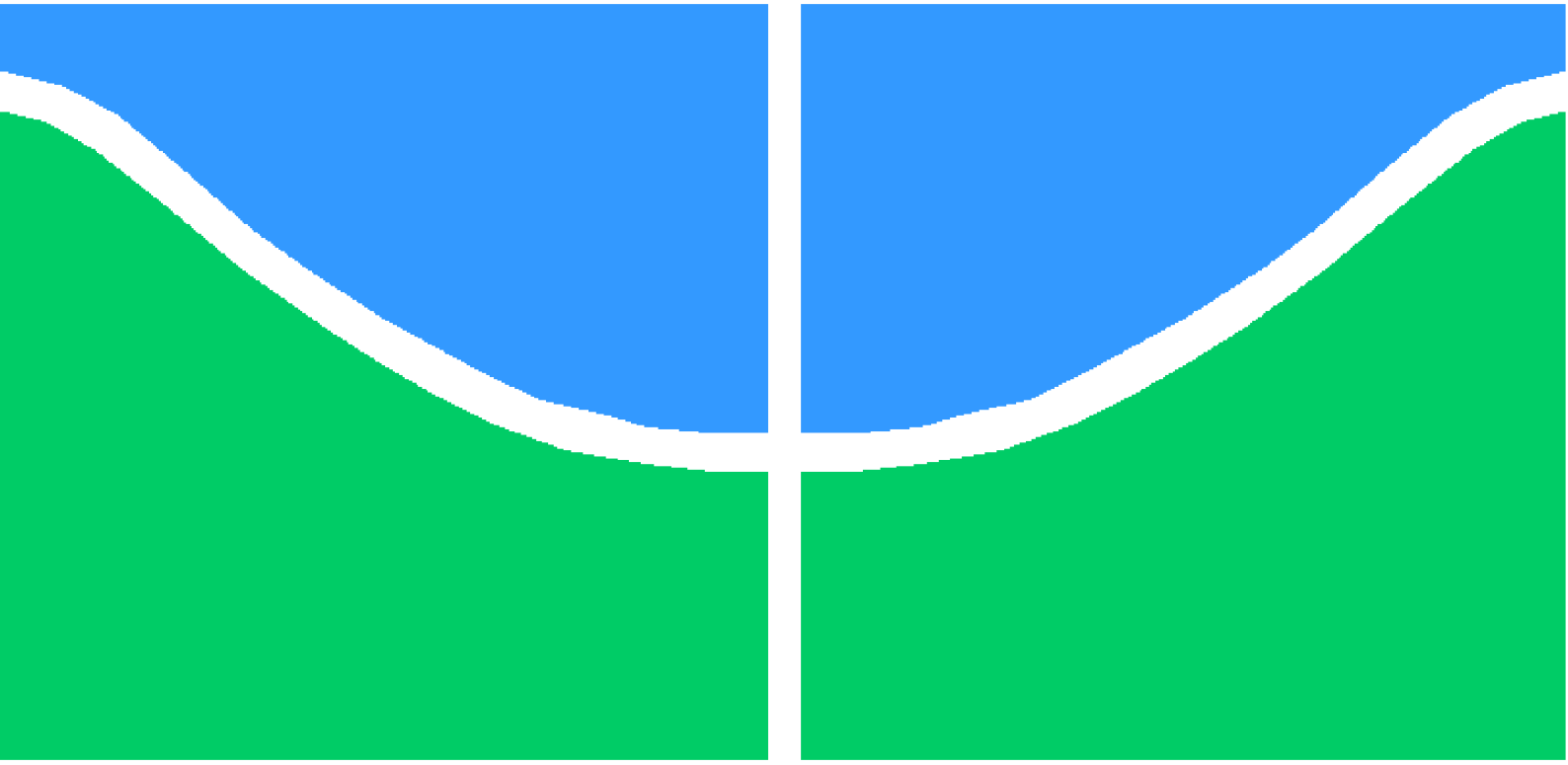

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Modelo de clusterização de dados para identificação de grupos de opinião em uma ferramenta de participação social

Autores: Tallys Gustavo Martins

Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Brasília, DF

2017

Tallys Gustavo Martins

Modelo de clusterização de dados para identificação de grupos de opinião em uma ferramenta de participação social

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Coorientador: Prof. Dr. Fábio Macedo Mendes

Brasília, DF

2017

Tallys Gustavo Martins

Modelo de clusterização de dados para identificação de grupos de opinião em uma ferramenta de participação social/ Tallys Gustavo Martins. – Brasília, DF, 2017-

67 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2017.

1. Clusterização, mineração de dados . I. Prof. Dr. Paulo Roberto Miranda Meirelles. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Modelo de clusterização de dados para identificação de grupos de opinião em uma ferramenta de participação social

CDU 02:141:005.6

Tallys Gustavo Martins

Modelo de clusterização de dados para identificação de grupos de opinião em uma ferramenta de participação social

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 17 de julho de 2017:

Prof. Dr. Paulo Roberto Miranda Meirelles
Orientador

Profa. Dra. Carla Silva Rocha Aguiar
Convidado 1

Prof. Dr. Teófilo de Campos
Convidado 2

Brasília, DF
2017

Agradecimentos

Não é possível viver um só dia sem agradecer. Tenho um carinho enorme por todos que passaram por mim durante os anos da minha graduação, que ajudaram na minha evolução e caminhada até aqui. Quero dizer obrigado a todos os professores do curso de Engenharia de Software da nossa Universidade, que lutaram todos os dias para passar seus conhecimentos e experiências com grande excelência e nos impulsionaram para um futuro melhor, pessoalmente e profissionalmente.

Um agradecimento especial aos professores do laboratório LAPPIS, que engrandeceram e deram oportunidades a muitos alunos com seu excelente trabalho. E também, a meu orientador, prof. Paulo Meirelles e co-orientador, prof. Fábio Macedo, que tiveram bastante paciência e foram grandes tutores no desenvolvimento desse trabalho.

Obrigado aos amigos, família e minha namorada, que dividiram a felicidade dessa conquista juntamente comigo, meus parceiros para toda a vida.

Resumo

A expansão da internet, com seu grande potencial para propagação de informação, juntamente com o crescimento de plataformas de rede social, criaram um ambiente digital cheio de movimentos de expressão e manifestação das opiniões sociais. Nesse mundo online, o desafio de entender e estudar como se comportam esses movimentos e opiniões é grande, principalmente devido ao modelo de interação baseado em "fóruns/threads" adotado pela maioria das ferramentas. Buscando facilitar o estudo das opiniões e comportamento das populações, e também aumentar o engajamento das pessoas para debater suas ideias, as plataformas Polis e Empurrando Juntos abordam uma dinâmica diferente de interação para que os indivíduos expressem seus pontos de vista. Nessa dinâmica, os usuários podem criar um debate sobre algum tema e escrever comentários a respeito do que pensam. Então, outros usuários podem reagir a estes comentários concordando, discordando ou ignorando os mesmos, algo similar à funcionalidade "Curtir" do Facebook. O objetivo desse trabalho é aplicar uma análise com algoritmos de clusterização para identificar grupos de opinião em conversas online criadas sobre essa dinâmica, reconhecendo usuários que reagem aos comentários de maneira parecida.

Palavras-chaves: Clusterização, Participação Social, Debates Online

Abstract

The internet expansion and it's big potential to spread information, together with the growth of the social network platforms, created a digital environment full of social movements and manifestation of opinions. In this connected world, is a big challenge to understand and study how this movements and opinions behave, mainly due to the interaction model based on "foruns/threads" adopted by most of the tools. Aiming to facilitate the studies of the populations behaviour, and also increase the engagement of people who want to debate their ideas, the platforms Polis and Pushing Together address a different dynamic of interaction for individuals to express their thoughts. In this dynamic the users can debate about any topic and write comments about what they think. So others users can react to this comments agreeing, disagreeing or ignoring them, something similar to the Facebook "like" feature. The purpose of this work is to apply an analysis with clustering algorithms to identify groups of opinion on online discussions created in this dynamic, recognizing users that react similarly to the comments.

Key-words: Clustering, Social Participation, Online Discussions

Lista de ilustrações

[1] Imagem retirada do Trabalho de Conclusão de Curso de Emilie T. de Morais e Ítalo P. Batista
[2] Imagem retirada do Trabalho de Conclusão de Curso de curso Emilie T. de Morais e Ítalo P. Batista

Lista de tabelas

Lista de abreviaturas e siglas

TCI	Tecnologias de Comunicação e Informação
EJ	Empurrando Juntos
PCA	*Principal Component Analisys*
SVD	*Single Value Decomposition*
sil	*Coeficiente de Silhueta*
RI	*Rand's Index*
API	*Application Programming Interface*

Sumário

1 Introdução

O uso das Tecnologias de Comunicação e Informação (TCIs) tem sido um grande fator de transformação e influência para a democracia (LéVY, 2002; BENKLER, 2006). Nos últimos anos, testemunhamos diversas manifestações políticas que se organizaram através do uso da Internet e das TCIs, como um meio de participação social. Neste cenário, um desafio constante é descobrir melhores maneiras de atrair e engajar cidadãos comuns juntamente com leigos e ativistas em discussões mais inclusivas e objetivas.

Em 2009, uma consulta pública sobre a Lei 12.965 de 2014, conhecida como Lei do Marco Civil da Internet[1], foi aberta como forma de comunicação entre governantes e população. Nesta consulta, era possível que as pessoas fizessem comentários nos parágrafos de um documento digital que descrevia a lei. O diálogo ocorreu em duas etapas, e foram contabilizados aproximadamente 2.000 comentários em todo o processo. Outras experiências de discussões online, como o debate público sobre o Código de Processo Civil[2], mostram ainda que a maioria dos comentários é feita por uma pequena parcela dos participantes e o nível de engajamento é relativamente baixo. A participação oscila entre algumas dezenas ou centenas de pessoas e comentários.

Discussões que ocorrem em vias comuns, como fóruns e "threads", não provêm uma dinâmica escalável para debates com muitos participantes. O trabalho de leitura para interpretar imensos textos de comentários espanta até mesmo os mais interessados. Além disso, é difícil mapear nesse tipo de conversa as diferentes opinião formadas, ou ainda quais são maioria ou minoria dentro da conversa. Uma plataforma para discussões online deve ser capaz de prover isso aos participantes.

Apresentamos então, o projeto Empurrando Juntos, descrito no Capítulo 3, como uma ferramenta para debates online que se baseia em um conceito diferente para estruturação da conversa. Nesse conceito, que aqui chamamos de participação "crowdsource", os participantes podem expressar sua opinião em pequenas ações e com pouco esforço, algo comparado a funcionalidade "Curtir" do Facebook. Dessa maneira, a participação se torna uma ação escalável a milhares de usuários ao empenho de um clique. Outras experiências de consultas públicas, como ArenaNetMundial[3], mostraram que esta é uma ótima opção para quebrar as barreiras do engajamento e abrir mais espaço para discussões.

Nesse paradigma de conversa, a opinião dos participantes pode ser composta em termos de concordo e discordo, 1 ou -1, "*like*" e "*dislike*". Com isso, podemos construir uma correlação da opinião de cada usuário em cada comentário, o que torna possível o

[1] <https://itsrio.org/wp-content/uploads/2017/02/marco_civil_construcao_aplicacao.pdf>
[2] <http://www.migalhas.com.br/arquivo_artigo/art20120210-05.pdf>
[3] <http://www.participa.br/articles/public/0007/0286/resultado-consulta-publica-pt.pdf>

uso de algoritmos de agrupamento de dados para obter análises sofisticadas sobre o que as pessoas pensam sobre um assunto em uma discussão.

Este trabalho apresenta um estudo feito sobre esses algoritmos e ferramentas necessárias para o desenvolvimento de um modelo de clusters que identifique grupos de opinião em uma plataforma de participação social.

1.1 Justificativa

Sabendo da importância da tecnologia nos dias de hoje e acreditando no seu potencial como forma de inovação para o bem social, este trabalho oferece uma colaboração para o projeto Empurrando Juntos na busca de uma solução que acrescente novas possibilidades para debates democráticos.

Complementarmente, esta pesquisa reunirá um conhecimento acerca de algoritmos de clusterização de forma que possa servir de insumo para outros estudos relacionados.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo principal desse trabalho é propor um algoritmo que identifique grupos de opinião em uma discussão online, com base nas reações (concordo, discordo, pular) de usuários aos comentários da discussão feito por outros usuários.

Numa primeira etapa foi feito um estudo teórico sobre estratégias de clusterização, para adquirir um domínio sobre os diferentes modelos e algoritmos. A partir deste estudo, desenvolvemos soluções baseadas nos algoritmos selecionados.

Executada a parte prática de implementação, será feita uma avaliação dos resultados comparando-os com resultados esperados, obtidos a partir de um modelo estatístico e um índice de validação para algoritmos de clusterização.

1.2.2 Objetivos Específicos

Buscando alcançar e satisfazer o objetivo geral, foram definidos seis objetivos específicos, os quais são descritos a seguir:

- Definir critérios da solução
- Efetuar estudo técnico sobre clusterização de dados
- Selecionar técnicas de clusterização a serem aplicadas

- Definir tecnologias e arquitetura para a solução
- Construir solução com base nos critérios definidos
- Analisar resultados da solução

1.3 Organização do Trabalho

O desenvolvimento do trabalho é dividido em 6 Capítulos. Primeiramente, no Capítulo 2, apresentamos uma revisão bibliográfica a respeito de algoritmos de clusterização. Logo após, nos Capítulos 3 e 4, é dada uma contextualização do problema junto ao contexto da plataforma Empurrando Juntos e uma proposta de solução é apresentada. Por fim, nos Capítulos 5 e 6, são apresentados respectivamente os resultados atingidos e conclusões sobre estes resultados, bem como suas limitações, pesquisas futuras e outras considerações finais.

2 Clusterização de dados

2.1 Definição

Sistemas de Recomendação e Recuperação de Informação comumente usam técnicas e metodologias de clusterização em atividades de Mineração de Dados pra reconhecer e extrair padrões em um conjunto de dados.

Nesse tipo de análise, o objetivo é organizar objetos em grupos ou clusters, de forma que os objetos do mesmo cluster possuam uma semelhança significativa entre si por algum tipo de critério estabelecido (GAN; MA; WU, 2007). Imaginando então alguns objetos sobre uma mesa, podemos querer agrupá-los por seu tamanho, cor, forma, ou ainda por uma combinação de todas as três características. É possível realizar essa separação apenas observando padrões sobre as qualidades dos objetos, não sendo necessário nenhum treinamento anterior para identificar os diferentes padrões. Na linguagem de aprendizado de máquina, essa técnica é descrita como uma abordagem de classificação não supervisionada.

O agrupamento de objetos é determinado em função de uma medida de distância. Desta forma, o objetivo de um algoritmo de clusterização é encontrar e classificar itens similares de maneira que a distância intra-cluster (soma das distância dos objetos de um cluster até o seu centro) seja a menor possível e a distância entre diferentes clusters seja maximizada, (AMATRIAIN et al., 2011).

Esta abordagem se encaixa bem como uma alternativa para agrupar usuários que votam de maneira parecida em comentários de uma conversa, pois consegue gerar uma análise sobre os dados de modo a tratar individualmente cada conversa com seu universo de características particular, como número de usuários, tema da conversa, número de comentários e votos feitos por estes, o que não seria adequado fazer com uma abordagem supervisionada.

2.2 Representação dos dados

Para que seja possível analisar um certo agrupamento de dados, é necessário identificar quais informações podem ser utilizadas para representar uma abstração prática dos objetos a serem agrupados. É preciso encontrar então um conjunto de atributos que descrevem os itens para que seja possível calcular o grau de semelhança entre os elementos. Esses atributos, por sua vez, podem ser escritos de forma numérica, categórica, binária e uma variedade de outros tipos. Como exemplo, podemos representar pessoas em vetores

compostos pelo valor numérico do ano de nascimento e um atributo binário, 0 ou 1, para masculino ou feminino.

$$\begin{bmatrix} pessoaA \\ pessoaB \\ pessoaC \\ .... \\ pessoaN \end{bmatrix} = \begin{bmatrix} 1992 & 0 \\ 2005 & 1 \\ 1993 & 1 \\ \vdots & \vdots \\ 1985 & 0 \end{bmatrix}$$

Vejamos também, como a representação de um mesmo atributo, a cor de um objeto, pode existir em diferentes formatos e escalas.

Tabela 1 – Representação nominal sobre a cor.

Objeto	**Cor**
A	Azul
B	Amarelo
C	Vermelho

Tabela 2 – Representação categórica para o atributo cor.

Objeto	**Azul**	**Amarelo**	**Vermelho**
A	1	0	0
B	0	1	0
C	0	0	1

Tabela 3 – Representação numérica representada pela frequência da cor em Terahertz.

Objeto	**Cor (THz)**
A	606
B	508
C	400

As Tabelas 1, 2 e 3 mostram os mesmos dados representados de formas diferentes, o formato depende basicamente da fonte dos dados, de como eles foram armazenados. Para as diversas representações existe um tipo de função de distância compatível. É possível, no entanto, utilizar qualquer uma das representações. Gan, Ma e Wu (2007) detalham uma série de outros tipos e escalas de representação dos objetos.

2.3 Transformação dos dados

Na maioria das aplicações do mundo real, o conjunto de dados precisa passar por algum tipo de tratamento antes de ser analisado (AMATRIAIN et al., 2011). Por exemplo, em casos onde a medida de similaridade é sensível a diferenças de magnitude e escalas das variáveis de entrada, como a distância Euclideana, faz-se necessário uma normalização de seus valores. Em outras situações onde os atributos dos objetos são de diferentes tipos, como categóricos e numéricos, há a necessidade de convertê-los para a mesma escala, ou buscar uma medida de similaridade que consiga tratar ambos os tipos de entradas.

Além de simples conversões de escala e normalizações, pode ser adequado aplicar transformações mais complexas no conjunto de dados, como a redução de dimensionalidade. Para esta tarefa, existem várias técnicas, de complexidade linear e não linear, que podem gerar resultados diferentes de acordo com as características dos dados. No geral, uma técnica bastante utilizada para esta transformação e que também é usada neste trabalho é a Análise dos Componentes Principais (PCA).

PCA é um método estatístico que permite representar em termos de outros eixos, com dimensões reduzidas. Estes eixos servem de base para uma projeção dos dados originais, ou seja, perde-se informação. Porém, essa projeção pode ainda representar suficientemente a quantidade de informação necessária para identificação de padrões e agrupamentos nos dados.

Para encontrar os eixos ou componentes principais, calcula-se os autovetores e autovalores da matriz de covariância dos dados, resultando em um autovetor para cada variável existente. Os respectivos autovalores descrevem a variância relativa ao conjunto total. Pode-se então ignorar os componentes com menor contribuição para a variância para obter os dados projetados com dimensão reduzida. A Figura 1 é um diagrama dos passos da análise dos componentes principais. Mais detalhes sobre esta técnica podem ser vistos no Apêndice B.1.

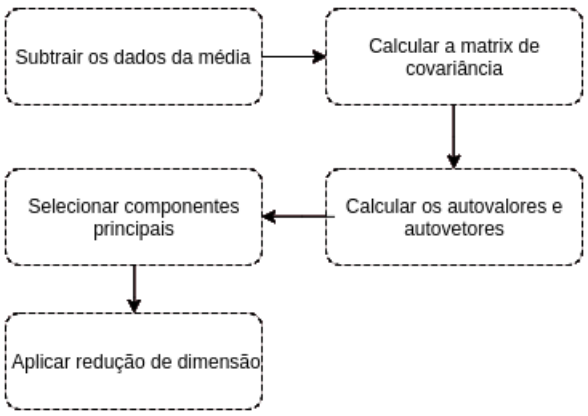

Figura 1 – Aplicação do PCA.

2.4 Medidas de Similaridade

Um ponto importante para análise de grupos com clusterização é definir uma maneira para medir a similaridade entre os itens, pois para que seja possível identificar elementos similares e agrupá-los em clusters, é preciso utilizar alguma função que calcule, de forma quantitativa, o grau de aproximação entre dois objetos distintos (SEGARAN, 2007). A escolha da função de distância é um passo importante, e geralmente se dá por uma combinação de experiência, conhecimento e sorte, (GAN; MA; WU, 2007).

Existem formas de cálculo que se adequam melhor a diferentes tipos de dados, como cálculos específicos para atributos numéricos, categóricos, binários, temporais, e também fórmulas que combinam atributos mistos. Uma medida muito comum, usada em vários contextos e que também faz parte da nossa proposta, é a distância Euclidiana simples, aplicável a dados numéricos. Esta medida é definida por:

$$\mathrm{d}(x, y) = \sqrt{\sum_{k=1}^{N}(x_k - y_k)^2}$$

Onde N significa a quantidade de atributos e x_k e y_k são os k-ésimos atributos que compõe os objetos x e k respectivamente.

Se quisermos, por exemplo, calcular a similaridade entre entre sete pessoas considerando as notas que deram em avaliações sobre dois filmes, podemos representá-las conforme a Figura 2. Os eixos representam as notas dos filmes em uma escala de 0 a 5 e os pontos representam cada usuário. Pessoas que avaliam de maneira parecida ficam mais próximos uns dos outras no gráfico.

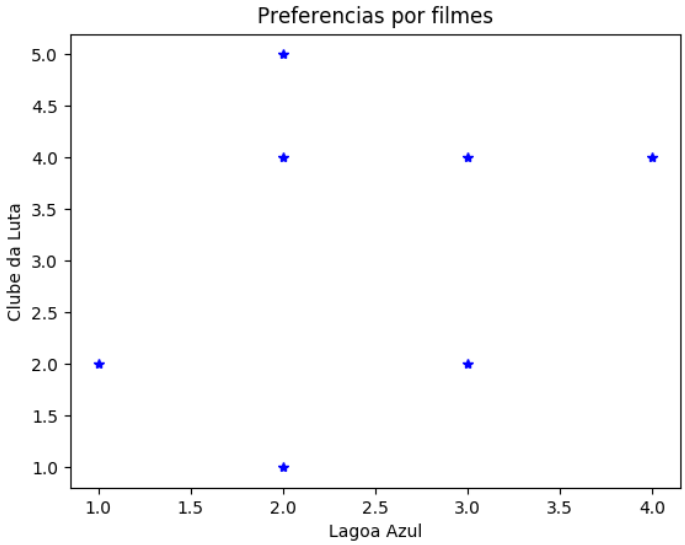

Figura 2 – Representação de sete pessoas e suas avaliações.

Para calcular a proximidade entre os usuários podemos utilizar os valores das coordenadas de notas com na distância euclidiana e obter um número absoluto, e quanto mais próximo de 0 esse valor, mais similares são as pessoas.

Um dos problemas com esta medida é que atributos com uma escala maior que outros podem influenciar negativamente no resultado final, distorcendo-a. Por exemplo, se a nota de uma pessoa é em uma escala de 0 a 50 e a de outra é dada de 0 a 5, elas terão um peso diferente no cálculo da semelhança, já que seus resultados se baseiam nos quadrados da raiz. Em cenários com variáveis em escalas diferentes onde, por exemplo, uma delas mede o tempo e a outra mede a distância, é necessário fazer uma normalização dos atributos para minimizar a distorção, porém perde-se a informação de escala.

Outras medidas para dados numéricos e outros tipos de dados são apresentadas no Apêndice A.

2.5 Modelos de Análise

A noção de cluster é algo que não possui uma definição precisa, e por isso há uma grande variedade de algoritmos e abordagens para modelar problemas de agrupamento de dados. Diferentes pesquisadores aplicam diferentes modelos de clusters, e para cada modelo existem diversos algoritmos. As propriedades que definem o que é um cluster variam significativamente de um para outro, bem como complexidade e aplicação dos mesmos. Entender efetivamente estes modelos é a chave para entender os diferentes algoritmos. Gan, Ma e Wu (2007) sumariza os conceitos de várias abordagens existentes, como modelos hierárquicos, modelos baseados em grafos, densidade e outros.

Para se ter uma ideia da variedade de resultados possíveis, vejamos a Figura 3 que exibe a aplicação de diferentes algoritmos sobre os mesmos conjuntos de dados. Estes algoritmos são implementados na biblioteca scikit-learn[1], escrita em Python, e são apenas uma parte do universo total de opções. Podemos ver em cada uma das quatro linhas um conjuntos de dados diferente e em cada coluna temos o resultado da aplicação de um algoritmo naquele conjunto. Os modelos e algoritmos aplicados são os baseados em centroide (MiniBatchKmeans, MeanShift), algoritmos baseados em grafo (Spectral), alguns da classe de modelos hierárquicos (Ward, AglomerativeClustering), e também um que faz análise de densidade (DBSCAN).

A escolha do modelo geralmente parte da observação e experiências de aplicação em problemas semelhantes, já que conhecer e testar diferentes abordagens e suas variações não é uma tarefa simples. Então, neste trabalho nos limitaremos ao estudo e características do modelo de algoritmos baseados em centroide.

[1] <http://scikit-learn.org/stable/modules/clustering.html>

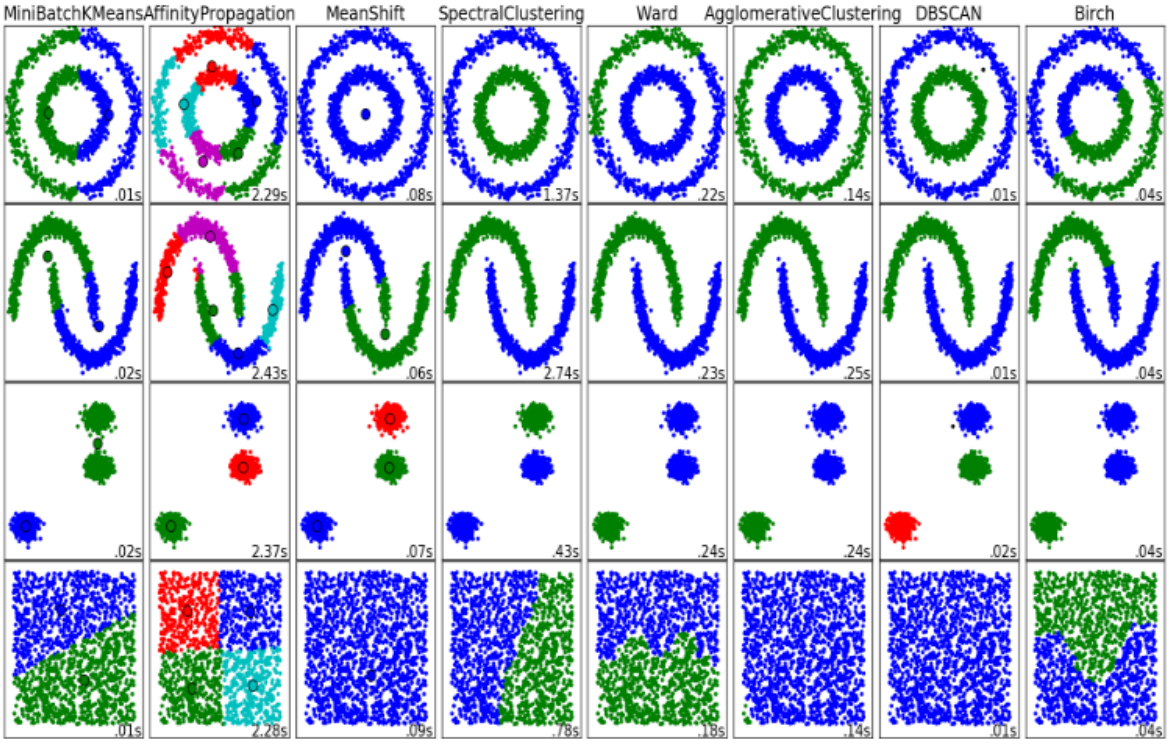

Figura 3 – Vários modelos de algoritmos aplicados nos mesmos dados.
Fonte: SCIKIT-LEARN... (2016)

2.5.1 Modelo centroide

Algoritmos baseados em centroides são muito eficientes em clusterização de grandes bases de dados e com alta dimensionalidade quando comparados a outros modelos (GAN; MA; WU, 2007), embora não sejam muito adequados para encontrar clusters com formatos arbitrários. Esse tipo de algoritmo representa cada cluster através das coordenadas de seu centro e se guia por uma função objetivo que define o quão boa é uma solução calculada e, a cada iteração, procuram agrupar membros aos clusters de forma que a função seja minimizada. Um clássico algoritmo desse modelo é o k-means.

O k-means é um algoritmo de clusterização bastante utilizado, rápido e de fácil implementação, (GAN; MA; WU, 2007). Ele requer o fornecimento de um parâmetro k que determina o número de clusters desejado, e uma função de erro é associada à sua execução. Desta forma, dado um k inicial, são operadas várias iterações que alocam repetidamente os dados ao conjunto mais próximo, até que não haja alterações significativas na função de erro ou nos membros de cada cluster. A descrição do algoritmo é vista a seguir.

Seja D um conjunto de dados qualquer, e seja C_1, C_2, ..., C_k os k clusters disjuntos de D, então a função de erro é definida por:

$$E = \sum_{i=1}^{K} \sum_{k=1}^{N} d(X_k, \mu(C_i))$$

Onde o resultado é a soma das distâncias entre cada elemento X e o centro de seu cluster.

O algoritmo é dado em duas fases, inicialização e iteração. Na fase de inicialização são selecionados k pontos aleatórios como centroides iniciais e os pontos são distribuídos randomicamente nos k grupos. Na segunda fase, de iteração, associa-se cada ponto ao cluster de centro mais próximo, e ao final, os centros de cada cluster são atualizados em função dos seus membros. A fase de iteração é repetida até que não haja mudança nos clusters ou até que seja atingido um limite do número de iterações pré-definido. Esse algoritmo pode então ser utilizado para calcular k grupos de opinião em uma conversa.

−> Inicialização
Escolha k pontos, dentro do conjunto de pontos como centroides iniciais
Associe cada ponto a um cluster aleatório
−> Iteração
Escolha k pontos, dentro do conjunto de pontos como centroides iniciais
Repita:
  Forme k clusters movendo os pontos para o cluster de centro mais próximo
  Recalcule o centro de cada cluster baseado nos pontos pertencentes ao mesmo
Até que os centroides não mudem.

A complexidade computacional do algoritmo é $O(nkd)$ por iteração, onde d é a dimensão dos dados, k é o número de clusters e n é o número de pontos no conjunto de dados e sua solução converge para uma ótima local e não global, dado que o problema resolvido pelo algoritmo é de natureza NP-difícil e seria inviável testar todas as soluções possíveis para então optar pela melhor. Desta forma, a qualidade da solução é bastante dependente dos k centros escolhidos na fase de inicialização.

2.6 Validação

Vários algoritmos são propostos na literatura de clusterização para diferentes aplicações e diferentes tamanhos de dados. Por ser um processo não supervisionado, não existem classes pré-definidas e nem exemplos que possam mostrar que os clusters encontrados pelo algoritmo são válidos, (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2002). E ainda, validar o número ótimo de clusters é um grande desafio quando este parâmetro não é dado pelo próprio algoritmo.

De fato, não existe um conceito de análise que seja absoluto para avaliação de algoritmos de clusterização, mas existem uma série de medidas que vêm de diversas áreas,

como estatística e visão computacional, que podem ser aplicadas (WALDE, 2006). Duas principais perspectivas devem ser levadas em conta para avaliação do modelo, avaliação interna e externa, ambas são aplicadas nesse trabalho.

2.6.1 Validação interna

No contexto de clusterização de maneira geral, quando não existem dados já classificados, de modo manual ou automático, que possam ser utilizados como referência para a saída esperada do algoritmo, aplicam-se métricas de avaliação interna. Nesses casos, o tipo de avaliação feito usa medidas que testam o modelo sobre ele próprio, o qual é executado com diferentes parâmetros e uma análise sobre seus resultados é aplicada com um algum índice que avalia, por exemplo, a densidade dos clusters gerados, ou a distância mínima entre os clusters, etc.

O coeficiente de silhueta se encaixa nesse contexto, e é bastante utilizado como medida de qualidade para auxiliar na seleção do melhor número de clusters k em algoritmos que requerem este parâmetro, como k-means. Essa métrica calcula um índice de avaliação para cada elemento do conjunto de dados. Para obter o coeficiente de silhueta sil de um determinado objeto i pertencente a um cluster C_A, é comparada a distância média a entre o objeto i e os outros objetos contidos no mesmo cluster C_A com a distância média b entre i e todos os objetos de um cluster vizinho mais próximo C_B. Assim, de acordo com a equação, aplica-se $-1 < sil(i) < 1$ para cada objeto. Se $sil(i)$ é um valor alto, então a média das distâncias dentro do cluster é menor que a média de distâncias até o cluster vizinho, então os objeto i foi bem classificado, caso contrário, ele foi mal classificado.

$$a(i) = \frac{1}{|C_A| - 1} \sum_{j \in C_A, i \neq j} d(i, j)$$

$$b(i) = min_{C_B \neq C_A} \frac{1}{|C_B|} \sum_{j \in C_B} d(i, j)$$

$$sil(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

O coeficiente individual de cada objeto pode ainda ser expandido para um coeficiente para cada cluster ou ainda um coeficiente geral de todo o conjunto aplicado a um número k de clusters. Para isso, basta tirar as médias para a amostragem desejada, por cluster ou por todo o conjunto:

$$sil(C_p) = \frac{1}{C_p} \sum_{j \in C_p} sil(j)$$

$$sil(k) = \frac{1}{k}\sum_{p=1}^{k} sil(C_p)$$

Outra medida bastante conhecida que pode ser utilizada para avaliação interna é a Soma do Erro Quadrado. Com ela, busca-se quais parâmetros do algoritmo minimizam esta função, o que indicaria clusters mais homogêneos. A soma dos erros quadrados E refere-se primariamente ao uso da distância Euclideana, mas é aplicável também a outras medidas, e é dada pela soma das distâncias de cada ponto até o centro de seu cluster:

$$E(C) = \sum_{i=1}^{k} \sum_{O \in C_i} d(O_i, cen_i)^2$$

2.6.2 Validação externa

Quando possuímos valores de referência externos, que podem ser utilizados como resultado esperado, podemos então avaliar o modelo aplicando métricas similares às utilizadas na categoria de algoritmos supervisionados para avaliar uma taxa de acerto do algoritmo.

O Índice de Rand é uma medida para calcular a acurácia da classificação feita pelo algoritmo em relação a uma saída esperada esperada. Rand (1971) define esta métrica de avaliação para problemas gerais de clusterização com base em uma análise sobre a correspondência entre os pares de objetos classificados. Dadas duas partições diferentes C e M feitas sobre um mesmo conjunto de dados:

Tabela 4 – Partição esperada × gerada para um cenário hipotético.

Classificação Esperada (M)	Classificação do Algoritmo (C)
M1={a,b,c} M2={d,e,f}	C1 = {a,b} C2 = {c,d,e} C3 = {f}

Sejam, A, B, C e D dados como:

* A → Número de pares de objetos no mesmo grupo tanto em M quanto em C, ou seja, $[(a,b),(d,e)] = 2$

* B → Número de pares de objetos em grupos separados tanto em M quanto em C, ou seja, $[(a,d),(a,e),(a,f),(b,d),(b,e),(b,f),(c,f)] = 7$

* C → Número de pares de objetos que estão no mesmo grupo em M, mas em diferentes grupos em C, ou seja, $[(a,c),(b,c),(d,f),(e,f)] = 4$

* D → Número de pares de objetos que estão em grupos diferentes em M, mas no mesmo grupo em C, ou seja, $[(c,d),(c,e)] = 2$

Então, pela definição do índice de Rand, a acurácia da partição C gerada pelo algoritmo em relação à partição esperada M é:

$$Rand(C,M) = \frac{A+B}{A+B+C+D} = \frac{A+B}{\binom{n}{2}} = \frac{2+7}{2+7+4+3} = 0.6$$

Hubert e Arabie (1985) definem ainda um ajuste para a medida do índice de Rand, de forma que seu limite de valores fique entre $-1 \leq indice \leq 1$, mostrando que, em resultados negativos, siginifica que as partições comparadas apresentam comportamento aleatório.

Estas são apenas algumas das várias medidas existentes que podem auxiliar na avaliação de modelos e algoritmos de clusterização. Para escolha da medida adequada apresentamos na próxima Seção, um estudo comparativo feito por Walde (2006) sobre métricas de validação.

2.6.3 Comparação entre medidas de validação

Na atividade de validação, Walde (2006) define quatro demandas que guiam um estudo comparativo entre métricas de qualidade de clusters.

1. Dado que o propósito da avaliação é comparar diferentes experimentos de clusterização e resultados, a métrica deve ser aplicável a todas as distâncias de similaridade usadas em clusterização, e ser também independente e não enviesada com a respectiva medida.

2. A métrica deve definir uma medida numérica indicando o valor que deve ser de fácil interpretação ou ilustração quanto à sua extensão e efeitos, para facilitar a avaliação.

3. A métrica de avaliação deve ser intransigente quanto à especificação do número e tamanho dos clusters.

4. A métrica de avaliação deve ser capaz de distinguir a qualidade individual de cada cluster e a qualidade total de todo o particionamento.

Com essas demandas, Walde (2006) monta também uma tabela comparativa entre algumas métricas com os seguintes levantamentos:

▷ Precisa de um valor de referência?

▷ É aplicável a todas as medidas de similaridade?

▷ É independente da medida de similaridade?

▷ Distingue valores para cada cluster e geral pra toda a partição?

▷ É enviesada ao número de clusters?

▷ É sensível à adição de erros (monotônica)?

▷ Qual é o intervalo de valores?

Tabela 5 – Comparação de métricas de avaliação, (WALDE, 2006) adaptado.

Métrica	Referência	Medida de Similaridade		Valor	
		Aplicável	Independente	Específico	Geral
Erro Quadrado	não	sim	não	sim	sim
Silhueta	não	sim	não	sim	sim
Rand Index	sim	sim	sim	não	sim
Rand Index$_{ajustado}$	sim	sim	sim	não	sim
Informação Mútua	sim	sim	sim	sim	sim
Fowlkes-Mallow	sim	sim	sim	não	sim

Métrica	Bias	Sensivel a Erros(monotônica)	Interpretação	
			Mín	Máx
Erro Quadrado	muito pequeno	não	0	∞
Silhueta	muito pequeno	não	-1	1
Rand Index	muito pequeno	sim	0	1
Rand Index$_{ajustado}$	não afetado	sim	-1	1
Informação Mútua	muito pequeno	sim	0	∞
Fowlkes-Mallow	não afetado	sim	0	1

A principal distinção entre cada medida é se ela necessita ou não de valores de referência. As medidas do Error Quadrado e Silhueta, não precisam destes dados, elas medem a qualidade dos clusters em referência aos próprios dados clusterizados, e são muito utilizadas para auxiliar para definição do melhor número de clusters K em algoritmos que requerem esse parâmetro.

A Tabela 5 mostra os pontos fortes e fracos individuais de cada métrica. Uma escolha mais adequada é uma medida que dê a qualidade de cada cluster e um valor geral para toda a partição, optando por métricas que possuam um intervalo de valores definido, o que facilita a interpretação. É importante também perceber se esta é afetada pelo tamanho dos clusters, causando um *bias*, e se é capaz de detectar a introdução de erros de maneira monotônica.

Como estamos desenvolvendo uma estratégia de clusterização baseada nos algoritmos do modelo centroide, que requer o número k de clusters, precisamos de uma medida

que nos auxilie para definição do melhor valor de k, como a Silhueta. E para compararmos os resultados obtidos com valores de referência gerados manualmente, o Índice de Rand ajustado e a medida de Fowlkes-Mallow são boas opções que podem ser aplicados. Essas medidas fornecem insumos que irão nos auxiliar na construção de um algoritmo que tenha uma maior coerência nos seus resultados.

3 Uma plataforma para participação social

Nascido em um Hackathon, a plataforma Empurrando Juntos é uma ferramenta de participação social que permite criação de debates online. A proposta, ainda em desenvolvimento, foi uma das 8 ideias de projetos acolhidas pelo laboratório de Inteligência Coletiva para a Democracia, organizado pelo MediaLab Prado, em Madri, na Espanha, em setembro de 2016 e foi idealizado por representantes da organização não governamental Cidades Democráticas. Seu objetivo é tornar os processos de participação online mais atraentes e acessíveis em um sistema que incentive o engajamento do máximo de pessoas nesses processos. A ferramenta foi concebida como uma extensão de uma outra plataforma já existente, de código aberto, chamada Polis.

3.1 Polis

O Polis é uma ferramenta criada para auxiliar organizações e comunidades a entenderem de forma visual quais são e como estão divididas suas opiniões, pensamentos e ideias. Seu sistema web oferece aos usuários a possibilidade de se criar discussões online sobre qualquer tema. Estes podem então expressar e escrever seus pontos de vista em comentários de 140 caracteres que são apresentados aos demais usuários como tópicos em cartões.

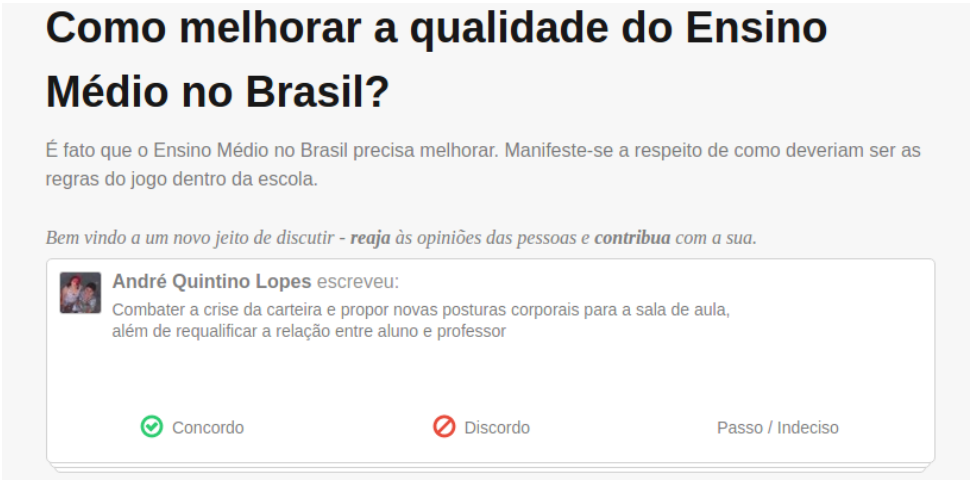

Figura 4 – Cartões com comentários.

A Figura 4 mostra uma parte da tela do sistema em uma conversa. O título da conversa é visto no topo, em destaque, e uma descrição é dada logo em seguida. Então, é mostrada uma sequência de comentários para que o usuário dê sua opinião em cada um deles. É possível concordar, discordar ou pular um dado comentário, e a cada resposta, um

novo comentário subsequente é apresentado. Também é possível participar adicionando novos comentários à conversa.

Desta forma, a ferramenta utiliza algoritmos de inteligência artificial para identificar usuários que votam de maneira parecida e os coloca em em grupos de opinião. A análise feita sobre os votos resulta também em dados sobre os comentários de maior concordância e discordância, tanto pela maioria geral quanto por cada grupo. A Figura 5 abaixo mostra os grupos formados em uma conversa.

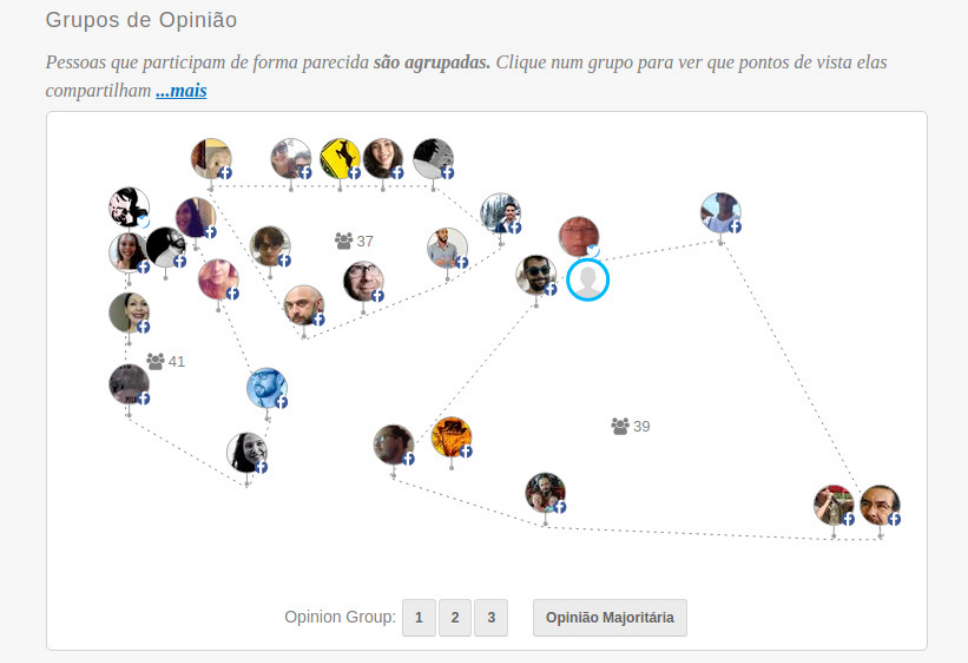

Figura 5 – Três diferentes grupos de opinião em uma conversa.

3.2 Plataforma Empurrando Juntos

O projeto EJ enxerga o potencial da plataforma Polis e seus recursos no contexto de uma ferramenta de participação social que incentive o engajamento e permita novos tipos de análises sobre a opinião de quem participa. Assim, sua proposta surgiu com a criação de um novo sistema, integrado à plataforma Polis através de sua API[1], agindo como uma extensão da mesma.

A ferramenta EJ oferece novos conceitos e funcionalidades para as conversas criadas com o Polis, provendo uma outra plataforma que objetiva uma maior integração da sociedade em debates online. Com isso, o projeto utiliza a ferramenta Polis como serviço para criação de conversas e identificação de grupos de opinião, e insere nestas conversas outras formas de interação entre os participantes.

[1] <https://pol.is/api.html>

Um destes conceitos é um recurso denominado “Push”. Os usuários em posse deste “poder” são capazes de enviar mensagens e criar eventos dentro da plataforma, algo que o Polis não provê.

Nesse contexto, existem três perfis de usuários especiais que são considerados os potenciais receptores do “Push” e seus efeitos a partir da formação dos grupos de opinião pelo Polis. A Figura 6 abaixo mostra os três perfis e seus poderes dentro da ferramenta Empurrando Juntos:

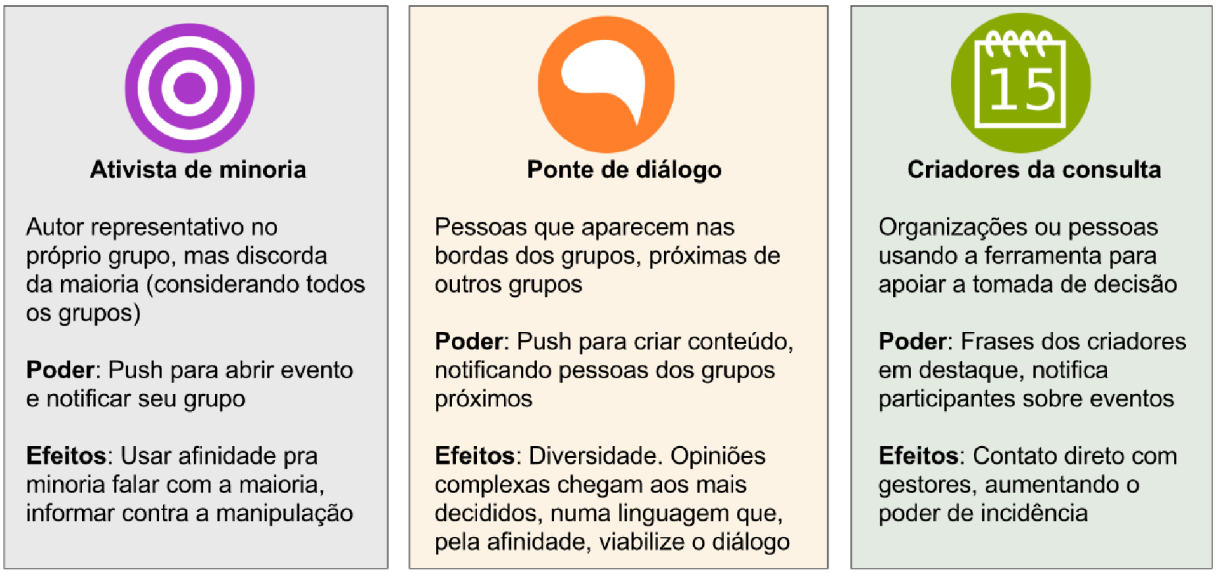

Figura 6 – Perfis considerados para recepção do “Push”.

Nos aspectos técnicos, o projeto é dividido em uma aplicação front-end, escrita em React Native e um servidor desenvolvido em NodeJS. Nesta arquitetura, o módulo servidor é responsável por gerenciar os usuários, o recurso “Push” e por fazer a interface com o serviço externo de clusterização feito pelo Polis. A Figura 7 representa esta estrutura:

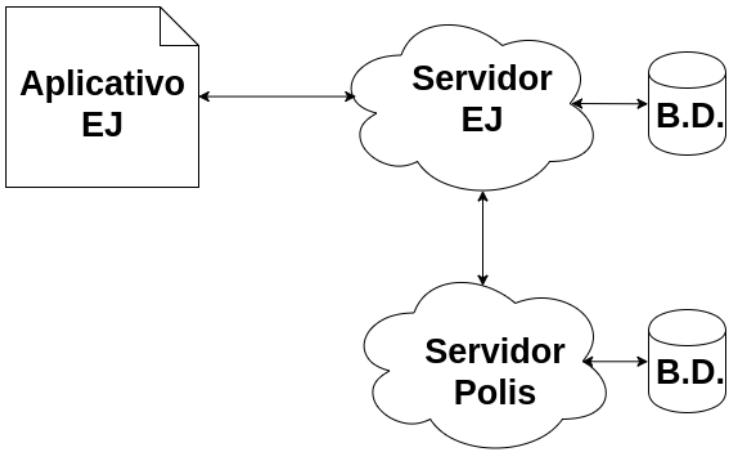

Figura 7 – Arquitetura simplificada do sistema.

No entanto, a integração das duas ferramentas está condicionada a algumas limitações da plataforma Polis. Apesar de ter seu código aberto, ela é bastante complexa, de difícil manutenção e implantação, o que é uma grande barreira para a continuação do projeto. Essas limitações despertaram então o interesse pelo desenvolvimento de um novo serviço de clusterização que atenda aos seus critérios e aspirações. Esses critérios foram estabelecidos durante sua concepção durante o laboratório em Madri.

1. **Código aberto:** O novo serviço de clusterização deve ser aberto, com licença livre para aplicação e estudo de terceiros;
2. **Tratamento dos Votos:** O novo serviço de clusterização deve ser capaz de manipular votos que figurem a Concordância, Discordância e Abstenção de opinião de um usuário em um comentário, de maneira similar à plataforma Polis.
3. **Grupos de opinião:** O novo serviço de clusterização deve ser capaz de identificar grupos de opinião em uma conversa com base nos votos dos usuários, que possam ser visualizados em gráficos similares aos da plataforma Polis.
4. **Opinião majoritária:** O novo serviço de clusterização deve ser capaz de identificar comentários que representem a opinião majoritária de todos os participantes, de maneira similar à plataforma Polis.
5. **Perfil ativista de minoria:** O novo serviço de clusterização deve ser capaz de identificar usuários com perfis de ativista de minoria em uma conversa.
6. **Perfil ponte de diálogo:** O novo serviço de clusterização deve ser capaz de identificar usuários com perfis de ponte de diálogo em uma conversa.

Os critérios 4, 5 e 6, no entanto, não estão no escopo deste trabalho. Então, para o desenvolvimento do modelo, serão levados em consideração apenas os critérios 1, 2 e 3. Para abranger esses critérios, construiremos uma aplicação simples, que provê a criação de conversas, comentários e de votos para os usuários. Nesta aplicação será implementado os algoritmos de clusterização para identificar os grupos de opinião e gerar resultados que possam ser exibidos em gráficos de duas dimensões, como o Polis. Esta aplicação, por sua vez, poderá ser adotada como alternativa para o serviço de clusterização utilizado pelo EJ.

4 Proposta de solução

Até este capítulo do trabalho foram apresentados diversos termos, técnicas e algoritmos existentes na literatura de clusterização de dados. Com este estudo feito, utilizaremos este conhecimento técnico para buscar responder então à seguinte pergunta: Como identificar grupos de opinião em uma conversa entre usuários?

Este capítulo apresenta como será desenvolvida a solução que responde à dada pergunta. Passamos primeiramente pelo tópico que diz respeito à representação dos itens e definição de seus atributos. Em seguida, trataremos da fase de pré-processamento dos dados e posteriormente a escolha de um algoritmo de análise de clusters que identifique os grupos de opinião almejados.

Para criar a aplicação que contempla nossa proposta, utilizaremos o *framework* Django[1] e a biblioteca científica Scikit-learn[2], que oferece diversos algoritmos de aprendizado de máquina. Elas são ferramentas livres, escritas na linguagem Python e irão facilitar no desenvolvimento.

4.1 Representação dos usuários

Para representar os usuários do sistema e posteriormente identificar os grupos com base nos votos, consideramos então os itens/objetos como sendo os usuários participantes e os comentários da conversa como sendo os atributos de cada usuário. Para o valor de cada atributo, utilizaremos a faixa de limites $[-1, 0, 1]$, correspondentes às possíveis escolhas de interação pelos usuários.

Discordo	Pular/Não Informado	Concordo
-1	0	1

Assim, podemos construir uma matriz "participantes × comentários" com os valores dos votos de cada participante, resultando em um conjunto de dados N-dimensional que é dependente do número de comentários e participantes na conversa, como pode ser visto na Tabela 6, a seguir:

Um importante ponto a favor sobre esta representação é que ela em si, contém toda a informação necessária para fazer a análise, dispensando ajustes de escala ou tratamento adicional de dados ausentes, já que consideramos como zeros os casos em que não há votos do usuário em algum dos comentários.

[1] <djangoproject.com>
[2] <scikit-learn.org>

Tabela 6 – Tabela de votos dos participantes (P) em comentários (C) de uma conversa.

	C1	C2	C3	C4	C5	...	Cn
P1	0	1	-1	-1	-1	...	-1
P2	1	0	0	1	1	...	1
P3	1	1	-1	0	1	...	0
P4	-1	1	1	0	1	...	1
P5	1	1	1	0	1	...	1
PN	1	1	-1	1	0	...	0

Vale ressaltar que a ausência de dados pode afetar diretamente no resultado do agrupamento, pois seu valor impacta na escolha e no cálculo das distâncias entre os objetos. A escolha desta representação foi inspirada na implementação da plataforma Polis e nos critérios (Seção 3.2) do projeto Empurrando Juntos.

4.2 Pré-processamento e transformação dos dados

O critério 3, descrito na Seção 3.2, define que: "O novo serviço de clusterização deve ser capaz de identificar grupos de opinião em uma conversa com base nos votos dos usuários, que possam ser visualizados em gráficos similares aos da plataforma Polis." Este critério, afirma uma necessidade de visualização dos agrupamentos em um gráfico de duas dimensões. Visto que o conjunto de dados é representado em múltiplas dimensões, proporcional ao número de comentários, faz-se necessário uma transformação de redução da dimensionalidade destes para algo tangível à percepção humana. Nesse sentido, optamos por aplicar uma redução também para 2 dimensões, utilizando a técnica PCA, descrita na Seção 2.3.

Para uma melhor qualidade dos resultados, seria ideal executar a clusterização utilizando todas as dimensões dos dados. Porém, no momento da visualização em duas dimensões, os dados estariam suscetíveis à superposição dos clusters. Esse problema pode acontecer quando há um cluster "atrás de outro" em uma dimensão extra não mostrada na projeção 2D, como pode ser observado na Figura 8, onde os centros dos clusters estão distorcidos após aplicar o PCA nos dados já clusterizados:

Por isso, aplicaremos a transformação de redução anteriormente ao cálculo dos clusters. Sabemos também, que esta redução gera outros impactos no sistema. O principal deles, é uma distorção que acarreta em uma certa perda de informação, já que se trata de uma projeção dos dados originais. Além disso, haverá ainda uma mudança na escala dos dados. Uma vez que o valores estavam limitados aos dígitos [-1, 0, 1], após a transformação eles se tornam elementos em uma faixa contínua, o que impacta diretamente na escolha do algoritmo e da função de distância a serem utilizados. As Tabelas 7 e 8 mostram uma transformação dos dados usando o PCA.

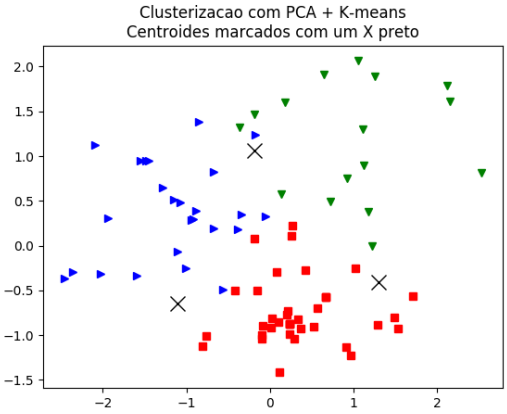

Figura 8 – Distorção dos centros.

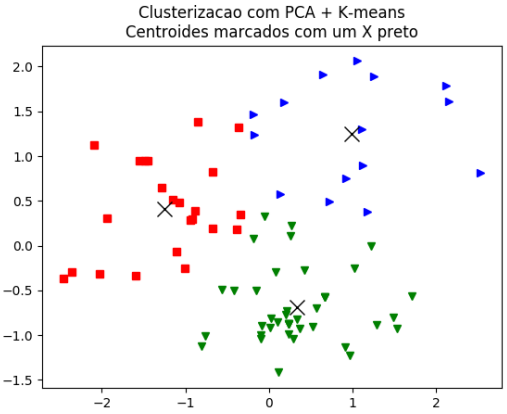

Figura 9 – Sem superposição.

Tabela 7 – Conjunto original.

	C1	C2	C3	C4	C5	C6	C7	C8
P1	0	1	-1	-1	-1	-1	-1	-1
P2	1	0	0	1	1	1	1	1
P3	1	1	-1	0	1	0	1	1
P4	-1	1	1	0	1	0	-1	0
P5	1	1	1	0	1	1	-1	0

Tabela 8 – Conjunto reduzido.

	D1	D2
P1	2.59	-1.18
P2	-2.08	-0.26
P3	-1.12	-1.38
P4	0.81	1.51
P5	-0.20	1.32

O PCA acaba descartando os comentários com poucas respostas, já que agregam uma baixa variância ao conjunto de dados. Aplicando a redução, obteremos um melhor desempenho do algoritmo de clusterização, já que utiliza-se apenas duas dimensões, o que otimiza o cálculo das distâncias entre os pontos, na verificação das similaridades.

4.3 Modelo de clusters

Após a representação dos objetos, ponto de partida para a implementação de qualquer modelo de clusterização, o próximo passo é escolher o algoritmo que irá executar a tarefa. O algoritmo para a ferramenta Empurrando Juntos, deve considerar que novos usuários comentam e votam a todo momento em uma dada conversa, processando um conjunto de dados de N participantes que votaram em N comentários. A Figura 10 ilustra como se dá esse fluxo.

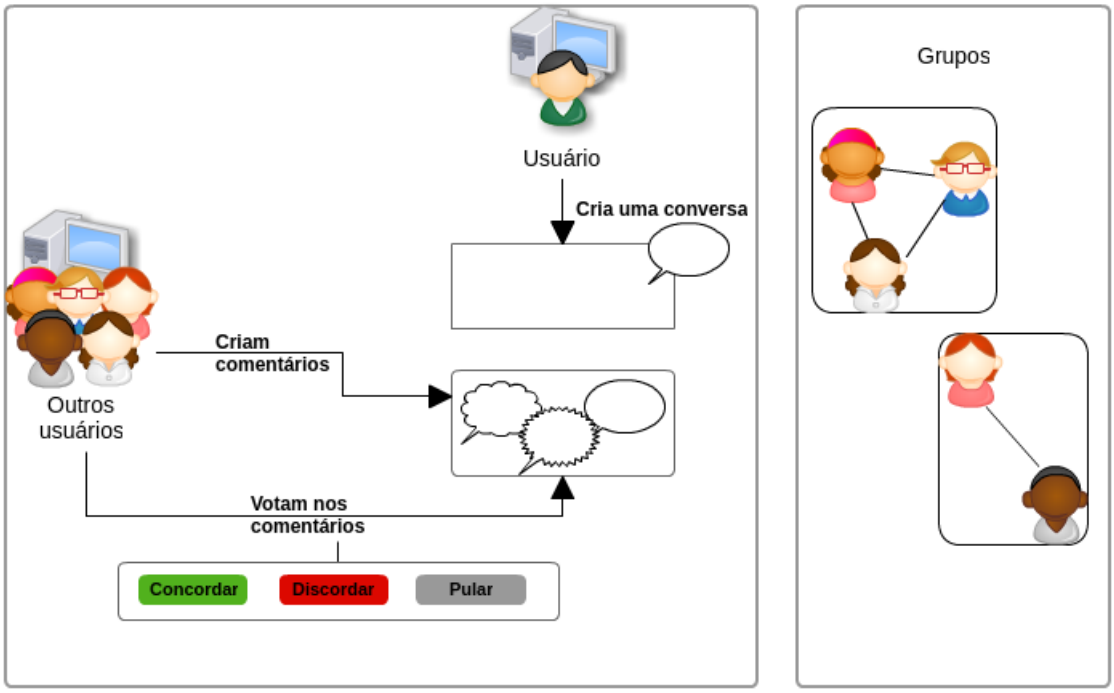

Figura 10 – Fluxo de participantes em uma conversa.

Dado o contexto e os critérios definidos na seção 3.2, e também características citadas dos modelos baseados em centroide, será aplicado o algoritmo *k-means* em conjunto com a técnica de silhueta para a escolha do melhor resultado. Assim, o modelo responderá a cada novo voto em uma conversa, considerando um k máximo de até 5 grupos na conversa e irá retornar o cenário de maior coeficiente de silhueta, o que indicaria a melhor divisão dos clusters. Esse valor de k foi determinado pensando-se na visualização dos grupos, mas certamente irá passar por refinamentos futuros.

Tratando-se da função de similaridade que será utilizada para aferir a aproximação entre os usuários, escolhemos a equação Euclidiana simples, vista na Seção 2.4. Essa escolha, por sua vez, se encaixa com a transformação feita nos dados com a técnica PCA, visto que os dados transformados passam a ser numéricos e não mais categóricos, e também porque não há uma diferença de escala entre o valor das variáveis, sendo este um cenário plausível para a utilização desta medida.

A Figura 11 representa o esquemático do modelo. O fluxo começa a partir de novos votos em comentários da conversa, que disparam o algoritmo de clusterização. O k-means é executado 5 vezes com os parâmetros de k igual a 1, 2, 3, 4 e 5 e o coeficiente de silhueta é calculado para cada uma das execuções. Ao final, o particionamento de maior coeficiente é salvo como resultado.

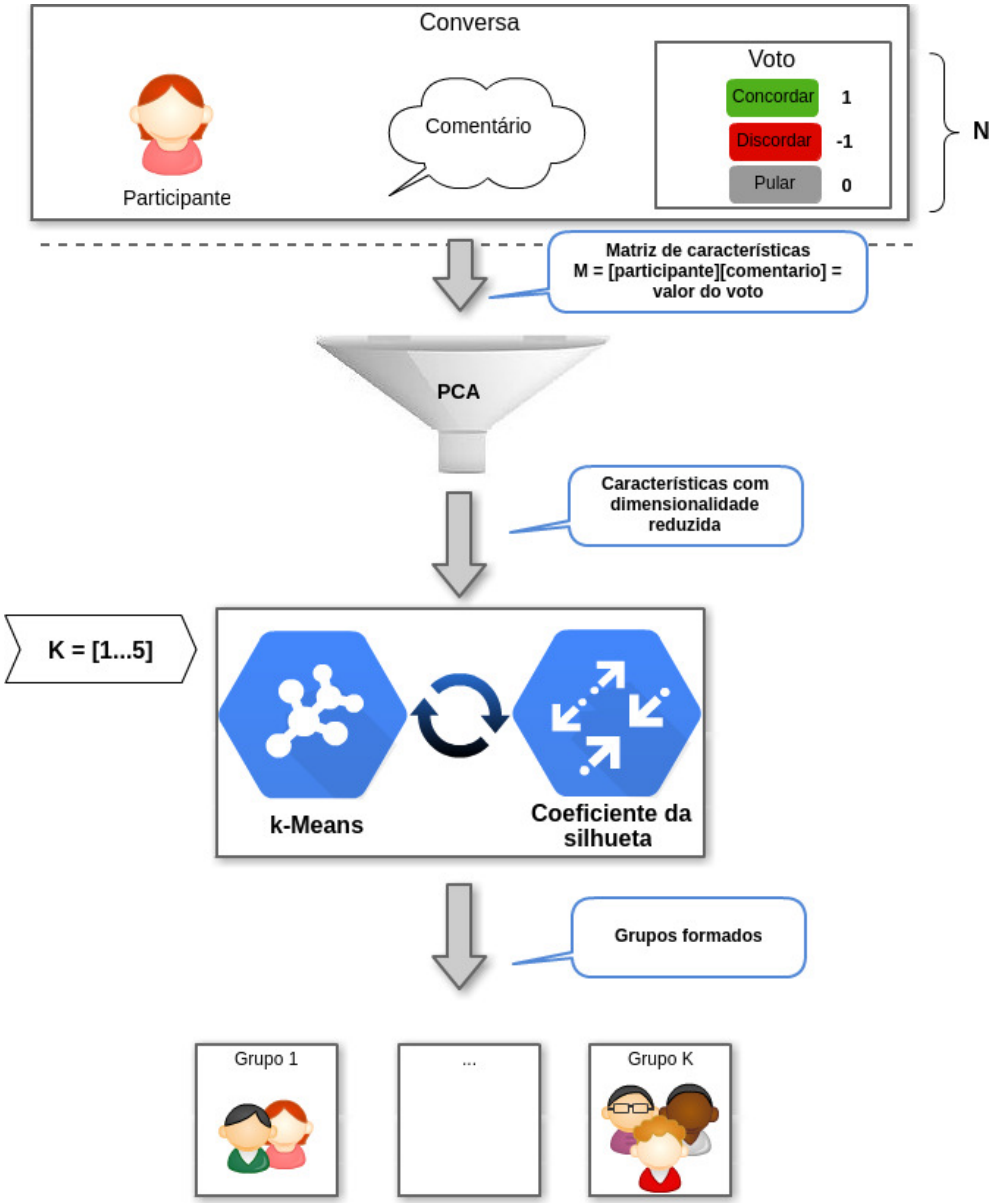

Figura 11 – Ilustração do modelo.

Tanto o algoritmo *k*-means, quanto o coeficiente de silhueta são implementados na biblioteca Scikit-learn e serão utilizados para construir o modelo. A vantagem de se utilizar uma biblioteca madura e amplamente testada é que podemos ter uma maior confiança nos seus resultados e na sua eficiência.

5 Implementação da Solução

Este capítulo descreve a etapa prática de desenvolvimento do software que contempla a solução proposta e resultados obtidos. São apresentados aspectos técnicos e decisões tomadas para se atingir o objetivo do trabalho de desenvolver um modelo de algoritmo capaz de identificar grupos de opinião em um debate online. Detalhamos a seguir como foi desenvolvida a aplicação que é capaz de gerenciar conversas, comentários e votos de usuários e identificar grupos de opinião avaliando usuários que votam de maneira parecida. Vale ressaltar que neste trabalho focamos no desenvolvimento da lógica do modelo e não foram desenvolvidos nenhum meio de interação externa com o sistema ou interface gráfica.

5.1 Sistema Django

Como já citado, o Django é um *framework* para a criação de sistemas web, o qual utilizamos para o desenvolvimento da aplicação. Foram criados dois módulos: o módulo **Conversations** e o módulo **Clusters**.

O módulo Conversations contém a lógica e estrutura que trata das conversas, votos e comentários. Já o módulo Clusters possui os algoritmos de clusterização, responsável por fazer o processamento dos dados das conversas. Consideramos então as seguintes relações para o desenvolvimento do sistema:

1. Um usuário pode criar várias conversas;
2. Um usuário pode criar vários comentários em uma conversa;
3. Um usuário pode votar em vários comentários de uma conversa;
4. Um usuário que cria um comentário ou um voto em uma conversa se torna um participante desta;
5. Um usuário que cria um comentário em uma conversa automaticamente recebe um voto de concordância com o mesmo;
6. Um usuário pode participar de várias conversas;
7. Uma conversa possui vários usuários participantes;
8. Uma conversa possui uma clusterização associada;

Com essas relações em mente, construimos diagrama de entidades e relacionamentos que mapeia conceitualmente no banco de dados, a ligação entre os objetos da aplicação. A Figura 12 mostra esse diagrama.

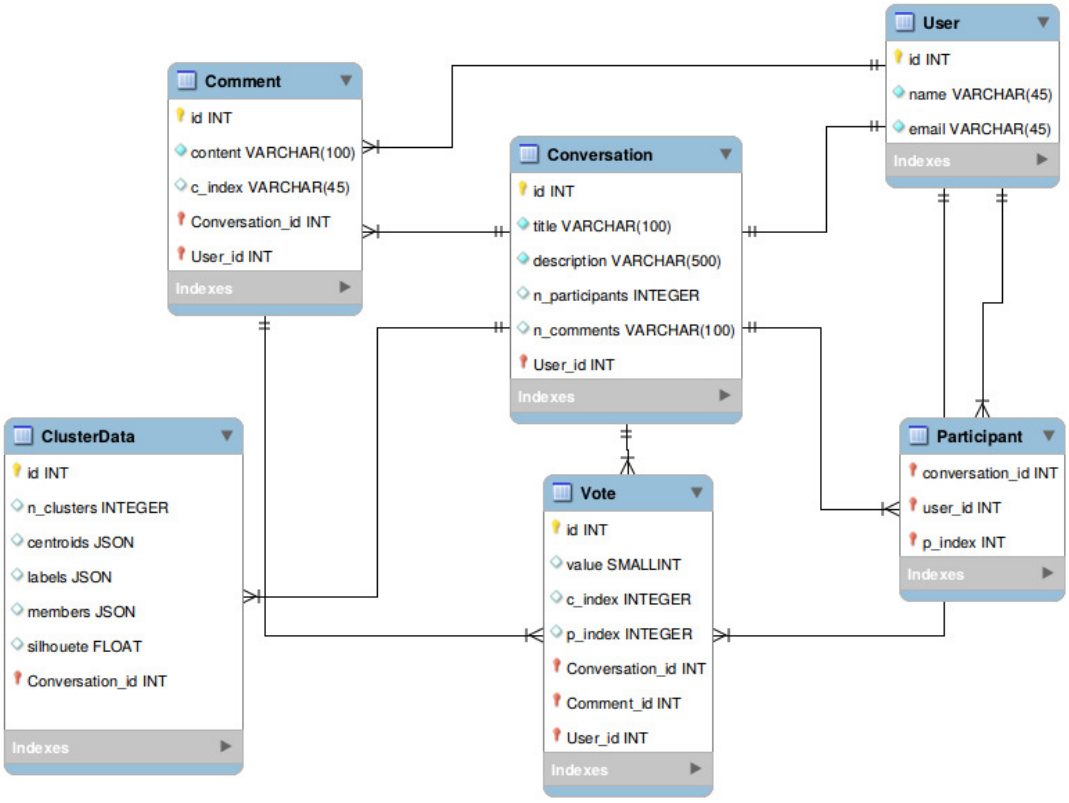

Figura 12 – Diagrama de Entidades e Relacionamentos - DER.

As entidades Conversation, Vote, Comment e Participant, fazem parte do módulo Conversations. A entidade ClusterData faz parte do módulo Clusters, que é o componente que contempla toda a lógica do modelo de clusterização. O módulo Clusters possui também outras duas classes, que não são persistidas no banco de dados e por isso não aparecem no diagrama da Figura 12 acima. Essas classes compartilham da responsabilidade de manipular os dados para o cálculo dos clusters e merecem ser apresentadas:

- **ClusterManager:** responsável por iniciar a computação dos clusters de uma conversa a partir de um novo voto em um comentário. A partir dela são recuperados os dados de uma conversa, aplicados os algoritmos e persistidos os resultados.
- **ConversationDataMatrix:** oferece recursos para recuperação e preparação do conjunto de dados de uma conversa (participantes × comentários), oferecendo também as transformações necessárias aos dados, como a redução de dimensionalidade através do PCA.
- **ClusterData:** esta classe guarda abstração dos resultados de uma clusterização. Ela armazena informações do cálculo como, número de clusters, coordenadas dos

centros de cada clusters, membros de cada cluster e coeficiente de silhueta. Essa classe tem suas informações persistidas no banco de dados.

Cada módulo foi concebido segundo a arquitetura de *apps* do framework Django. A estrutura dos *apps* é representada conforme a Figura 13, as classes de cada *app* em retângulos brancos e funções em vermelho. Estabelecemos assim, uma comunicação entre os dois apps através de um sinal que é enviado para a classe ClusterManager cada vez que um novo voto é inserido no sistema, agindo como um ativador para cálculo dos clusters.

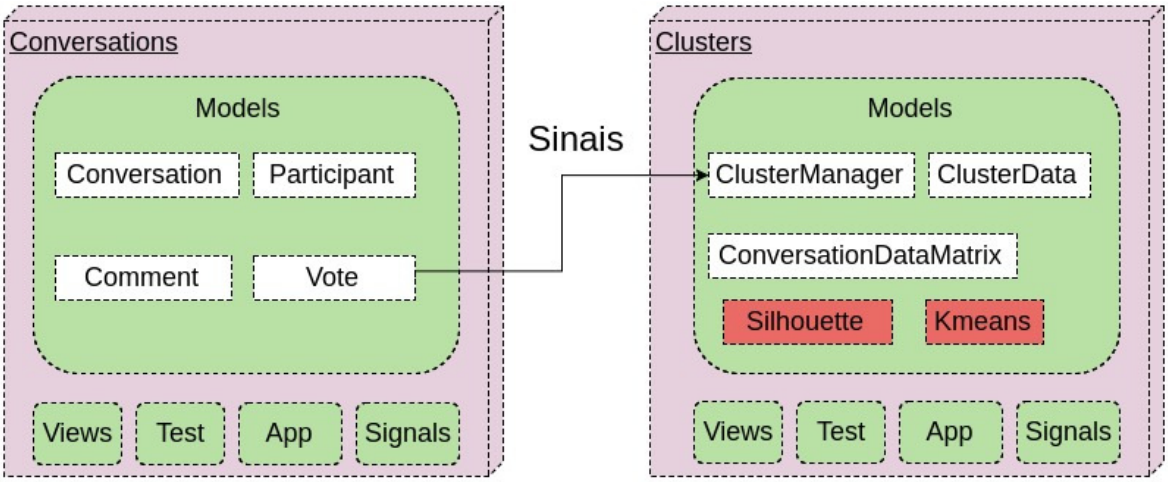

Figura 13 – Estrutura dos *apps* Conversations e Clusters.

5.2 Validação do modelo

Para a validação dos resultados foi tomada a decisão de produzir manualmente usuários, comentários e votos cujos valores serão utilizados como referência. Estes dados são submetidos ao algoritmo e uma análise da taxa de acerto é feita segundo o índice de Rand ajustado, apresentado na Seção 2.6.3

5.2.1 Definição dos casos de teste

Para uma análise mais abrangente, definimos diferentes cenários para a validação, variando a quantidade de grupos esperados, quantidade de participantes e comentários na conversa. Definimos em cada cenário a quantidade de grupos e suas características, de modo a manipular os votos para gerar grupos da maneira desejada, possibilitando, assim, uma comparação entre a classificação esperada e a obtida pelo algoritmo. A Tabela 9 mostra os cenários considerados nos testes. Buscamos cobrir diferentes intervalos de teste para obter uma melhor análise do modelo.

Tabela 9 – Cenários de teste do algoritmo.

Cenário	N grupos/clusters	N comentários	N membros/grupo	Total participantes
1	2	8	4	8
2	2	50	5	10
3	2	50	10	20
4	2	150	10	20
5	3	8	4	8
6	3	50	5	15
7	3	50	10	30
8	3	150	10	30
9	4	50	5	20
10	4	50	10	40
11	4	150	10	40
12	5	50	5	25
13	5	50	10	50
14	5	150	10	50

5.2.2 Obtenção dos votos dos usuários

Para gerar os votos em cada cenário, utilizamos algumas formulações estatísticas. Para entender este processo, é preciso esclarecer alguns conceitos. Considerando que para as três possibilidades de votos: discordo, pular, concordo ou (-1, 0, 1), a chance de um usuário aleatório hipotético escolher qualquer uma delas é 1/3, logo a soma total das probabilidades é igual a 1. Isso caracteriza um vetor de probabilidades discretas com três variáveis independentes, o qual podemos calcular as probabilidades para qualquer combinação de resultados, como a chance de tirar o número 6 ao rolar dois dados. Podemos representar essas probabilidades como um vetor (θ):

$$\theta = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$$

Se queremos manipular essas probabilidades, causando um viés para, por exemplo, indicar uma maior chance de um usuário concordar com um comentário, podemos dar um peso maior para a probabilidade correspondente a essa variável:

$$\theta = \left(\frac{0.9}{3}, \frac{1}{3}, \frac{1.1}{3}\right)$$

Independente do peso dado, se a soma total das probabilidades ainda for igual a 1 e nenhuma delas for negativa, caracterizamos um vetor de probabilidades modelado sobre uma distribuição de Dirichlet, que garante que seus valores são normalizados e positivos.

Então, para cada grupo serão produzidas probabilidades de votos para os N comentários do cenário, seguindo essa distribuição, que indicam as chances de um usuário daquele grupo discordar, pular ou concordar com um dado comentário. Desta forma, cada

grupo possui um conjunto de probabilidades diferentes, que determina a maneira como se comportam/votam os integrantes daquele grupo em cada comentário. A tabela 10 mostra um conjunto de probabilidades para 5 comentários que pode representar o comportamento de voto de um determinado grupo de participantes.

Tabela 10 – Probabilidades de votos.

comentário	θ**(-1)**	θ**(0)**	θ**(1)**
1	0.001	0.256	0.743
2	0.825	0.174	0.
3	0.395	0.042	0.564
4	0.615	0.365	0.02
5	0.007	0.09	0.904

Tabela 11 – Probabilidades acumuladas.

Comentário	θ**(-1)**	θ**(0)**	θ**(1)**
1	0.001	0.257	1
2	0.825	0.999	1
3	0.395	0.437	1
4	0.615	0.980	1
5	0.007	0.097	1

Depois de calculadas as probabilidades dos N comentários para um grupo, precisamos gerar K usuários e obter os seus votos para cada comentário. Então, para um dado usuário u membro do grupo G, o qual se deseja obter os votos, geramos aleatoriamente um valor x_i entre 0 e 1 seguindo uma probabilidade uniforme, relacionada a cada comentário i, tal que $i \in N$. Posteriormente calculamos em qual faixa de valor ela se encaixa na tabela de probabilidades acumuladas do grupo G, e isso determina qual será o voto do usuário.

Desta forma, para calcular os votos de um usuário V_u, pertencente ao grupo G, temos:

N = número de comentários do cenário de teste

i = índice do comentário, tal que $i \in N$

$G(i)$ = probabilidades de voto acumuladas do grupo G para cada comentário i

x_i = faixa em que se situa o voto no i-ésimo comentário

$V_u(i)$ = Voto do usuário u no comentário i, sendo $V_u(i) = [-1, 0, 1]$

Então, iniciamos $V_u(i) = 1$

se $x_i < \theta(0)$, definimos $V_u(i) = 0$

se $x_i < \theta(-1)$, definimos $V_u(i) = -1$

Tabela 12 – Votos V_u de um usuário u, gerados para cada comentário i, segundo as probabilidades θ de um grupo G.

Comentário(i)	θ(-1)	θ(0)	θ(1)	x_i	$V_u(i)$
1	0.001	0.257	1	0.213	0
2	0.825	0.999	1	0.146	-1
3	0.395	0.437	1	0.851	1
4	0.615	0.980	1	0.434	-1
5	0.007	0.097	1	0.111	1

Assim, são gerados os N votos para os K usuários dos grupos determinados no cenário de teste. Como cada grupo possui probabilidades de votos diferentes, espera-se que o algoritmo identifique e classifique cada usuário em seu respectivo grupo.

Utilizamos uma função da biblioteca Numpy para obter o vetor de probabilidades pseudo aleatórias segundo a distribuição de Dirichlet:

```
>>> import numpy as np
>>> np.random.dirichlet(alpha, size=None)
```

O parâmetro *alpha* é um vetor k-dimensional, composto por números reais, e cada dimensão desse vetor corresponde ao peso de uma variável dentro da distribuição. Já o parâmetro *size* indica o número de exemplos a serem gerados e simboliza a quantidade de comentários.

Para produzir as probabilidades de cada opção de voto, Discordar, Pular, Concordar, para 5 comentários, podemos fornecer três valores em *alpha* e configurar *size=5*. Cada linha da saída corresponde a um comentário e cada valor dentro de um vetor representa às chances de um usuário discordar, pular e concordar com o dado comentário.

```
>>> numpy.random.dirichlet([0.8, 0.8, 0.8], 5)
array([[ 0.15770032, 0.5041667 , 0.33813298],
       [ 0.50116975, 0.4946626 , 0.00416766],
       [ 0.34957988, 0.05330692, 0.5971132 ],
       [ 0.48945452, 0.24721013, 0.26333535],
       [ 0.12949316, 0.06325824, 0.8072486 ]])
```

Vale ressaltar que a escolha dos valores de *alpha* impactam nas probabilidades geradas. Quanto mais próximo de 0, os valores gerados tendem a ter uma discrepância maior entre si, ou seja, cada variável tende a possuir probabilidades mais distantes das outras. Para valores altos de *alpha*, acima de 10, a tendência é que as probabilidades fiquem mais igualitárias, tendendo a valores próximos de [0.33, 0.33, 0.33]. Os números de *alpha* utilizados nos testes foram [0.2, 0.4, 0.6, 0.8, 1.0], um para cada possível população. Se o cenário define 2 grupos, utilizamos [0.2, 0.2, 0.2] e [0.4, 0.4, 0.4] para gerar as probabilidades de cada grupo, se são 3 grupos no cenário, repetimos [0.2, 0.2, 0.2] e [0.4, 0.4, 0.4] e acrescentamos [0.6, 0.6, 0.6], e assim sucessivamente.

Os valores x_i usados para obter o voto dos usuários nos comentários também são dados com uma função da biblioteca Numpy, que recebe como parâmetro a quantidade

de exemplos a serem gerados, que para nós significa o número de comentários no cenário:

```
>>> numpy.random.uniform(size=5)
array([ 0.4467375, 0.9460769, 0.4989846, 0.7278234, 0.4280037 ])
```

Para manter a replicabilidade e coerência dos testes, é necessário utilizar sempre as mesmas probabilidades para cada cenário. Foram utilizados os mesmos valores de *alpha* e também o contador utilizado pela linguagem para gerar números pseudo aleatórios é sempre iniciado com o mesmo valor antes da execução dos testes de cada cenário, conforme o código a seguir.

```
>>> numpy.random.seed(57)
```

5.2.3 Exemplo de cenário de teste

Esclarecendo um pouco mais os passos para produzir os valores de referência utilizados para validação do algoritmo, tomemos como exemplo o **cenário de teste 1**, da Tabela 9, para o qual foram definidos os seguintes parâmetros:

- Número de grupos esperados = 2
- Número de comentários na conversa = 8
- Número de membros em cada grupo = 4
- Total de participantes = 8

Inicialmente, obtemos as probabilidades de voto para os dois grupos da conversa, segundo a distribuição de Dirichlet, vistos nas Tabelas 13 e 14, e calculamos também os valores acumulados.

Então, geramos as probabilidades de voto de cada usuário e calculamos em qual faixa de valor elas se encaixam na tabela de probabilidades acumuladas do seu grupo, o que resulta no valor dos votos para cada comentário, conforme as Tabelas 15 e 16.

Tabela 13 – Probabilidades de votos grupo(0), $\alpha = 0.2$

Comentário	θ**(-1)**	θ**(0)**	θ**(1)**
1	0	0.168	0.832
2	0.898	0.102	0
3	0.365	0.013	0.622
4	0.990	0.010	0
5	0	0	1
6	0.132	0.868	0
7	0.389	0.611	0
8	0.103	0.409	0.488
9	0.059	0.564	0.377
10	0.993	0.007	0

Tabela 14 – Probabilidades de votos grupo(1), $\alpha = 0.4$

Comentário	θ**(-1)**	θ**(0)**	θ**(1)**
1	0.367	0.151	0.483
2	0.499	0.008	0.493
3	0.040	0.794	0.166
4	0.004	0.814	0.182
5	0.850	0.008	0.143
6	0.580	0.069	0.352
7	0.255	0.055	0.690
8	0.017	0.972	0.011
9	0.798	0.123	0.079
10	0.181	0.140	0.678

Tabela 15 – Votos (V) de 4 usuários do grupo 0.

Comentário	θ(-1)	θ(0)	θ(1)	$V_u(1)$	$V_u(2)$	$V_u(3)$	$V_u(4)$
1	0	0.168	1	1	1	1	1
2	0.898	1	1	-1	-1	-1	-1
3	0.365	0.378	1	1	-1	1	1
4	0.990	1	1	-1	-1	-1	-1
5	0	0	1	1	1	1	1
6	0.132	1	1	0	0	0	0
7	0.389	1	1	0	0	0	-1
8	0.103	0.512	1	0	0	0	0
9	0.103	0.512	1	0	0	1	0
10	0.103	0.512	1	-1	-1	-1	-1

Tabela 16 – Votos (V) de 4 usuários do grupo 1.

Comentário	θ(-1)	θ(0)	θ(1)	$V_u(5)$	$V_u(6)$	$V_u(7)$	$V_u(8)$
1	0.367	0.518	1	-1	-1	1	-1
2	0.499	0.507	1	1	-1	-1	1
3	0.040	0.834	1	0	1	0	0
4	0.004	0.818	1	1	0	0	0
5	0.850	0.858	1	-1	-1	-1	-1
6	0.580	0.649	1	-1	1	-1	1
7	0.255	0.310	1	-1	1	-1	1
8	0.017	0.989	1	0	0	0	0
9	0.798	0.921	1	-1	-1	-1	-1
10	0.181	0.321	1	-1	1	0	0

De acordo com as características desse cenário, esperamos então que o algoritmo classifique os 8 usuários em 2 grupos diferentes, o grupo 0 e o grupo 1. Para verificar a taxa de acerto do modelo, o teste escrito para este cenário cria uma conversa com 10

comentários, insere os votos gerados para os usuários e executa a clusterização com este conjunto de votos, calculando ao final, o índice de Rand ajustado entre os resultados esperados e obtidos.

Resultado esperado: [0, 0, 0, 0, 1, 1, 1, 1]

Resultado obtido: [1, 1, 1, 1, 0, 2, 0, 2]

Índice de Rand Ajustado: 0.7

Vejamos, neste exemplo, que o algoritmo classificou os participantes em 3 clusters. O cluster 0 esperado corresponde ao cluster de número 1 nos resultados obtidos e o cluster 1 foi dividido nos clusters 0 e 2. O valor 0.7 do índice indica que, para cada par de objetos selecionado, existe uma probabilidade de 70% deles estarem agrupados igualmente nas duas classificações. O número de clusters ou a permutação dos rótulos(número do grupo) não afeta no cálculo do índice. Na próxima seção apresentamos e discutimos os resultados para cada um dos cenários de teste propostos.

5.3 Resultados e discussões

Buscando uma maior confiança na operação do sistema, desenvolvemos testes automatizados para as classes e métodos implementados. Para isso, utilizamos a biblioteca de testes Pytest. Estes testes auxiliaram tanto no desenvolvimento do algoritmo quanto na avaliação da acurácia do modelo após a sua implementação. Para cada um dos cenários definidos na Tabela 9, escrevemos um teste para medir o índice de Rand ajustado entre a saída esperada e a saída obtida, capturando os resultados dos testes de acurácia para todos os 14 cenários definidos. Esses resultados são vistos na Tabela 17.

Tabela 17 – Dados de cada cenário e acurácia obtida.

Cenário	N comentários	N membros/grupo	Total participantes	N grupos esperados	Grupos gerados	Acurácia
1	8	4	8	2	3	0.69
2	50	5	10	2	2	1.0
3	50	10	20	2	2	1.0
4	150	10	20	2	2	1.0
5	8	4	8	3	4	0.87
6	50	5	15	3	3	1.0
7	50	10	30	3	3	1.0
8	150	10	30	3	3	1.0
9	50	5	20	4	4	1.0
10	50	10	40	4	3	0.69
11	150	10	40	4	4	1.0
12	50	5	25	5	4	0.75
13	50	10	50	5	3	0.51
14	150	10	50	5	4	0.76
Média						0.87

Na média, o modelo obteve uma acurácia de 87%. Para entendermos os pontos onde o resultado obtido foi diferente do esperado, mostramos os dados em gráficos para

compreender de forma visual cada cenário, fazendo um contraste entre os dados classificados manualmente com as probabilidades da distribuição de Dirichlet e o resultado gerado pelo algoritmo.

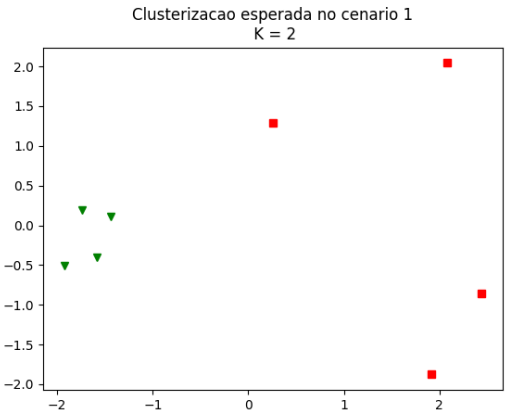

Figura 14 – Cenário 1 esperado.

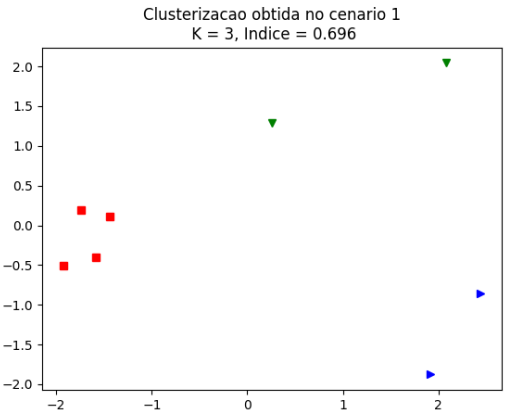

Figura 15 – Cenário 1 obtido.

O cenário 1 acima, era o mais simples, com menor quantidade de comentários, grupos e participantes. A Figura 14, que representa a classificação feita manualmente com o método das probabilidades, mostra que acabamos obtendo alguns valores de voto muitos distantes para membros do mesmo grupo, como podemos notar nos pontos em vermelho bastante dispersos. Com isso o algoritmo identificou-os como sendo dois grupos distintos e a acurácia foi de 69%.

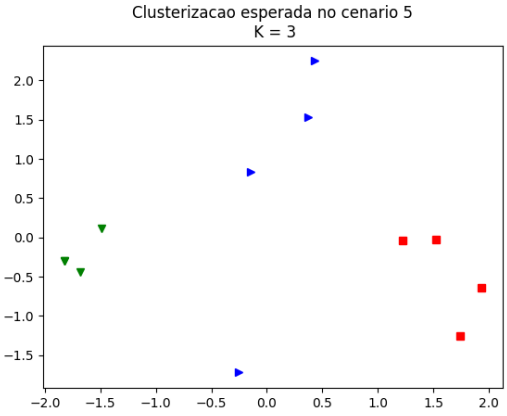

Figura 16 – Cenário 5 esperado.

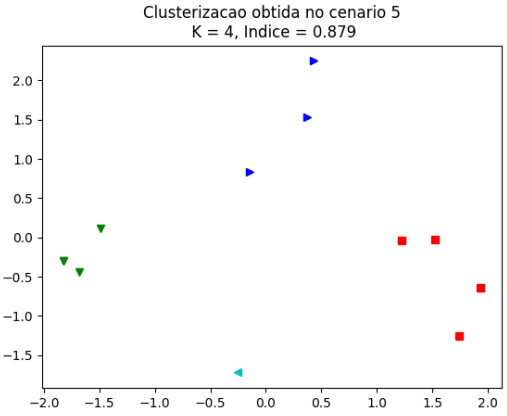

Figura 17 – Cenário 5 obtido.

O cenário 5, teve os mesmos efeitos do primeiro, mas com apenas 1 ponto fora do esperado. Aqui, o algoritmo identificou um cluster com um membro isolado, o que é perfeitamente plausível. Foi calculado nesse cenário uma acurácia foi de 87%

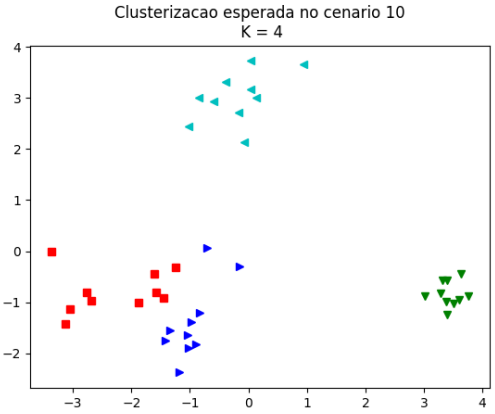

Figura 18 – Cenário 10 esperado.

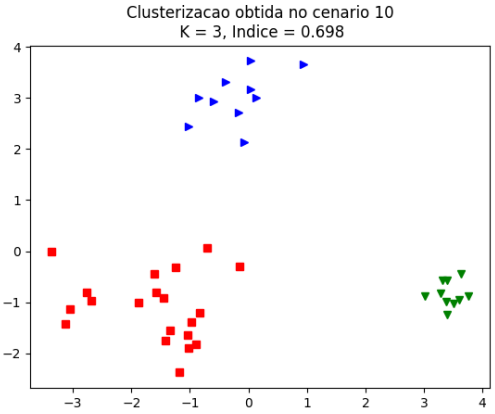

Figura 19 – Cenário 10 obtido.

O cenário 10, teve um efeito contrário aos cenários 1 e 5. Nesse caso de teste, as probabilidades geradas para dois grupos foram muito próximas uma das outras (pontos vermelhos e azuis) e o algoritmo as identificou como sendo um grupo só, o que resultou em uma acurácia de 69%.

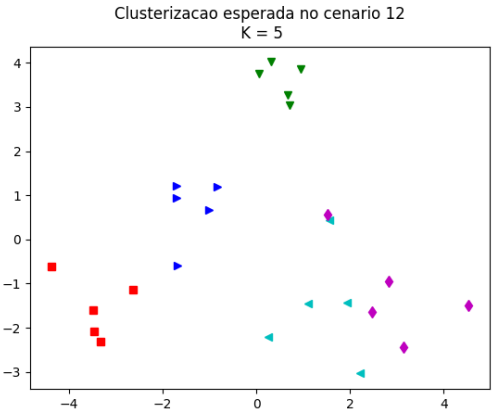

Figura 20 – Cenário 12 esperado.

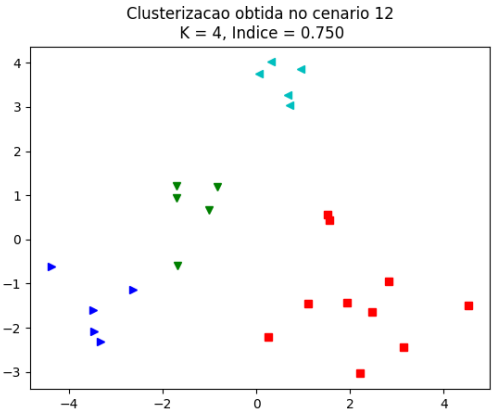

Figura 21 – Cenário 12 obtido.

No cenário 12, a proximidade entre as probabilidades de dois grupos também foi muito alta, havendo até uma superposição de pontos. O algoritmo também acabou juntando dois clusters em um, como no cenário 10. Nesse cenário, a acurácia foi de 75%. A medida que cresce o número de comentários, participantes e grupos esperados das conversas, fica mais difícil identificar todos os clusters corretamente, pois a taxa de probabilidades parecidas e até repetidas entre os grupos aumenta.

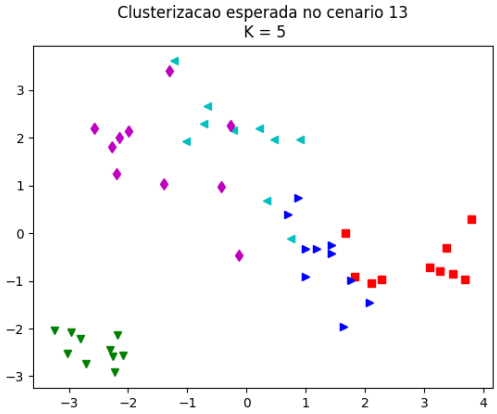

Figura 22 – Cenário 13 esperado.

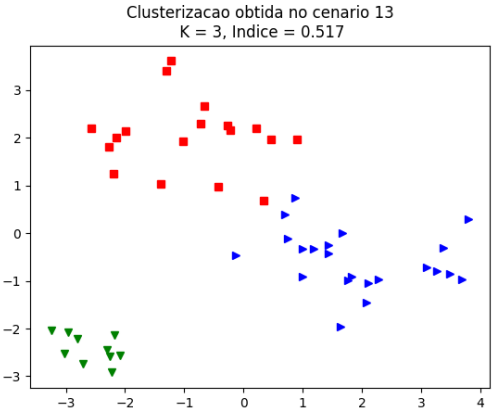

Figura 23 – Cenário 13 obtido.

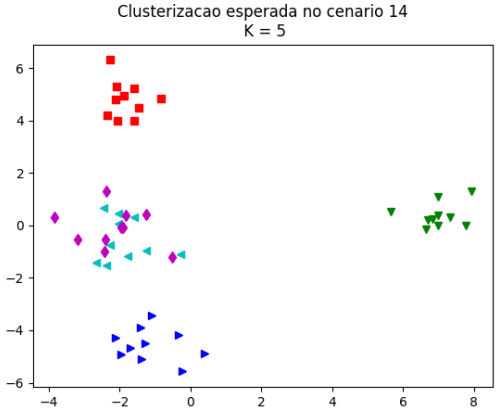

Figura 24 – Cenário 14 esperado.

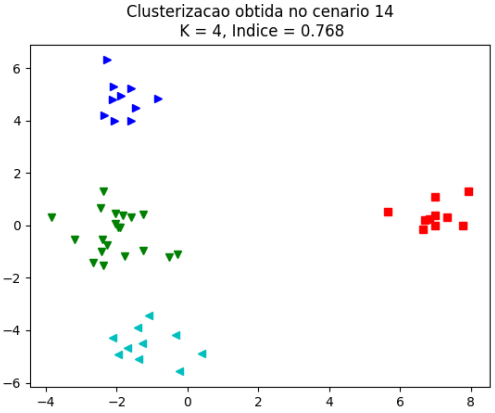

Figura 25 – Cenário 14 obtido.

Nos testes para os cenários 13 e 14, a superposição foi mais acentuada, o que reforça a necessidade de ajuste nos valores das probabilidades geradas. Aqui, a acurácia caiu abruptadamente para 51% e 76%, respectivamente.

Podemos ver que os resultados gerados pelo algoritmo são mais coerentes que os gerados através das probabilidades. Os resultados dos cenários 12, 13 e 14 mostram que esses foram os cenários mais afetados pelas probabilidades similares geradas, com algumas superposições entre os grupos, levando o algoritmo a identificar esses pontos como sendo do mesmo cluster, o que faz total sentido.

Apesar dos erros, o método de validação através das probabilidades demonstrou ser uma boa saída para formulação da base de dados de teste caso seja implementada uma pequena melhoria que evitaria a geração de probabilidades parecidas. Podemos estabelecer um limiar mínimo para a diferença entre a distância probabilística entre os valores gerados para cada grupo, que pode ser calculado como um somatório da diferença entre as probabilidades.

$$\frac{1}{2}\sum(P_i - Q_i)$$

Todavia, analisando as imagens da clusterização feita pelo algoritmo já é possível perceber que o modelo possui um nível de coerência maior do que o refletido nos números, separando bem os clusters.

6 Conclusão

Com o estudo e desenvolvimento desse trabalho foi possível compreender melhor o universo por trás da vasta área de inteligência artificial e aprendizado de máquina. Pudemos notar o quão extenso é esse tópico, com vários modelos, técnicas e algoritmos no contexto de clusterização, embora só tenhamos explorado uma mínima parte, com o algoritmo k-means, do modelo centroide.

Consideramos então que os resultados obtidos nesse trabalho foram bastante proveitosos e satisfatórios. Pudemos aplicar um método prático para encontrar de maneira sistemática, quais são os grupos de usuários que possuem opiniões de voto parecidos em uma conversa, utilizando um algoritmo bastante conhecido e de fácil entendimento.

Reconhecemos também a necessidade de testar outros algoritmos e técnicas, tanto de clusterização em si, quanto de redução de dimensionalidade. O modelo desenvolvido possui ainda algumas limitações. A técnica PCA usada para redução dos dados em duas dimensões, por exemplo, não garante que os dois componentes selecionados tenham uma variância significativa dos dados para identificação dos reais grupos de opinião. Existem várias outras técnicas para redução de dimensionalidade que podem talvez, oferecer resultados melhores e devem ser testadas e avaliadas como solução.

O método de validação utilizado demonstrou ser bastante útil na manipulação manual dos votos, apesar dos ajustes necessários para aperfeiçoar os valores das probabilidades de cada grupo. Mas ainda é fato que, mesmo com os índices de acurácia gerados, não é possível medir uma qualidade absoluta do modelo. Nós propomos então que os resultados calculados sejam utilizados como base para futuras evoluções do modelo e testes com outros algoritmos, pois podem servir como um parâmetro a mais de análise. De maneira geral, entendemos que avaliar a qualidade de um modelo requer uma observação por diferentes perspectivas, que utilize índices numéricos, visualizações gráficas e outros métodos convenientes.

E para experimentos futuros, uma comparação desejável seria submeter os mesmos casos de testes para análise com a ferramenta Polis, afim de se obter um raio-x entre os dois sistemas. Esse teste pode ajudar a compreender melhor o comportamento da modelagem feita pelas duas ferramentas e aprofundar mais nos métodos de clusterização aplicados por ambas.

Um outro ponto importante que vemos de maneira positiva nesse trabalho foi que conseguimos desenvolver um modelo inicial em uma aplicação capaz de manipular os dados das conversas, comentários e votos como um primeiro passo para o desenvolvimento de um novo serviço de clusterização, que está disponível para uso da comunidade do

projeto Empurrando Juntos. Todo o código desenvolvido está aberto e acessível sob licença GPL.V3[1], hospedado em <gitlab.com/pentano/server>, para livre uso.

Esse trabalho conta ainda a colaboração dos colegas de curso Ítalo Paiva Batista e Emily Trindade de Morais, que estão dando continuidade a essa pesquisa no Trabalho de Conclusão de Curso da dupla. Eles irão dar seguimento à evolução da aplicação que aqui iniciamos, com o desenvolvimento de novas funcionalidades, integração via API e outros recursos, objetivando prover uma primeira versão da ferramenta.

[1] <gnu.org/licenses/gpl-3.0.en.html>

Referências

AMATRIAIN, X. et al. Data mining methods for recommender systems. In: _____. *Recommender Systems Handbook*. Boston, MA: Springer US, 2011. cap. 2, p. 39–67. ISBN 978-0-387-85820-3. Disponível em: <http://dx.doi.org/10.1007/978-0-387-85820-3_1>. Citado 2 vezes nas páginas 22 e 24.

BENKLER, Y. *The wealth of networks : how social production transforms markets and freedom*. [S.l.]: Yale University Press, 2006. ISBN 0-300-11056-1,978-0-300-11056-2,9780300127232,0300127235,9781281740809,1281740802. Citado na página 19.

GAN, G.; MA, C.; WU, J. *Data Clustering: Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability)*. [S.l.]: SIAM, Society for Industrial and Applied Mathematics, 2007. (ASA-SIAM Series on Statistics and Applied Probability). Citado 8 vezes nas páginas 22, 23, 25, 26, 27, 60, 62 e 63.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. Cluster validity methods: part i. *ACM Sigmod Record*, ACM, v. 31, n. 2, p. 40–45, 2002. Citado na página 28.

HUBERT, L.; ARABIE, P. Comparing partitions. *Journal of classification*, Springer, v. 2, n. 1, p. 193–218, 1985. Citado na página 31.

LéVY, P. *Ciberdemocracia*. Instituto Piaget, 2002. (Epistemologia e Sociedade). ISBN 9789727716722. Disponível em: <https://books.google.com.br/books?id=zNc-AQAACAAJ>. Citado na página 19.

RAND, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, Taylor & Francis Group, v. 66, n. 336, p. 846–850, 1971. Citado na página 30.

SCIKIT-LEARN DEVELOPERS, *scikit-learn user guide*. 2016. Disponível em: <scikit-learn.org/stable/modules/clustering.html>. Citado na página 27.

SEGARAN, T. *Programing Collective Intelligence*. [S.l.]: O'Reilly Media, 2007. Citado na página 25.

SHIRKHORSHIDI, A. S.; AGHABOZORGI, S.; WAH, T. Y. A comparison study on similarity and dissimilarity measures in clustering continuous data. *PLoS ONE*, v. 10, n. 9, p. 1 – 20, 2015. ISSN 19326203. Disponível em: <http://search-ebscohost-com.ez54.periodicos.capes.gov.br/login.aspx?direct=true&db=aph&AN=111544981&lang=pt-br&site=ehost-live>. Citado 2 vezes nas páginas 60 e 63.

WALDE, S. S. im. Experiments on the automatic induction of german semantic verb classes. *Comput. Linguist.*, MIT Press, Cambridge, MA, USA, v. 32, n. 2, p. 159–194, jun. 2006. ISSN 0891-2017. Disponível em: <http://dx.doi.org/10.1162/coli.2006.32.2.159>. Citado 3 vezes nas páginas 29, 31 e 32.

Apêndices

APÊNDICE A – Medidas de distância

Neste apêndice apresentamos algumas outras medidas de similaridade para dados numéricos, categóricos e binários. Gan, Ma e Wu (2007) cita ainda outras medidas que foram desenvolvidas para outros contextos de dados, como atributos temporais, imagens, sons, contextos mistos, e mais.

A.1 Medidas para dados numéricos

- Distância Média: Esta é uma versão modificada da distância Euclidiana. Esta função acaba sendo mais intuitiva, pois distribui o impacto dos atributos dividindo o resultado pela quantidade total destes, gerando uma média. Ela minimiza possíveis disparidade dos valores dos atributos em conjuntos de dados onde há valores ausentes para alguns atributos ou com grandes diferenças de escala entre eles.

$$\mathrm{d}(x,y) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(x_k - y_k)^2}$$

- Distância Euclidiana com pesos: Se existe alguma ponderação diferente para os atributos, esta medida pode ser usada. Seu cálculo aplica a relevância individual de cada propriedade do objeto.

$$\mathrm{d}(x,y) = \sqrt{\sum_{k=1}^{N} w_k(x_k - y_k)^2}$$

- Similaridade do Cosseno: Esta medida é comumente utilizada para calcular distâncias entre vetores de documentos de texto (SHIRKHORSHIDI; AGHABOZORGI; WAH, 2015). Nesta definição, $|| x ||_2$ e $|| y ||_2$ representam as normas dos vetores $x = (x_1, x_2, ..., x_n)$ e $x = (y_1, y_2, ..., y_n)$, dadas por: $|| x ||_2 = \sqrt{x_1^2 + x_2^2 + x_n^2}$.

$$\mathrm{d}_{\cos}(x,y) = \frac{\sum_{k=1}^{N} x_k y_k}{|| x ||_2 || y ||_2}$$

A.2 Medidas para dados categóricos

Atributos categóricos são também comumente chamados de atributos nominais, os quais simplesmente utilizam nomes, como a cor de um objeto ou a marca de um carro. Esse tipo de informação é analisada de forma diferente de dados numéricos contínuos, já que sua natureza não é quantificável.

- Distância de correspondência simples: Esta é a distância mais simples e conhecida para medida de atributos categóricos, onde é simplesmente feita uma contagem da quantidade de itens iguais e diferentes entre os objetos. A distância de dois objetos x e y para um atributo k, representada por δ, é dada por:

$$\delta(x_k, y_k) = \begin{cases} 0 \text{ se } x_k = y_k, \\ 1 \text{ se } x_k \neq y_k \end{cases}$$

Assim, seja um vetor d-dimensional de atributos para os objetos x e y, a distância de simples correspondência é calculada pela soma dos δ de cada um deles, e quanto maior o valor, mais distantes são os objetos:

$$\mathrm{d}(x, y) = \sum_{k=1}^{d} \delta(x_k, y_k)$$

No entanto, podemos normalizar o resultado para um intervalo entre [0 - 1) da seguinte forma:

$$\mathrm{d}(x, y) = 1 - \frac{1}{1 + \sum_{k=1}^{d} \delta(x_k, y_k)}$$

Note que o resultado será igual a zero para objetos idênticos, onde $\delta = 0$, mas nunca será igual a 1 para objetos totalmente distintos.

- Outras medidas de correspondência: Além da disância de correspondência simples, existem outros possíveis coeficientes para dados categóricos. Vejamos então, três definições prévias para o seu entendimento:

$N_{a+d} \Rightarrow$ Soma total do número de atributos correspondentes entre dois objetos.

$$N_{a+d} = \sum_{k=1}^{d} [1 - \delta(x_k, y_k)]$$

$N_d \Rightarrow$ Soma total do número de atributos indefinidos. Algumas medidas contabilizam a falta de informação no cálculo da distância, e o valor que representa este dado é de escolha da aplicação, podendo ser 0, “indefinido”, -1, “nulo”, etc.

$$N_d = \sum_{k=1}^{d} [\delta(x_k, nulo) + \delta(y_k, nulo) - \delta(x_k, nulo) \times \delta(y_k, nulo)]$$

$N_{b+c} \Rightarrow$ Soma total do número de atributos não correspondentes entre dois objetos.

$$N_{b+c} = \sum_{k=1}^{d} \delta(x_k, y_k)$$

Dadas as definições, podemos então entender algumas das medidas de similaridade, sumarizadas na Tabela 18:

Medida	s(x,y)	Peso das correspondências e não correspondências
Russel and Rao	$\frac{N_{a+d}-N_d}{N_{a+d}+N_{b+c}}$	Pesos iguais
Simple matching	$\frac{N_{a+d}}{N_{a+d}+N_{b+c}}$	Pesos iguais
Jaccard	$\frac{N_{a+d}-N_d}{N_{a+d}-N_d+N_{b+c}}$	Pesos iguais
Dice	$\frac{2N_{a+d}-2N_d}{2N_{a+d}-2N_d+N_{b+c}}$	Dobro do peso para pares correspondentes
Rogers-Tanimoto	$\frac{N_{a+d}}{N_{a+d}+2N_{b+c}}$	Dobro do peso para pares não correspondentes
Kulczynski	$\frac{N_{a+d}-N_d}{N_{b+c}}$	Pares correspondentes excluídos do denominador

Tabela 18 – Algumas medidas de similaridade para dados nominais.
Fonte: Gan, Ma e Wu (2007), adaptado

A.3 Medidas para dados binários

Existem também medidas específicas para o cálculo da distância para contextos onde os objetos são representados por características binárias, ou seja, só são possíveis dois valores, como "verdadeiro" e "falso". Essa categoria de atributos pode ainda ser dividida em duas classes, binários simétricos e assimétricos. Valores simétricos representam dados com peso igualmente importantes, como "masculino" e "feminino". Já a classe de valores assimétricos tratam valores onde um possui mais relevância que o outro, como "sim" para a presença de uma determinada condição e "não" para sua ausência. Esses atributos são uma especialidade de dados categóricos e podem até ser calculados com as mesmas medidas. Mas também, possuem suas medidas específicas que foram propostas para seu contexto.

Assim, sejam x e y dois objetos representados por um vetor em um espaço d-dimensional, e sejam A, B, C, D definidos como a soma S da quantidade de correspondências dos valores em x e em y para as combinações possíveis em cada atributo de um objeto, e α como sendo uma relação dessas correspondências:

$$A = S_{11}(x, y), \text{soma dos elementos tal que } x_i = 1 \text{ e } y_i = 1$$

$$B = S_{10}(x, y), \text{soma dos elementos tal que } x_i = 1 \text{ e } y_i = 0$$

$$C = S_{01}(x, y), \text{soma dos elementos tal que } x_i = 0 \text{ e } y_i = 1$$

$$D = S_{00}(x, y), \text{soma dos elementos tal que } x_i = 0 \text{ e } y_i = 0$$

$$\alpha = \sqrt{(A+B)(A+C)(B+D)(C+D)}$$

Então, sobre essas definições podemos observar alguma dessas medidas para esse tipo de dado, vide a Tabela 19.

Medida	**s(x,y)**	**Limite de s(x,y)**
Jaccard	$\frac{A}{A+B+C}$	$[0, 1]$
Dice	$\frac{A}{2A+B+C}$	$[0, \frac{1}{2}]$
Pearson	$\frac{AD-BC}{\sigma}$	$[-1, 1]$
Yule	$\frac{AD-BC}{AD+BC}$	$[-1, 1]$
Russell-Rao	$\frac{A}{d}$	$[0, 1]$
Sokal-Michener	$\frac{A+D}{d}$	$[0, 1]$
Rogers-Tanimoto	$\frac{A+D}{A+2(B+C)+D}$	$[0, 1]$
Rogers-Tanimoto-a	$\frac{A+D}{A+2(B+C)+D}$	$[0, 1]$
Kulzinsky	$\frac{A}{B+C}$	$[0, \infty]$

Tabela 19 – Algumas medidas de similaridade para dados binários.
Fonte: Gan, Ma e Wu (2007), adaptado

Muitas medidas de similaridade são descritas na literatura, o que causa uma certa dúvida e confusão na hora de escolher a solução apropriada. A avaliação das medidas é um tanto subjetiva e há várias maneiras de comparar os resultados de cada uma. Shirkhorshidi, Aghabozorgi e Wah (2015) apresentam um interessante estudo de comparação entre diversas funções em diferentes conjuntos de dados com atributos numéricos contínuos. Nesse estudo são utilizados dados de baixa e de alta dimensionalidade advindos dos mais variados contextos. A conclusão, no entanto, revela que não há uma medida que resolva todos os casos, cada contexto exige uma análise particular para um resultado aprimorado.

APÊNDICE B – Técnicas

B.1 Análise de Componentes Principais - PCA

Esta é uma técnica é comumente utilizada para transformação dos dados com o objetivo de reduzir sua dimensão, quando há um grande número de variáveis nos dados, o que pode causar o problema da maldição da dimensionalidade. Elucidamos a seguir os passos para aplicação do PCA para reduzir um conjunto de dados de duas para uma dimensão. A Tabela 20 abaixo mostra 20 objetos relacionados para as variáveis X1 e X2.

Objetos	X1	X2	Objetos	X1	X2
1	0.3	0.5	11	0.6	0.8
2	0.4	0.3	12	0.4	0.6
3	0.7	0.4	13	0.3	0.4
4	0.5	0.7	14	0.6	0.5
5	0.3	0.2	15	0.8	0.5
6	0.9	0.8	16	0.8	0.9
7	0.1	0.2	17	0.2	0.3
8	0.2	0.5	18	0.7	0.7
9	0.6	0.9	19	0.5	0.5
10	0.2	0.2	20	0.6	0.4

Tabela 20 – Conjunto de dados

Então, como primeiro passo para a análise dos componentes principais, centralizamos os dados em torno dos eixos subtraindo a média de cada variável no conjunto de pontos, como mostra a figura 27.

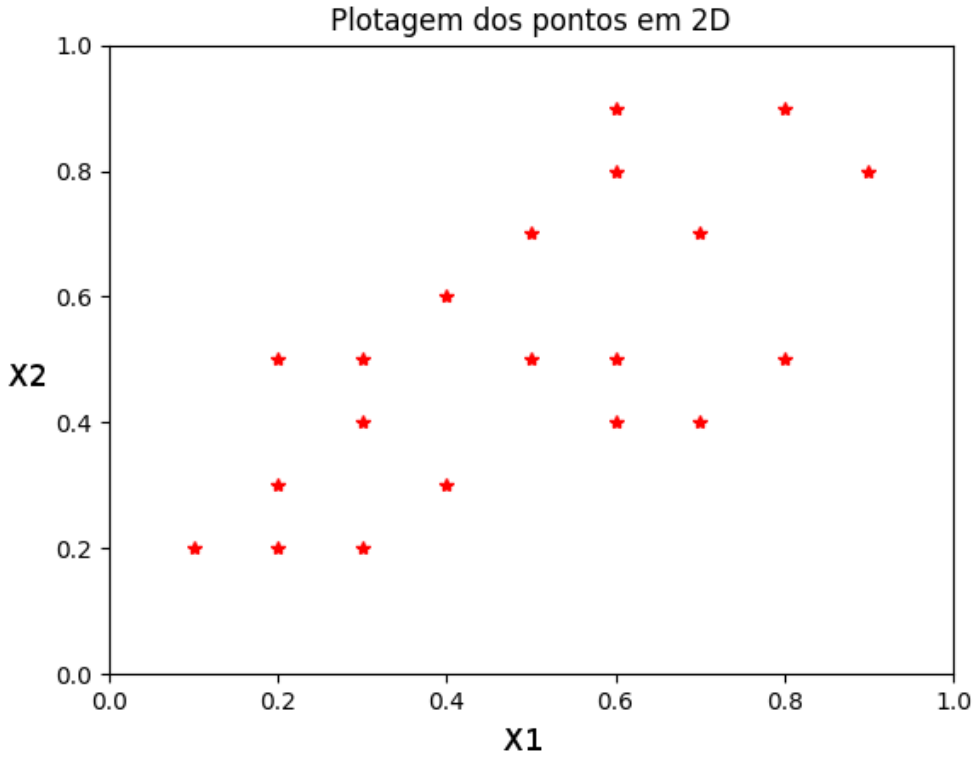

Figura 26 – Pontos plotados

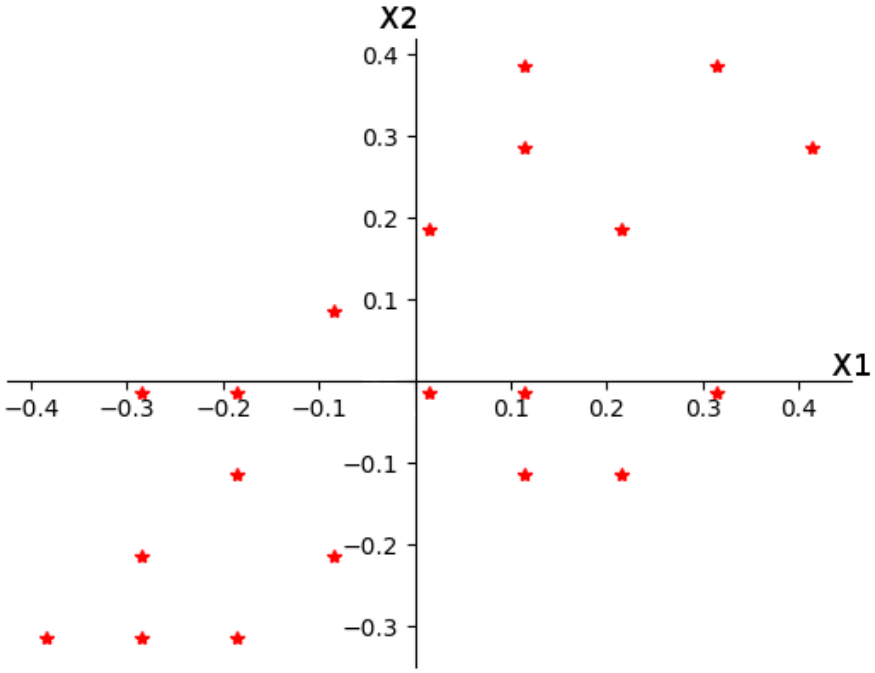

Figura 27 – Pontos centralizados

Após a translação do eixo, calculamos a matriz de covariância entre as variáveis, o quenos dá informações sobre a relação que elas possuem. Se a covariância é positiva, então as duas variáveis crescem e decrescem juntas. Se seu valor é negativo, então enquanto uma variável cresce, a outra decresce e vice-versa. Esses valores determinam a dependência linear entre as variáveis que serão usadas para redução da dimensão dos dados, como mostra a tabela 21.

-	**X1**	**X2**
X1	0.33	0.25
X2	0.25	0.41

Tabela 21 – Matriz de covariância entre as variáveis

Os valores da diagonal principal representam a covariância de cada variável com ela mesma, o que é igual a sua medida de variância. A variância mede o quão dispersos os dados estão em relação a média. A diagonal secundária mostra a covariância entre as duas variáveis. Os valores positivos mostram que as duas crescem e decrescem juntas.

Podemos então encontrar os autovetores e autovalores da matriz de covariância. Como o conjunto possui duas dimensões, variáveis X1 e X2, o número de autovetores será igual a 2. A figura 28 representa os autovetores do exemplo:

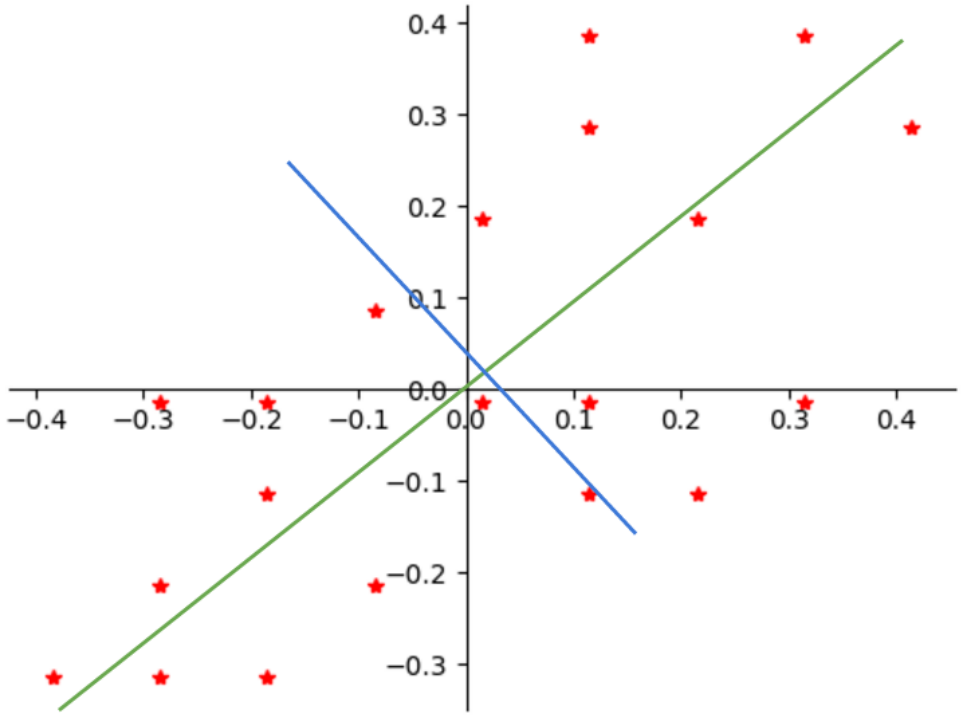

Figura 28 – Autovetores dos dados

Os autovetores calculados a partir da matriz de covariância representam as direções em que os dados acumulam a maior variância. Já os respectivos autovelores determinam a quantidade de variância naquela direção. Uma vez que obtemos os autovetores, podemos selecionar aqueles onde serão projetados os dados. Os autovetores selecionados são chamados componentes principais.

O critério para seleção dos autovetores é o total de variância que ele representa. Como já foi dito, essa variância é representada pelo autovalor de cada autovetor. A soma dos autovalores é igual a variância total dos dados. Se queremos projetar os dados para 1 dimensão, selecionamos o autovetor que possui o maior autovalor.

A tabela 22 mostra os autovalores para os autovetores calculados. Podemos observar que o primeiro autovetor representa cerca de 85% da variância dos dados, enquanto o segundo explica cerca de 15%. Os valores acumulados somam a variância total.

-	**Autovalores/Variância**	**Variância acumulada**
1	84.5979	84.5979
2	15.4021	100

Tabela 22 – Autovalores dos autovetores

Uma maneira comum de selecionar os autovetores é estabelecer um valor limite de variância desejada para o resultado. Por exemplo, se queremos manter 90% da representatividade, selecionamos quantos componentes quanto necessários para obter este valor, que é o acumulado da soma das variâncias de cada componente. Como neste exemplo iremos aplicar a redução para 1 dimensão, será selecionado o primeiro autovetor como componente principal. Por consequente, o resultado final terá 85% da representatividade dos dados originais.

Selecionados os componentes, o último passo é projetar os dados nas novas coordenadas. A figura 29 mostra o resultado desta projeção.

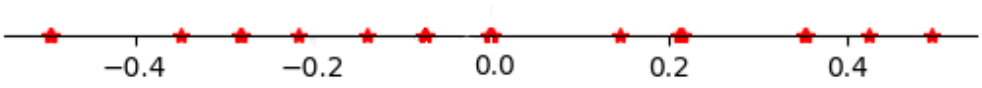

Figura 29 – Projeção dos dados no componente selecionado