ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E SISTEMAS DIGITAIS
ENGENHARIA ELÉTRICA COM ÊNFASE EM COMPUTAÇÃO

ERIC KOJI YANG IMAI
HENRIQUE MINORU HATTORI

TOPICALIZAÇÃO DE NÍVEL DE SERVIÇO

SÃO PAULO
2020

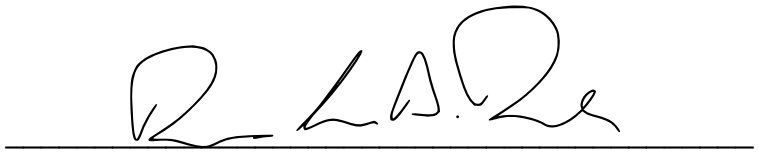
ERIC KOJI YANG IMAI
HENRIQUE MINORU HATTORI

TOPICALIZAÇÃO DE NÍVEL DE SERVIÇO

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica com ênfase em computação.

Orientador: Prof. Dr. Ricardo L. de Azevedo Rocha

SÃO PAULO
2020



Prof Dr. Ricardo Luis de Azevedo da Rocha

A todos que tornaram este projeto possível e de alguma forma ou de outra participaram de nossa formação.

AGRADECIMENTOS

Agradecemos aos nossos pais e familiares por serem os alicerces de nossas vidas, nos apoiando e nos incentivando em todas as esferas. A conclusão do nosso ciclo acadêmico na graduação é uma vitória compartilhada por nossos familiares e não teria sido alcançada sem esta parte tão importante de nossas vidas.

Ao todo corpo docente da Escola Politécnica e em especial o Professor Ricardo Rocha, que não só atuou como orientador deste trabalho de conclusão de curso, como foi um ótimo amigo durante um período tão complicado de pandemia de Covid-19, ano em que foi realizado este projeto. A ativa participação dos professores em nossa trajetória acadêmica fez de nós o que somos hoje.

Ao grupo de extensão Poli Júnior e os colegas de trabalho de estágio que nos formaram de forma complementar ao nosso curso e que iniciaram a nossa carreira profissional de modo inimaginável. Além de terem nos apoiado e incentivado a trabalhar com uma temática de conclusão de curso tão interessante e desafiador.

“O cientista não é o homem que fornece as verdadeiras respostas; é quem faz as verdadeiras perguntas”.

- Claude Lévi-Strauss

RESUMO

As redes sociais e o foco ao cliente, diferente de décadas atrás em que se focalizava no produto, são elementos desta nova realidade do século XXI, que deu espaço para termos como social listening. Neste contexto, algoritmos de Processamento de Linguagem Natural cada vez mais veem sendo procurados, a fim de trazer eficiência estratégica para as empresas. Este trabalho de conclusão de curso também é produto desta nova realidade e traz como produto um topicalizador de temas de comentários de nível de serviço baseado em Processamento de Linguagem Natural.

Palavras-Chave: Social Listening, Processamento de Linguagem Natural, Topicalização.

ABSTRACT

Social networks and the focus on the customer, different from decades ago in which they focused on the product, are elements of this new reality of the 21st century, which gave space to terms such as social listening. In this context, Natural Language Processing algorithms are increasingly being sought, in order to bring strategic efficiency to companies. This course conclusion work is also the product of this new reality and brings as a product a feedback topicalization of service level based on Natural Language Processing.

Keywords: Social Listening, Natural Language Processing, Topicalization

LISTA DE IMAGEM

LISTA DE TABELAS

LISTA DE SIGLAS

PLN	Processamento de Linguagem Natural
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
LDA	Latent Dirichlet Allocation
TF-IDF	term frequency -inverse document frequency
CBOW	Continuous Bag of Words
DM	Distributed Memory
DBOW	Distributed Bag of Words
LGPD	Lei Geral de Proteção de Dados

SUMÁRIO

1. INTRODUÇÃO

O trabalho de conclusão de curso aqui descrito teve origem como solução e inspiração na percepção de um problema real existente de uma empresa em que um dos alunos estagiou. Além de se apresentar alinhado a necessidades de outras empresas no segmento de processamento de dados qualitativos.

1.1. Contextualização

Conjuntamente a virada do século XXI, inúmeras empresas começaram a aderir um *modos operandis* diferente influenciadas pela publicação do Manifesto Ágil em fevereiro de 2021. Esta mudança de comportamento, se mostrou necessária dada a rápida velocidade em que as novas tecnologias começaram a surgir e as suas consequências de cunho econômico e social se tornaram relevantes, hoje o uso de plataformas como o *Facebook, Instagram* e o *Google* participam do nosso dia-a-dia de forma orgânica e fazem parte da cultura do século XXI.

Dada esta realidade, inúmeras companhias adotaram novas estratégias e organogramas com áreas antes não existentes, como a área de *social listening*, a fim de embasar a tomada de decisão a partir do seu público em tempo real. Porém, inúmeras dificuldades acompanham essas novas estruturas, sendo que uma delas é como, de forma automatizada, conseguir processar dados qualitativos que representam grande parte do tipo de informação advinda das interações nas plataformas digitais.

O trabalho de conclusão de curso aqui descrito busca sanar esta dor de empresas que necessitam ou possuem o potencial de tirar conclusões de dados qualitativos de forma automatizada. Um algoritmo que aponta as principais causas dos acertos e erros de produtos ou serviços no ponto de vista do cliente.

1.2. Motivação

Dois pilares principais foram os vetores para que o tema deste trabalho de conclusão de curso fosse a topicalização de nível de serviço: a primeira, se tratou da percepção de que o produto final deste projeto poderia se tornar uma solução de grande valia para inúmeras necessidades de mercado, como a percebida anteriormente no estágio de um dos alunos autores. E de forma paralela, o segundo pilar que é o aprendizado de uma temática em que ambos os alunos deste trabalho

gostariam de se aprofundar, a inteligência artificial aplicada ao processamento de linguagem natural (PLN).

1.3. OBJETIVO

Inicialmente, o trabalho de conclusão de curso possuía escopo de topicalizar as informações específicas dos dados de nível de serviço fornecidos pela empresa do estágio de um dos alunos autores, a fim de apresentarmos a correlação entre temáticas dos *feedbacks* e as notas dadas pelos consumidores. Entretanto, após percebermos a possibilidade de abrangerem o escopo para qualquer comentário de nível de serviço, assim foi feito.

Desta forma, como objetivo do algoritmo proposto temos a identificação de tópicos dentro de uma base de comentários e correlacionarmos com as notas dadas, além da apresentação das conclusões de forma clara e intuitiva com um índice de assertividade próximo a 80%

2. METODOLOGIA DE TRABALHO

A dinâmica de trabalho adotada foi pautada com base nos meios que forneceriam maior eficiência e segurança para os envolvidos, dado o projeto de conclusão de curso ter sido realizado no período de pandemia do Covid-19, que carecia dos cuidados de distanciamento social. Para isto, foi adotado métodos e plataformas sociais e de hospedagem de informações para concluir os objetivos propostos.

2.1. Método de Rotina

Com a impossibilidade de encontros presenciais em função da pandemia de Covid-19, foi adotado reuniões semanais de grupo nos sábados via vídeo chamada por meio do Google *Hangout* e reuniões quinzenais nas quintas-feiras com o professor orientador pela mesma plataforma,

De forma paralela, objetivando a maior eficiência possível, foi utilizado um modelo de pauta no *Google Drive*, onde era definido, seguindo a metodologia SMART, as entregas destinadas a cada um dos alunos do grupo.

2.2. Plataformas Utilizadas

As plataformas utilizadas com a finalidade de comunicação e hospedagem de informações, no contexto da pandemia de Covid-19 , durante o processo de desenvolvimento do trabalho de conclusão de curso podem ser observadas abaixo.

2.2.1. Google Hangout

Plataforma destinada a comunicação via chat ou vídeo conferência.

2.2.2. Google Drive

Plataforma destinada a hospedagem de conteúdos de referência, aprendizados e pautas de reunião, além da própria monografia.

2.2.3. Google Classroom

Plataforma destinada a comunicação via chat com o Professor orientador.

2.2.4. Github

Plataforma destinada a hospedagem de código fonte do projeto de topicalização de nível de serviço.

2.3. Método de Projeto

O trabalho de conclusão de curso possui limitações de tempo, pessoas e recursos. Dada esta realidade, é de grande importância a adesão de métodos de projeto que buscam obter a maior eficiência possível no desenvolvimento do produto desejado, no caso um algoritmo de topicalização de dados qualitativos. Para isto, o desafio foi abordado em etapas com algumas características de metodologia ágil, pautada na modularização de unidades funcionais. Desta forma, o desenvolvimento do projeto foi realizado em ciclos, sendo que os módulos foram sendo desenvolvidos de forma paralela por cada um dos integrantes para integração posterior. As etapas de ciclos são descritas abaixo:

2.3.1. Planejamento

Etapa que define o que deve ser entregue, a sua razão de entrega e priorização, por quem deve ser entregue e a sua respectiva data de entrega.

2.3.2. Estudo conceitual

Etapa que é realizado estudos sobre a implementação e a busca de referências para embasamento na construção lógica funcional da unidade a ser executada.

2.3.3. Produção

Etapa destinada ao desenvolvimento do código fonte do módulo funcional e submissão deste na plataforma *Github*.

2.3.4. Checagem de resultados

Reunião de checagem da funcionalidade e requisitos do módulo desenvolvido anteriormente na etapa de produção.

2.3.5. Correção

Etapa apenas realizada caso a etapa de checagem de resultados, não tenha tido resultados satisfatórios. Ela é destinada a ser um ciclo mais curto de planejamento, estudo conceitual, produção e checagem.

2.3.6. Integração

Estágio destinado a integração das unidades funcionais executadas em paralelo por cada um dos integrantes do projeto.

3. ASPECTOS CONCEITUAIS

3.1. Processamento de Linguagem Natural

Processamento de linguagem natural (PLN) é uma subárea da ciência da computação, inteligência artificial e da linguística que estuda os desafios da compreensão e geração automática de línguas humanas naturais. Sistemas de geração de língua natural convertem informação em linguagem compreensível ao ser humano e sistemas de compreensão de língua natural convertem a linguagem humana em representações mais facilmente manipuláveis por programas de computador. Usaremos a PLN para tentar compreender a linguagem natural de comentários, fazendo com que nossos m odelos extraiam sentido de linguagem humana e identifiquem o tópico abordado no comentário.

3.2. Pré-Processamento

Para utilizar técnicas de PLN se faz o tratamento e pré-processamento do texto. Dividimos na normalização do texto e em técnicas de análise de texto.

3.2.1. Normalização de texto

Para a aplicações de qualquer técnica de NLP, é necessário normalizar o texto, a fim de facilitar o processamento dos dados. Assim, executaremos as seguintes atividades:

- Tokenização de palavras no texto: o processo de separar uma sequência de caracteres em unidades, costumeiramente denominadas tokens, que individualmente possuam significado semântico.
- Normalização do formato das palavras: padronizar o formato de cada token para simplificar, posteriormente, o processamento dos dados.
- Segmentação de sentenças no texto: processo de identificar e separar as sentenças (frases) numa sequência de caracteres.

3.2.2. Técnicas de análise de texto

Aplicamos também algumas técnicas de análise de linguagem natural, para simplificação do texto e extração de sentido do texto. As técnicas serão:

- Stemming: tem como objetivo remover sufixos (ou prefixos) das palavras, a fim de encontrar a chamada raiz, a qual pode ser entendida como uma palavra primitiva de onde outras são derivadas. Esta técnica ajuda a comprimir o espaço de palavras, já que todas aquelas originadas de uma mesma primitiva terão o mesmo valor.
- Lematização: processo utilizado para encontrar a raiz de uma palavra, denominada aqui como lema. Assim mapeamos as diversas variações (também chamadas de lexemas) de palavras para uma única que tenha o mesmo significado
- Remoção de Stopwords: no processamento de linguagem natural são encontradas palavras que o valor semântico atrelados às mesmas é desprezível. Estas são denominadas de stopwords. E são ignoradas.

3.3. Bag of Words e TF-IDF

Bag of words é a representação de um texto em um documento que representa o conjunto de palavras do texto e sua frequência absoluta de ocorrência no texto. Assim, o texto é representado como um vetor.Dada uma coleção de documentos D, o conjunto de todas as palavras que aparecem nesta coleção V, conhecido como vocabulário, e considerando $w_{j,i}$ como a frequência absoluta da palavra $p_j \in V$ no documento $d_i \in D$ podemos representar d_i como:

$$d_i = [w_{1,i}\ w_{2,i} \ldots\ w_{n,i}]$$

Dada a coleção D, podemos representar os vetores de todos seus documentos em uma matriz conhecida como termo-documento. Nesta, cada coluna é a representação de um dos documentos da coleção e cada linha representa uma das palavras do vocabulário.

$$M = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix}$$

Uma desvantagem do uso da frequência absoluta das palavras para representar o texto, é a ocorrência de palavras, como artigos e preposições, terem maiores

frequência e não acrescentam nenhum valor para análise. Para amenizar este problema se usa o algoritmo TF-IDF (term frequency -inverse document frequency).

O TF-IDF, ao invés de considerar a frequência absoluta da no cálculo de $w_{j,i}$, ele considera o produto de dois termos. O primeiro, TF, é responsável por aumentar o peso para palavras frequentes em um documento, como uma frequência relativa da ocorrência da palavra no documento. Considerando $f_{t,d}$ a frequência absoluta da palavra *t* no documento *d* temos:

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

O termo IDF diminui o peso de palavras que aparecem na maioria dos documentos. Sendo *N* o número total de documentos na coleção e df_i o número de documentos no qual a palavra *t* aparece calcula-se IDF:

$$IDF(t) = log_{10} \frac{N}{df_i}$$

Assim, a matriz termo-documento é atualizada:

$$f_{i,j} = TF(t,d) * IDF(i)$$

3.4. Word Embeddings, Word2Vec e Doc2Vec

Word embedding é a representação vetorial de uma palavra. Vetores de palavras individuais podem ser obtidos a partir de redes neurais treinadas para tarefas de classificação de texto. Representações geradas desta forma resultam em vetores densos em um espaço contínuo com um número de dimensões que pode ser definido como hiperparˆametro da rede.

Word2Vec é uma nomenclatura dada a um conjunto de duas técnicas bem sucedidas para se obter word embeddings: CBOW e Skip-gram (MIKOLOV et al., 2013). Tanto o CBOW (Continuous Bag of Words) quanto o Skip-gram se baseiam na hipótese de que palavras com significados similares costumam aparecer nos mesmos contextos. Assim, se possuirmos uma quantidade suficientemente grande de documentos, podemos gerar representações confiáveis de palavras olhando suas palavras vizinhas. Tal vizinhança é definida por um hiperparâmetro *n* que determina a

distância entre a ocorrência de duas palavras para estas serem consideradas vizinhas.

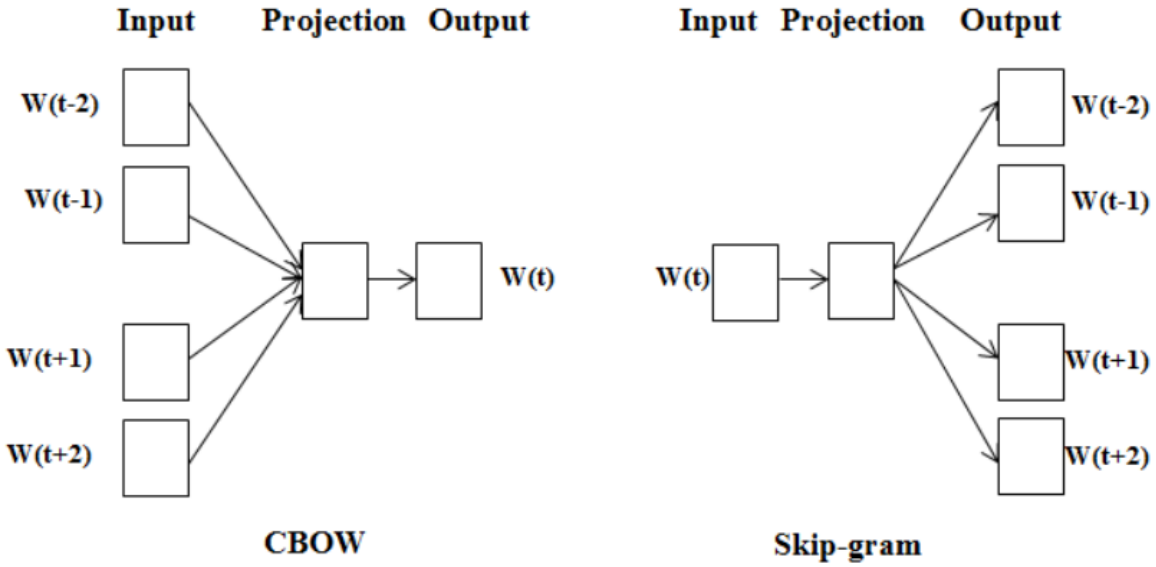


Figura 1 - Modelo CBOW e Skip-gram (MIKOLOV et al., 2013).

O CBOW é treinado da seguinte forma: primeiro, realiza-se a extração de conjuntos de 2n+1 palavras (uma central e suas vizinhas a esquerda e a direita), dado a ocorrência desta sequência em um documento. Então, é utilizada uma rede neural para prever a probabilidade da palavra central, dada às suas vizinhas. A rede recebe como entrada os embeddings das palavras vizinhas que são inicializados de forma aleatória e são posteriormente atualizados junto com os parâmetros da rede, buscando maximizar o acerto na predição. Já o Skip-gram é treinado da forma inversa, recebendo o embedding da palavra central e sendo treinado para prever as palavras vizinhas a esta.

Uma extensão possível das técnicas acima se baseia em considerar aspectos da estrutura interna da palavra. Isto é feito considerando cada palavra como um conjunto constituído de um identificador único e de todas as cadeias de n caracteres que ocorrem naquela palavra. Neste esquema a rede neural aprende representações para cada um dos elementos deste conjunto e estas são somadas para constituir a representação da palavra nal (BOJANOWSKI et al., 2017). Esta extensão ao Word2Vec ficou conhecida como FastText.

Uma vez obtido o word embedding é possível gerar a representação de um documento a partir da soma ou média ponderada dos vetores de suas palavras.

Uma representação de documentos a partir de médias ou soma de word embeddings permite a utilização das relações semânticas aprendidas pelos embeddings para gerar a estrutura semântica do documento final. Apesar disto, a utilização de operações como média e soma de vetores remove noções importantes para a semântica de um documento como, por exemplo, a sequência na qual ocorrem as palavras.

Para evitar esta perda de informação, foram propostos dois algoritmos para o aprendizado de embalagem dos documentos, reunidos em um pacote chamado de Doc2Vec. A premissa dos algoritmos é de que a representação de documentos seja apreendida junto com os word embeddings através da incorporação de um identificador de documento no treino da rede neural (LE; MIKOLOV, 2014). No algoritmo conhecido como Distributed Memory (DM), esse identificador é colocado como um input adicional ao modelo do CBOW. Da mesma forma que é feito com as palavras, o algoritmo inicializa o embedding do documento com valores aleatórios e aprende o valor adequado ao longo do treino.

Na versão conhecida como Distributed Bag of Words (DBOW), é utilizado apenas o embedding do documento para prever palavras aleatórias amostradas deste documento.

Uma vez treinado o modelo é possível utilizar os embeddings aprendidos de qualquer um dos dois algoritmos como representação vetorial do documento.

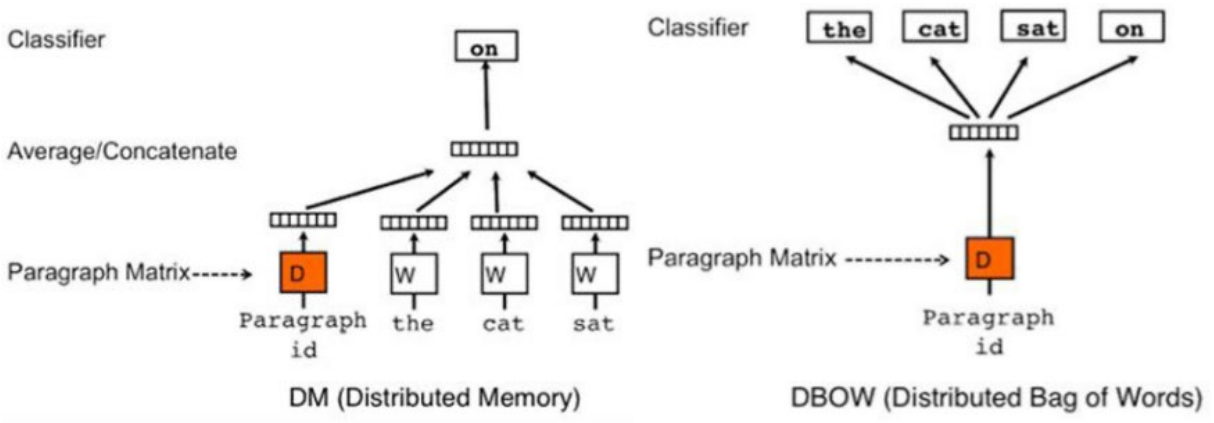


Figura 2 - Os modelos DM e DBOW, respectivamente (LE; MIKOLOV, 2014).

3.5. K-Means

K-Means é um algoritmo de aprendizado de máquina não supervisionado. Dado um conjunto de dados, o algoritmo criará uma classificação de acordo com as métricas, agrupando em clusters. Um centróide é o local imaginário ou real que representa o centro de cada cluster, sendo assim, se defini um número de destino K, que se refere ao número de centróides que você precisa no conjunto de dados. O número definido como K representa a quantidade de classes no qual desejamos segmentar os dados.

Em outras palavras, o algoritmo k-Means identifica o número k de centróides e, em seguida, aloca todos os pontos de dados para o cluster mais próximo, enquanto mantém os centróides o menor possível.

Para achar o K ótimo, usa-se o método do cotovelo. Ao rodar várias vezes o algoritmo aumentando K, a diferença entre clusters vai ficando muito pequena e a diferença intra-clusters aumentando. Busca-se um valor K de equilíbrio que torna a diferença entre os clusters mais homogênea que os clusters sejam os mais diferentes possíveis entre si. Então busca-se uma quantidade de agrupamentos em que a soma dos quadrados intra-clusters seja a menor possível, sendo zero o resultado ótimo.

Exemplificando o parágrafo anterior, na figura seguinte, os pontos laranjas são os números de clusters em função da distância intra-cluster. Observa-se que ao aumentar o número de clusters, a distância intra-cluster diminui, mas a derivada dessa diferença tende a um limiar. A reta verde é a reta que liga o ponto com mais cluster e menos clusters, o ponto ótimo é o ponto laranja mais longe da reta verde.

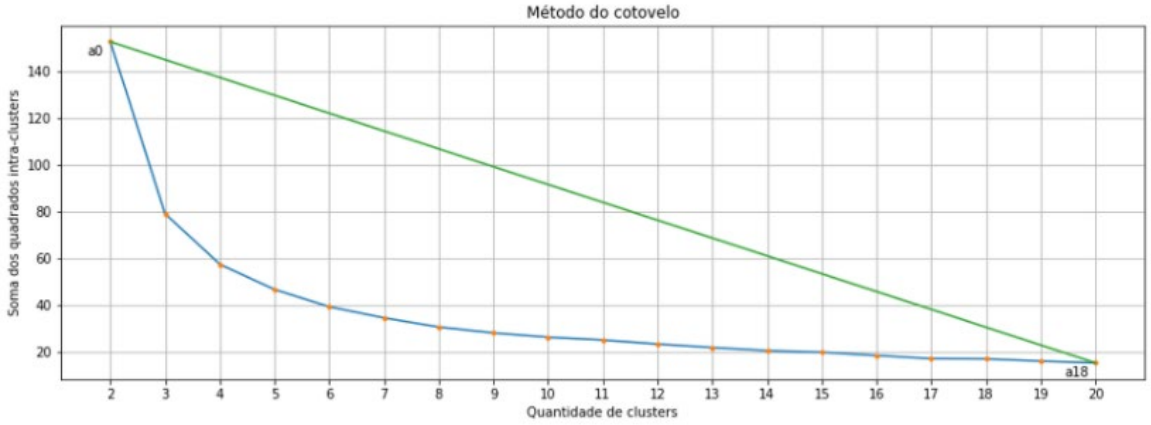


Figura 3 - Exemplo do método do cotovelo.

3.6. Hierarchical Density-Based Spatial Clustering of Applications with Noise

Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) que é um algoritmo de clusterização hierárquica desenvolvido por Campello, Moulavi, and Sander [1]. Assim como o K-means, ele procura clusterizar em clusters de acordo com a distância dentro de um centróide, levando em conta a densidade de pontos para definir o centróide.

Assumindo um set de dados de pontos cartesianos, busca dividir esse set de dados em diferentes com a maior densidade possível de pontos dentro da área do cluster. Porém, pontos que tem poucos pontos à sua volta ou estão muito longe de regiões com muitos pontos, diminuem a densidade de pontos ao ser incorporado a um cluster. A fim de desconsiderar esses pontos “ruídos”, escolhe um k, número de pontos dentro de um cluster. Quanto menor o K, menos robusto a ruídos o algoritmo.

Assim para cada ponto, cria-se uma área de cluster que engloba esse número de pontos, o raio desse cluster se chama *core distance*. Quanto maior o *core distance* desse ponto, menor a densidade de pontos, maior a chance dele ser um ruído. Como exemplo visual, a próxima imagem mostra um ponto (à esquerda) com maior densidade e menor *core distance* e outro com menor densidade e maior *core distance*.

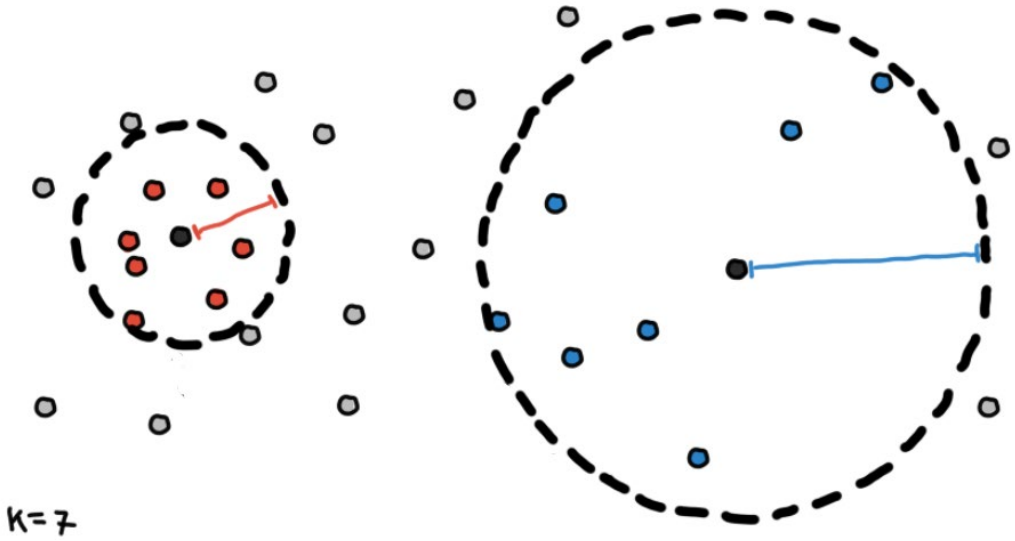


Figura 4 - Exemplo da relação de *core distance* e densidade de pontos.

O próximo passo é definir qual o nível de densidade (λ) de cada cluster, ou seja o *core distance* mínimo para cada cluster.

Começa conectando pontos “próximos” uns aos outros. “Perto” é determinado pelo nível de densidade de pontos definido por λ e dizemos que dois pontos estão próximos o suficiente se sua distância euclidiana for menor que λ.

Desenhamos uma esfera com raio λ em torno de cada ponto.

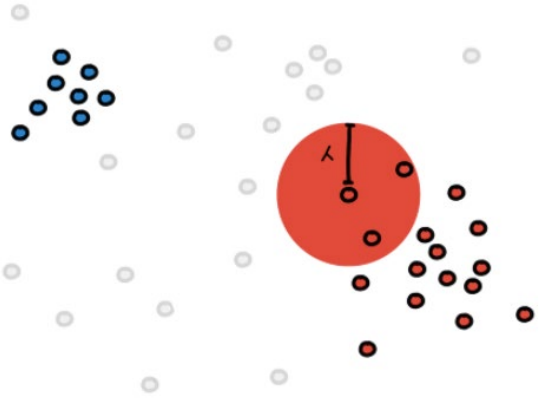


Figura 5 - Regiões com *core distance* menor que λ.

Desenha-se uma esfera com raio λ em torno de cada ponto. Conecta-se o ponto a todos os pontos dentro de sua esfera λ. Se dois pontos estiverem conectados, eles pertencem à mesma região e devem ter a mesma cor.

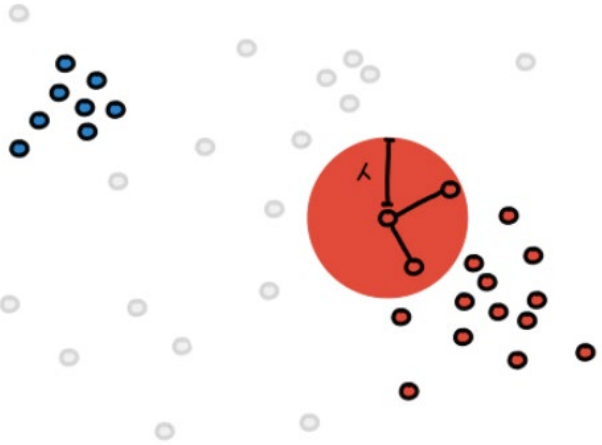


Figura 6 - Conecta os pontos dentro da esfera de raio λ.

Fazendo-se isso para cada ponto e o que se forma são vários pontos conectados, os clusters.

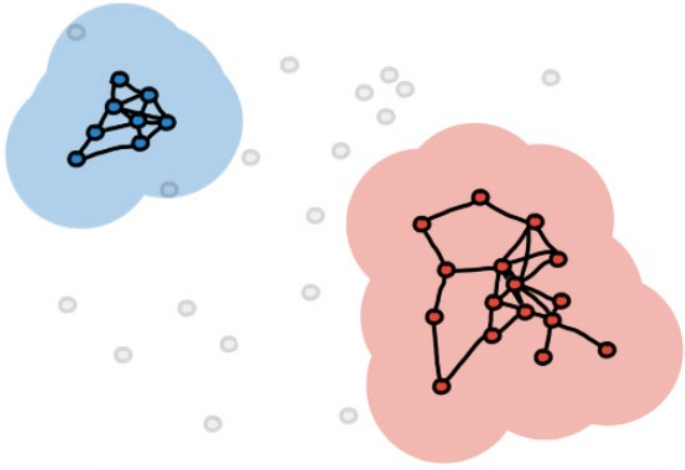


Figura 7 - Clusters formados.

Conforme se aumentando λ mais cluster podem surgir, ou crescer, ou fundirem. Para um ponto ser incluído dentro da esfera de raio λ, a core distance entre o ponto mais próximo dentro da esfera de raio λ, menor que λ.

Assim dois pontos a e b, serão do mesmo cluster se estiver na dentro da região de distância de alcance mútuo (*mutual reachability distance*), que é definida por:

$$mutual\ reachability\ distance\ (a,b) = max(core\ distance(a), core\ distance(b), distance(a,b))$$

Onde:

- core distance(a) ≤ λ
- core distance(b) ≤λ

- distance(a, b) ≤λ

Depois, o algoritmo procura aglutinar cluster que tem pontos com distância de alcance mútuo até que não haja clusters para aglutinar sob esse critério. É o equivalente a construir uma árvore de abrangência mínima e o algoritmo usa teoria dos grafos para ser mais eficiente.

3.7. Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) é um método para gerar um modelo probabilístico para conjunto de textos. A premissa adotada pela técnica é que existe um conjunto de k tópicos latentes responsáveis pela geração de documentos. Os tópicos são uma distribuição de probabilidade sobre as palavras do conjunto.

Partindo da hipótese de que a distribuição de probabilidade dos tópicos sobre os documentos e amostrada de uma distribuição de Dirichlet, é possível, a partir da conjunto de documentos, estimar os parâmetros associados às funções de probabilidades do modelo. Então, para cada documento na conjunto, é possível extrair uma distribuição representando a probabilidade deste documento pertencer a cada um dos k tópicos. A distribuição neste caso, é uma representação vetorial explícita do documento.

4. TECNOLOGIAS UTILIZADAS

4.1. Linguagem de Programação

O projeto de conclusão de curso foi desenvolvido a partir da linguagem de programação python em sua versão 3.7.3. Esta linguagem foi escolhida em função de possuir características que facilitam a utilização de bibliotecas externas de processamento de linguagem natural, além de ser uma linguagem que ambos os integrantes do grupo possuíam naturalidade.

4.2. Bibliotecas

A fim de conferir maior velocidade no desenvolvimento do produto final deste trabalho de conclusão de curso, foi utilizado algumas bibliotecas externas ao longo da construção de cada um dos módulos funcionais. Nesta sessão é listado as bibliotecas utilizadas em cada segmento de tecnologia necessário.

4.2.1. Manipulação de Dados

- Pandas: biblioteca criada objetivando a manipulação e análise de dados. Em particular, oferece estruturas e operações para manipular tabelas numéricas e séries temporais
- Regex: biblioteca com métodos para manipulação de expressões regulares. Esta também foi utilizada para algumas técnicas de pré-processamento em NLP.

4.2.2. Operações Vetoriais

- Numpy: pacote em linguagem Python que suporta vetores e matrizes multidimensionais, possuindo uma larga coleção de funções matemáticas de suporte a estas estruturas.

4.2.3. Processamento de Linguagem Natural

4.2.3.1. Pré-processamento e vetorização

- Spacy: biblioteca open source, com modelo estatístico pré treinado para português. Foi usado seu modelo treinado para portugues em que a cada palavra é vetorizada em um vetor de 96 valores.
- Gensim: biblioteca para modelagem de tópicos não supervisionados e processamento de linguagem natural. Desta biblioteca utilizamos o modelo:
- NLTK: é um conjunto de bibliotecas e programas para processamento simbólico e estatístico da linguagem natural. No projeto foi utilizado principalmente como ferramenta de pré-processamento linguístico.

- String: biblioteca utilizada para manipulação de strings na etapa de pré-processamento.

4.2.4. Clusterização

4.2.4.1. Modelos de clusterização

- Sklearn: biblioteca open source de machine learn com apredizado supervisionado ou não. Foi utilizado os seguintes modelos de clusterização:
  - KMeans: modelo que usa o algoritmo K-means.
  - DBSCAN: modelo que usa o algoritmo HDBSCAN.
- Yellowbrick: biblioteca utilizada para cálculo do ponto ótimo de k (k = número de clusters), no algoritmo K-means.

4.2.4.2. Rotulação de temática

- Gensim (LDAmodel): biblioteca utilizada para importação do algoritmo de LDA, que foi o método responsável pela rotulação de temáticas de clusters do projeto.

5. ESPECIFICAÇÃO

5.1. Requisitos

Os requisitos de projeto buscam definir as necessidades funcionais e não funcionais de um projeto de software. A sua descrição busca definir características de escopo a serem respeitados que servem de guia para todas as etapas de desenvolvimento do produto final.

5.1.1. Requisitos Funcionais

Os requisitos funcionais são definidos por serem as características de funcionalidade que devem ser respeitadas a fim do projeto entregar o serviço a qual foi destinado de forma integral. Estes são:

- Realizar a topicalização de dados qualitativos por temática.
- Identificar tópicos e nomeá-los sem a necessidade de definição prévia.
- Possuir assertividade acima de 80%.
- Retornar um arquivo em formato *.xlsx*

5.1.2. Requisitos Não Funcionais

Os requisitos não funcionais são definidos por descrever não o que o sistema fará, mas sim como ele fará. Estes são listados abaixo no caso do projeto relativo a este documento.

- Realizar a topicalização a partir de dados qualitativos em formato *.xlsx.*
- Retornar um arquivo em formato *.xlsx* com dados dispostos de forma de fácil utilização posterior.

5.2. Visão funcional

O algoritmo de topicalização de nível de serviço foi desenvolvido de forma modular. Ele pode ser observado no fluxograma abaixo, com os respectivos dados de entrada e saída.

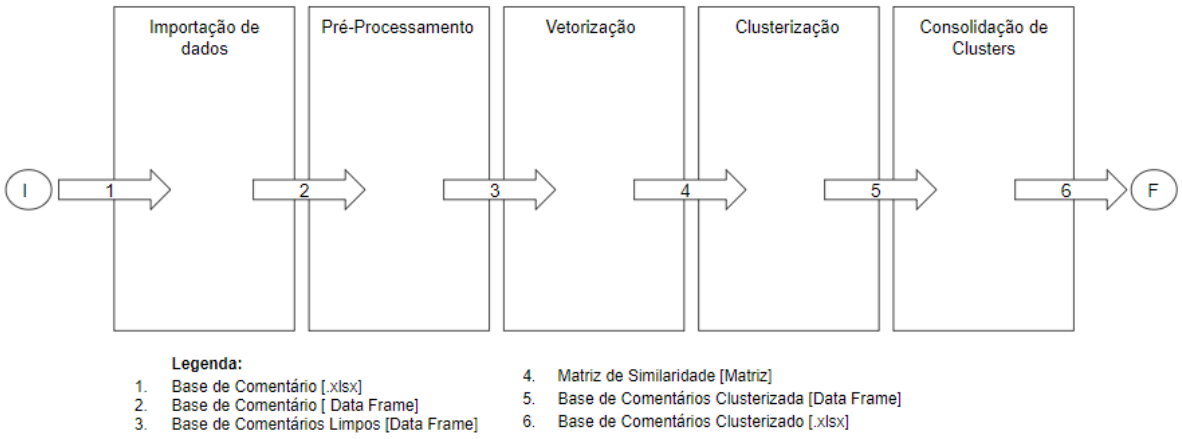


Figura 8 - Visão funcional.

5.2.1. Importação de dados

Unidade responsável pela importação de dados qualitativos de uma base em formato *.xlsx* para formato *Data Frame* para a etapa de Pré-Processamento.

5.2.2. Pré-processamento

Unidade responsável pela exclusão de dados vazios, normalização das informações, formatação e extração do conteúdo semântico dos dados qualitativos disponibilizados em *Data frame*. O seu retorno também possui formato *Data Frame*

5.2.3. Vetorização

Unidade que objetiva, a partir de métodos de processamento de linguagem natural, construir uma matriz de similaridade entre os dados qualitativos pré-processados pela etapa anterior e retornar em formato de lista de listas.

5.2.4. Clusterização

Unidade responsável pela topicalização e identificação das temáticas de cada um dos tópicos gerados a partir de uma matriz de similaridade entre os dados qualitativos, retornando um *Data Frame* com a identificação de tópico.

5.2.5. Consolidação dos Clusters

Unidade que a partir dos resultados da clusterização busca realizar o agrupamento de temáticas semelhantes separadas na etapa anterior. O seu retorno é dado em uma planilha de excel, conjuntamente a um gráfico.

5.3. Visão de Código

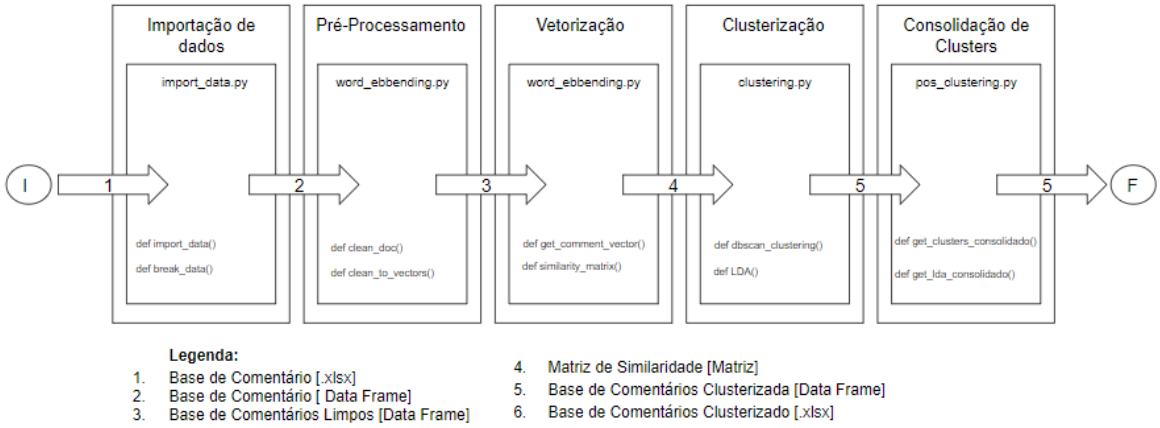


Figura 9 - Visão de código.

5.3.1. Importação de dados

Etapa funcional organizada no arquivo import_data.py. Os métodos utilizados são: *import_data()* e *break_data()*.

5.3.2. Pré-processamento

Etapa funcional organizada no arquivo word_ebbending.py. Os métodos utilizados são: *clean_doc()* e *clean_to_vectors()*

5.3.3. Vetorização

Etapa funcional organizada no arquivo word_ebbending.py. Os métodos utilizados são: *get_comment_vector()* e *similarity_matrix()*

5.3.4. Clusterização

Etapa funcional organizada no arquivo clustering.py. Os métodos utilizados são: *dbscan_clustering()* e *latent_dirichlet_allocation()*.

5.3.5. Consolidação de Clusters

Etapa funcional organizada no arquivo pos_clsutering.py e os seus métodos são get_clusters_consolidado() e get_lda_consolidado()

6. IMPLEMENTAÇÃO

A partir da metodologia de trabalho antes citada no tópico 2 deste relatório e seguindo as especificações do tópico 5, foi desenvolvido o algoritmo proposto. Nesta sessão é detalhada a forma de implementação de cada módulo funcional, assim como eventuais erros realizados pela dupla de integrantes.

6.1. Importação de Dados

O módulo de importação de dados foi desenvolvido sem maiores problemas, nele foi utilizado a biblioteca *pandas* para extrair os dados fornecidos em formato *.xlsx* pela empresa de estágio de um dos integrantes, pelo método *import_data().*

Posteriormente, nesta mesma etapa, a função *break_data()* é responsável por quebrar os comentários por períodos, a partir de marcadores de pontuação (“.”, “!”, “?”) e alguns conectivos (“mas”, “porém”, “entretanto”, “e”).

6.2. Pré-Processamento

Algumas técnicas foram utilizadas no módulo de pré-processamento, etapa esta que objetiva preparar os dados para sua transformação vetorial. As técnicas utilizadas são listadas em ordem de aplicação abaixo e os seus detalhes são descritos no tópico de numeração 3.2 deste mesmo relatório.

Técnicas de Pré-Processamento:

- Normalização do formato
- Tokenização de palavras
- Remoção de Stopwords
- Stemming e Lematização

Estas técnicas foram utilizadas na construção dos métodos clean_doc() e clean_to_vectors(). Para isto, foi utilizado a biblioteca externa Spacy, apesar de, inicialmente, esta etapa ter sido desenvolvida a partir da biblioteca NLTK. Esta mudança foi feita ao longo do projeto dado que a biblioteca Spacy possui alguns métodos de grande utilidade quando comparado com o primeiro a ser implementado.

6.3. Vetorização

A etapa de vetorização objetivou dois módulos distintos a serem testados, dado que não havia conhecimento prévio de desempenho em cada um destes. O primeiro

partiu da premissa de que as características vetoriais de cada dimensão do comentário seriam relevantes para a topicalização e o segundo de que não haveria essa relevância, apesar de ambos os casos terem utilizado a biblioteca Spacy para a vetorização de comentários baseado em técnicas de *word ebbending*.

A primeira versão de módulo possui formato matricial (vetor de vetores), sendo que cada linha representa um comentário e cada coluna o seu valor da respectiva dimensão advindo do processo de *word ebbending.* De forma distinta o segundo modelo é organizado também em formato matricial (vetor de vetores), entretanto, de dimensão NxN, sendo N o número de comentários da base de dados, isto porque cada valor da matriz é o valor de semelhança vetorial de cada um dos comentários.. Esta semelhança é igual o valor cosseno entre os vetores, calculados a partir da fórmula cos = <u,v>/(|u|.|v|).

6.4. Clusterização

A etapa de clusterização objetiva topicalizar os dados conforme a sua semântica predominante. Para isto foi realizado dois modelos de módulo, sendo que o segundo foi adotado após o primeiro não atingir resultados esperados.

O primeiro modelo adotado foi desenvolvido a partir do método de clusterização por centróides a partir do algoritmo de K-Means, via biblioteca sklearn, e executado em três parâmetros distintos a fim de se entender os resultados. Estas três variantes foram a execução do algoritmo com k (k = número de clusters) igual a 40, igual a 90 e otimizado via método do cotovelo pela biblioteca YellowBrick, entretanto, como será apresentado no tópico de número sete não houve resultados satisfatórios.

O segundo modelo, escolhido como método final de projeto, foi a clusterização de densidade implementado via biblioteca dbscan. Assim como modelo anterior foi executado com diferentes parâmetros de min_sample (quantidade de amostras mínimas para se identificar um cluster) sendo estes iguais a 3,4, e 5.

Por fim, foi aplicado técnicas de LDA, a partir da biblioteca gensim, para rotular as temáticas de cada um dos clusters. Concluído esta etapa, os dados foram exportados via .xlsx para a etapa seguinte de consolidação.

6.5. Consolidação de Clusters

A fim de diminuir o número de clusters, esta etapa consiste no agrupamento dos clusters criados na clusterização, que possuem o mesmos rótulos de temática ou com o grupo de palavras iguais identificados na aplicação do modelo de LDA.

Diferentes clusters podem ter palavras com maior peso semelhante, mas não são clusterizados no mesmo cluster devido a influência de outras palavras ou termos que não alteram a temática do comentário. Por isso, ao agrupar esses clusters com a mesma palavra de maior peso se torna possível e efetivo.

7. TESTES E RESULTADOS

Em busca de um resultado de topicalização satisfatório foi realizado testes em diferentes opções de modelos, sendo que a base de comentários foi submetida a distintos métodos com combinações de diferentes métodos de de pré-processamento, vetorização de comentários, clusterização, rotulação e agrupamento de clusters. Neste tópico é apresentando a sequência de modelos construídos e testados em ordenação temporal, assim como os aprendizados advindos de cada um deles que foram utilizados para os modelos seguintes.

7.1. Modelo 1

7.1.1. Especificações

- Base: 918 amostras
- Vetorização: vetor de comentário
- Clusterização: KMeans e Método do Cotovelo

7.1.2. Resultados e aprendizados

Nesta fase, foi usado a primeira base de comentários recebida por nós. Está englobava diversos comentários de vários bares diferentes. Os comentários desta base foram apenas pré processados, e a forma em que foram vetorizados os foram feita de duas formas diferentes:

- Usando a média dos vetores das palavras - através do método word2vec, cada palavra é vetorizada em um vetor de 96 posições, dentro do comentário é feito a média escalar desses vetores.
- Vetor do comentário como um todo - pelo método doc2vec.

Pelo método do cotovelo, o número ótimo de clusters foi 5, em ambos os casos. Porém os clusters formados não estavam bem definidos, tendo comentários de assuntos variados e confirmados pela topicalização. Em um mesmo cluster, havia comentários sobre cerveja, atendimento, comida, por exemplo, todos misturados. Como se verifica nos tópicos gerados para cada clusters.

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
atendimento	excelente	nao	top	bom
lugar	atendimento	atendimento	lugar	atendimento

cerveja	super	cerveja	otimo	lugar
otimo	ambiente	lugar	bom	otimo
ambiente	melhor	cervejas	atendimento	excelente
excelente	bom	bem	sensacional	sensacional
bom	cerveja	bar	incrivel	ambiente
boa	lugar	boa	maravilhoso	cerveja
recomendo	otimo	bom	excelente	maravilhoso
comida	sensacional	excelente	ambiente	lindo

Tabela 1 - Tópicos dos clusters no teste 1.

A primeira hipótese que levantamos poderia ser a base que havia apenas 918 comentários, uma base de dados muito pequena. E fizemos a requisição de uma base maior.

7.2. Modelo 2

7.2.1. Especificações

- Base: 8000 amostras
- Vetorização: vetor de comentário e cosseno entre vetores
- Clusterização: KMeans e Método do Cotovelo

7.2.2. Resultados e aprendizados

Com uma base maior com cerca de 8000 comentários, foi feito o mesmo procedimento no teste 1. Porém, esta base de comentários tinha uma temática diferente, era sobre vendas de um e-commerce.

Foi feito novas etapas de tratamento dos vetores dos comentários igual a do teste 1 e foi implementada outras duas formas de análise de similaridade entre vetores de comentários:

- similaridade euclidiana entre vetores: para a etapa de clusterização foi utilizado os vetores integrais para análise.
- similaridade utilizando cosseno entre vetores: para a etapa de clusterização foi utilizado cossenos entre os vetores.

Da mesma forma que no teste 1, o método do cotovelo nos indicou que o melhor número de clusters era 5. Usando os 2 métodos de vetorização, o resultado de cada método foi clusterizado e topicalizado. Com uma base maior, esperávamos uma melhor clusterização e ainda com uma temática diferente poderíamos analisar o desempenho do nosso modelo para outras temáticas.

O resultado foi insatisfatório. Os clusters dos diferentes métodos de vetorização não haviam diferenças, O que foi confirmado com a topicalização, onde os tópicos gerados de cada cluster tinham palavras parecidas. Nessa nova base, notamos que tinha poucas temáticas para se clusterizar, pois os comentários apenas catequizavam a entrega (rápida, pontual atrasada) e qualidade da bebida entregue (gelada ou quente).

Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**
entregar	entregar	preco	entregar	entregar
produto	produto	dia	comprar	rapido
pedir	preco	atendimento	pedir	entrega
receber	qualidade	entregar	dia	preco
chegar	rapido	produto	produto	atendimento
comprar	variedade	variedade	receber	produto
dia	rapidez	recomendar	chegar	prazo
prazo	comprar	ok	cervejar	demora
site	atendimento	prazo	prazo	fretar
cervejar	promocao	entrega	site	frete

Tabela 2 - Tópicos dos clusters no teste 2.

Porém, foi notado uma melhora na clusterização pelo método de similaridade vetorial baseada em cossenos, principalmente na primeira base dos comentários dos diversos bares, que possuíam maiores temáticas dentro de cada comentário. Método de vetorização que foi decidido ser utilizado.

A partir destes resultados levantamos a seguinte hipóteses: o problema de, no

mesmo comentário, ter vários temas sendo abordados dificultava mais que o baixo volume de dados.

7.3. Modelo 3

7.3.1. Especificações

- Base: 8000 amostras
- Quebra de comentários em períodos menores
- Vetorização: cosseno entre vetores
- Clusterização: KMeans e Método do Cotovelo

7.3.2. Resultados e aprendizados

Para testar a hipótese levantada no teste 2, foi implementada o break_data(). Este método quebra os comentários em em sua pontuação gramatical. Por exemplo o comentário :

“Gostei do produto, veio tudo certo , chegou antes do prazo. Só a transportadora que é meio difícil. Mas a empresa se colocou disponível pra ajudar a resolver.” , id 51015396

é quebrado em três “sub-comentários”:

1. “Gostei do produto, veio tudo certo , chegou antes do prazo”, id 51015396-0
2. “Só a transportadora que é meio difícil”, id 51015396-1
3. “Mas a empresa se colocou disponível pra ajudar a resolver “, id 51015396-2

O id do comentário original é mantido, e nos “sub-comentários” recebem novos ids formados pelo id do comentário original acrescido de um hífen e número.

Assim, a base de comentários “quebrados” passa pelo processo de pré processamento, clusterização e topicalização.

Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**
cervejar	atendimento	atendimento	cervejar	cervejar
atendimento	cervejar	excelente	ter	atendimento
bar	excelente	otimo	voltar	recomendo
peno	otimo	top	comido	excelente

chopp	agradavel	felipe	conhecer	super
excelente	bar	maravilhoso	melhor	ambiente
visitar	comido	sensacional	esperar	chopp
pra	melhor	barros	estar	otima
restaurante	super	otimos	degustacao	comido
qualidade	chopp	luiz	visitar	comida

Tabela 3 - Tópicos dos clusters no teste 3.

Nesta tabela, mostra todo processo feito na base de dados dos diferentes bares. Apesar de as palavras dos tópicos do LDA não mudarem muito, houve uma melhora ao se ler os comentários de cada cluster. E identificou-se a presença de comentários e “sub-comentários” que não agregam nenhum sentido, apenas poluindo o cluster e tornando a topicalização ruim.

Para isso, decidiu-se buscar um novo modelo para se eliminar os comentários que não agregam valor semântico para o cluster, neste projeto chamado de outliers.

7.4. Modelo 4

7.5. Especificações

- Base: 1647 amostras
- Quebra de cometários em períodos menores
- Vetorização: cosseno entre vetores
- Clusterização:HDBSCAN

7.6. Resultados e aprendizados

O novo modelo escolhido foi o HDBSCAN, que executa o algoritmo desejado após os resultados do teste 3. Também, foi recebido uma nova base de comentários de um local específico, com 1647 comentários e ao se quebrar os comentários se resulta numa base de 3705 comentários.

Com a substituição do modelo de clusterização K-means pelo HDBSCAN, houve uma melhora na clusterização. Porém, aumentou de forma significativa o número de clusters.

Com a topicalização desses clusters, houve uma melhora na identificação do tema de cada cluster, e também notou que alguns clusters tinham tópicos parecidos. Diante disso, foi criado um método para consolidar os clusters que tinham a mesma temática. E o cluster de outliers foi feito uma outra rotina para identificar os clusters que contêm estes comentários, e depois descartados os que realmente não apresentam nenhum valor para análise.

Usando o parâmetro de min_sample do HDBSCAN igual a quatro, foi obtido 40 clusters, que foram reduzidos a 16 novos clusters. Destes, 7 foram descartados por não acrescentarem nenhum valor semântico, resultando em 9 clusters.

Novo cluster	**Temática**
New_cluster_10	lugar
New_cluster_11	otima cerveja
New_cluster_12	otimo ambiente
New_cluster_13	prato
New_cluster_15	recomendar
New_cluster_3	ambiente
New_cluster_5	atendimento
New_cluster_7	cerveja
New_cluster_8	comida

Tabela 4 - Reclusterização para min_sample =4.

Destes 9 clusters foram agrupados em 5 temáticas de acordo com similaridade entre a temática dos clusters:

- cerveja
- atendimento
- lugar
- comida
- recomendar

7.7. Modelo 5

7.7.1. Especificações

- Base: 1647 amostras
- Quebra de comentários em períodos menores
- Vetorização: cosseno entre vetores
- Clusterização:HDBSCAN

7.7.2. Resultados e aprendizados

O mesmo procedimento do teste 4 foi feito utilizando min_sample do HDBSCAN igual a três. A clusterização resultou em 91 clusters, que foram reclusterizados em 26 novos clusters, dos quais 13 foram descartados por não apresentarem sentido semântico.

Novos cluster	Temática
New_cluster_2	almoçar
New_cluster_3	ambiente
New_cluster_4	atendimento
New_cluster_6	cerveja
New_cluster_7	comida
New_cluster_10	experiencia
New_cluster_14	lugar
New_cluster_15	petisco
New_cluster_16	porções
New_cluster_17	prato
New_cluster_20	recomendar
New_cluster_21	refeições
New_cluster_26	valer(valer a pena)

Tabela 5 - Reclusterização para min_sample = 3.

Assim como no teste anterior, foi reagrupado nas cinco temáticas e medido sua acurácia dentro destes grupos e o total ponderado. O total de comentários clusterizados foram 672 com um acerto de 79,8%.

:

Tópico	Número total de comentários	Número de comentários corretos	Acurácia
cerveja	152	118	78%
atendimento	273/106*	112/92*	41%/87%*
lugar	140	99	71%
comida	73	63	86%
recomendar	31	29	94%
Total	**672**	**420**	**62,54% / 79,8%***

*Cálculo de assertividade também realizado partindo da premissa que os dados adicionais não relacionados a atendimento são termos genéricos como “excelente” no caso do cluster “atendimento”.

Tabela 6 - Reagrupamento em tópicos e acurácia para min_sample = 3.

8. CONCLUSÕES

Com os resultados obtidos da clusterização dos comentários aliado ao *rating* que acompanha os comentários de avaliação da internet, é possível identificar problemas e fortalezas do local analisado. Assim, definir planos de ação dentro da estratégia do estabelecimento.

Como mostrado no item anterior, o modelo HDBSCAN junto com o método de agrupamento de clusters devolve resultados muito bons. O K-means por outro lado se mostrou pouco eficaz, porém é uma ótima ferramenta para identificar possíveis temáticas que podem ser levadas em conta.

Outro processo que ficou evidente da sua importância, é o tratamento e pré-processamento dos dados. Estes têm impacto muito significativo para o desempenho de qualquer modelo de machine learning não supervisionado.

O uso de outras técnicas como o uso de bigrams ou trigrams para a topicalização também indicam bons resultados de acordo com a literatura encontrada. Além disso, usar um modelo de vetorização de palavras da língua portuguesa melhor treinado, também pode melhorar a acurácia da clusterização e topicalização.

9. REFERÊNCIAS

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. 2013.

LE, Q.; MIKOLOV, T. Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32. [S.l.]: JMLR.org, 2014. (ICML'14), p. II–1188–II–1196.
HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. The Elements of Statistical Learning.

New York, NY, USA: Springer New York Inc., 2001. (Springer Series in Statistics).
GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. [S.l.]: MIT press, 2016.

JURAFSKY, D.; MARTIN, J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

IGOR, BOLSHAKOV.; ALEXANDER, GELBUKH. Computational Linguistic: Models, Resources, Applications. 1st. ed.Tresguerras 27,06040, DF, México, 2004.