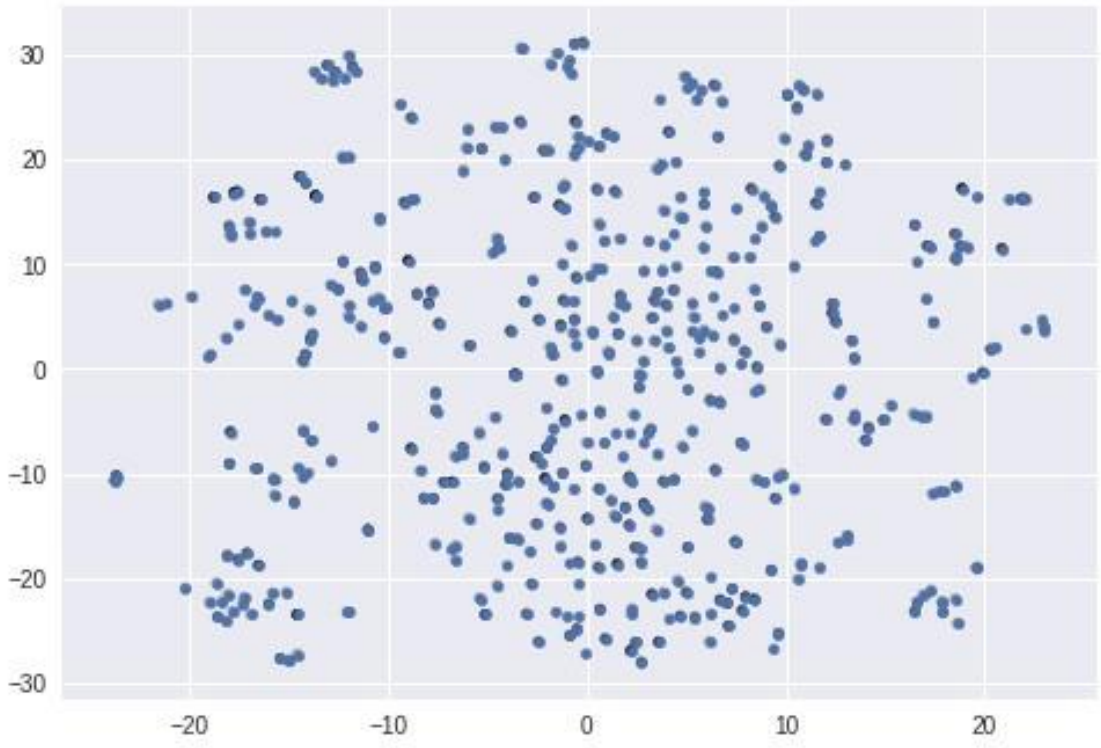
Fonte: Elaborado pelo autor

Visualização nº 3

Perplexidade = 50, Número de iterações: 1000

Iteração 1000, Perplexidade = 50
Preprocessing the data using PCA...
Computing pairwise distances...
Computing P-values for point 0 of 500...
Mean value of sigma: 0.607315
Iteration 1000: error is 0.582678

Figura 4 t-SNE para 500 beneficiários com perplexidade 50 e após 1000 iterações



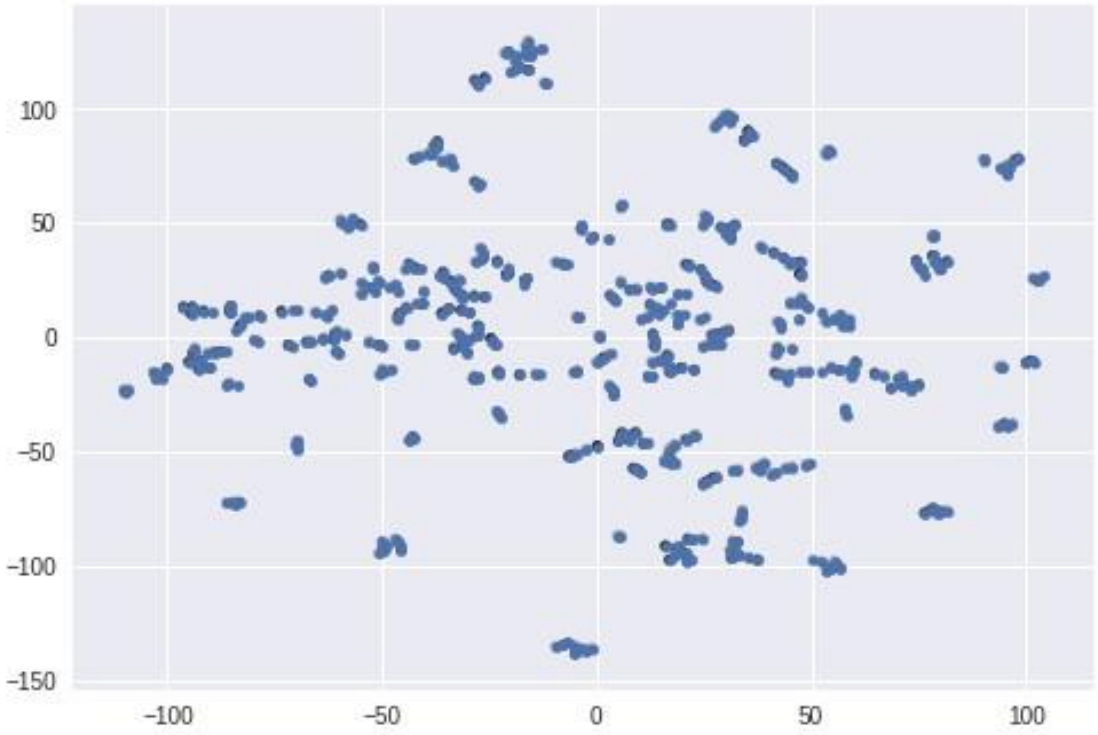
Fonte: Elaborado pelo autor

Visualização nº 4

Perplexidade = 5, Número de iterações: 3000

Iteração 3000, Perplexidade = 5
Preprocessing the data using PCA...
Computing pairwise distances...
Computing P-values for point 0 of 500...
Mean value of sigma: 0.414684
Iteration 3000: error is 0.568455

Figura 5 – t-SNE para 500 beneficiários com perplexidade 5 e após 3000 iterações



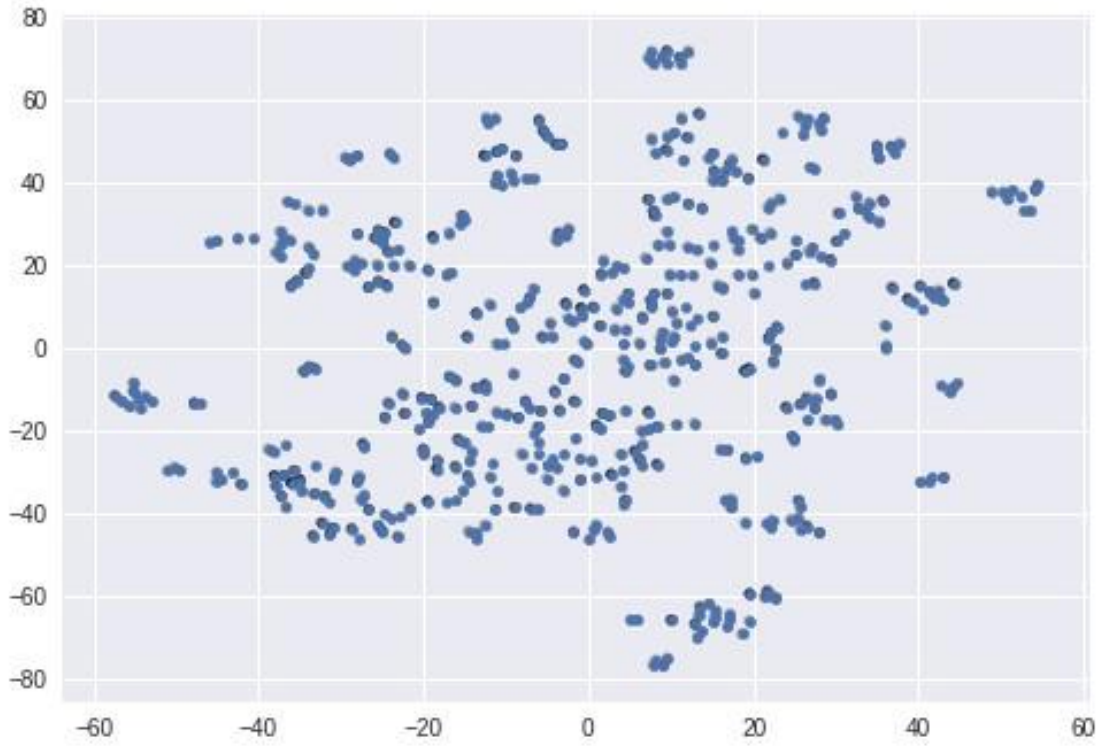
Fonte: Elaborado pelo autor

Visualização nº 5

Perplexidade = 25, Número de iterações: 3000

Iteração 3000, Perplexidade = 25
Preprocessing the data using PCA...
Computing pairwise distances...
Computing P-values for point 0 of 500...
Mean value of sigma: 0.531219
Iteration 3000: error is 0.602319

Figura 6 t-SNE para 500 beneficiários com perplexidade 25 e após 3000 iterações



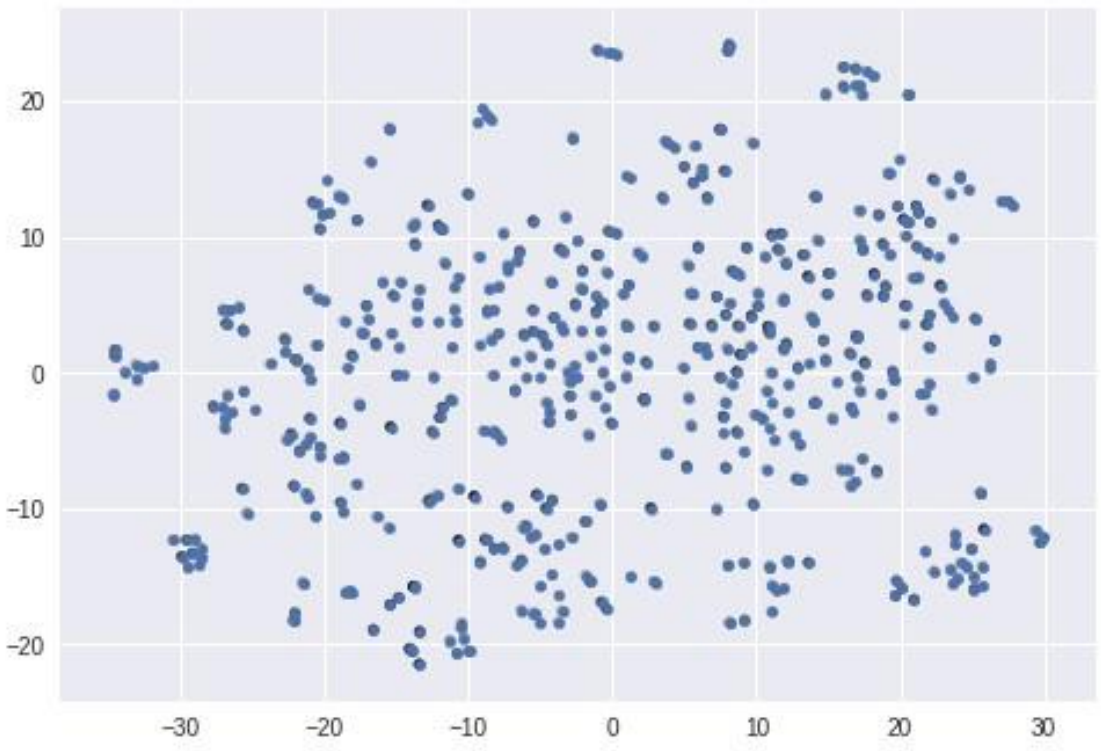
Fonte: Elaborado pelo autor

Visualização nº 6

Perplexidade = 50, Número de iterações: 3000

Iteração 3000, Perplexidade = 50
Preprocessing the data using PCA...
Computing pairwise distances...
Computing P-values for point 0 of 500...
Mean value of sigma: 0.607315
Iteration 3000: error is 0.597460

Figura 7 – t-SNE para 500 beneficiários com perplexidade 50 e após 3000 iterações

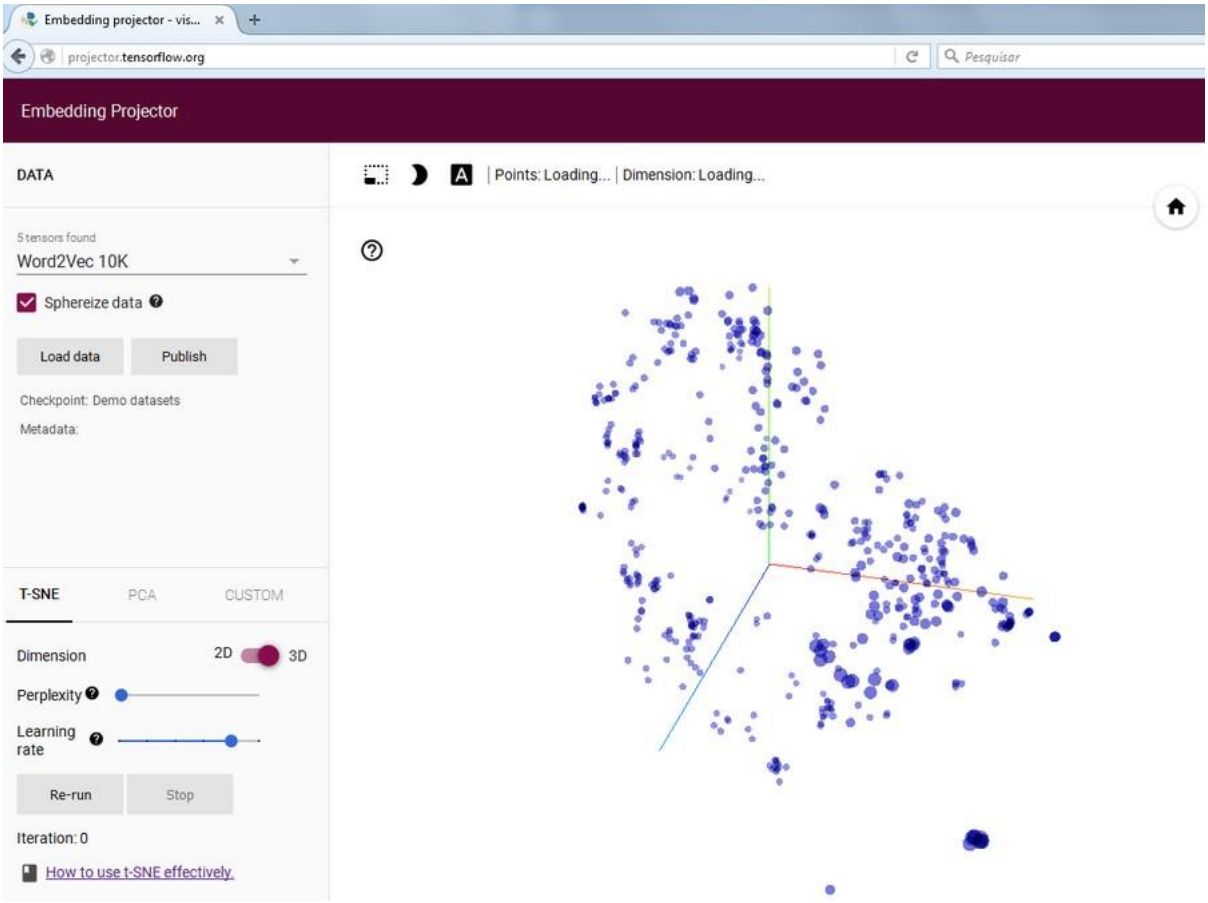


Fonte: Elaborado pelo autor

Visualização nº 7

Utilizando a ferramenta dinâmica online (http://projector.tensorflow.org/):

Figura 8 Resultado utilizando o site http://projector.tensorflow.org/



Fonte: Elaborado pelo autor

Discussão

É fato que o processo de monitoramento e avaliação do PBF poderá se utilizar do t-SNE, testado neste presente trabalho, haja vista a capacidade de permitir a visualização e interpretação dos dados.

Algumas dificuldades foram encontradas durante a elaboração dessa investigação:

1) Identificar se, de fato, os dados escolhidos são relevantes para a visualização de *clusters.*

2) Definir as métricas adequadas do algoritmo, avaliando os momentos de instabilidade durante a execução do algoritmo.

3) Impossibilidade de executar o algoritmo para uma maior quantidade de dados, forçando a inferência do resultado amostral sobre a totalidade dos indivíduos.

4) Possuir infraestrutura com poder computacional suficiente para produzir gráficos dinâmicos, permitindo a visualização do movimento dos pontos ao longo das iterações, com possibilidade de ajustar o parâmetro “perplexidade“ e a taxa de aprendizado.

No futuro, este trabalho pode ter sua continuidade a partir da superação das dificuldades encontradas em sua elaboração e com incremento de novas variáveis e testes. Algumas sugestões para trabalhos futuros:

1) Incrementar os dados dos beneficiários com informações complementares encontradas nos demais registros governamentais.

Utilizar ferramentas que implementam o t-SNE com melhor experiência e facilidade de uso, sobretudo de produzir gráficos 3D interativos. Uma demonstração dessa capacidade pode ser vista em Wattenberg; Viégas; Johnson, (2016) e PROJECTOR.TENSORFLOW.ORG (2019)

Conclusão

Observa-se a capacidade do t-SNE de agrupar as informações ao longo das iterações e das diferentes perplexidades experimentadas. Cada gráfico mostra que as peculiaridades da interpretação, segundo o trabalho de Wattenberg; Viégas; Johnson, (2016), são evidentes, cabendo ao analista perceber o ponto de estabilidade do algoritmo durante a execução.

Mais ainda, é importante registrar que a amostra reduzida pode ter efeito limitador nos resultados, deixando de lado a capacidade de visualizar toda a base de dados pretendida, que no caso desse trabalho é o conjunto total de benefícios do Programa Bolsa Família.Ainda assim, é possível concluir que o método pode ser aplicado para a melhoria dos processos de avaliação e monitoramento de benefícios.

REFERÊNCIAS

MAATEN, L. van der; HINTON, G. SUPPLEMENTAL MATERIAL FOR VISUALIZING DATA USING T-SNE. In: **Journal of Machine Learning Research 9**. [S.l.: s.n.], 2008a. p. 1 – 45.

MAATEN, L. van der; HINTON, G. Visualizing Data using t-SNE. In: **Journal of Machine Learning Research, 9**. [S.l.: s.n.], 2008b. p. 1 – 27.

PROJECTOR.TENSORFLOW.ORG. **projector.tensorflow.org**. Disponível em: <projector.tensorflow.org>. Acesso em: 2019.

ROSSANT, C. 2015. Disponível em: <https://www.oreilly.com/learning/an-illustratedintroduction-to-the-t-sne-algorithm>.

WATTENBERG, M.; VIÉGAS, F.; JOHNSON, I. **How to Use t-SNE Effectively**. 2016. Disponível em: <http://doi.org/10.23915/distill.00002>. Acesso em: 16/02/2019.