Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Abordagem para Agrupar Dados de NF-e com base em Autoencoder

Johannes Peter Schulte

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Geraldo Pereira Rocha Filho

Brasília
2022

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Abordagem para Agrupar Dados de NF-e com base em Autoencoder

Johannes Peter Schulte

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Geraldo Pereira Rocha Filho (Orientador)
CIC/UnB

Prof. Dr. Luís Paulo Faina Garcia
CIC/UnB

Prof. Dr. Vinicius Ruela Pereira Borges
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 25 de março de 2022

Dedicatória

Dedico este trabalho às três mulheres mais importantes da minha vida. À minha mãe, Elfriede Carla Schulte, por todos os sacrifícios feitos para que eu pudesse chegar até aqui. À minha irmã, Heidi Luise Schulte, por ser minha inspiração durante toda a vida. Por fim, à minha namorada, Giovana Souza Batista, por ser a primeira pessoa que proveu o conforto necessário para que eu pudesse externar meus sentimentos.

Agradecimentos

Gostaria de agradecer aos meus familiares que forneceram o suporte necessário durante esta jornada, à minha namorada que sempre esteve ao meu lado, e aos meus amigos que me fizeram sorrir em momentos difíceis. Em especial, agradeço ao meu amigo Renato Avellar Nobre pela ajuda em todas as fases de realização deste trabalho.

Agradeço ao professor Geraldo Pereira Rocha Filho que foi um orientador muito atencioso e prestativo, além de ser a pessoa que mais me ensinou durante meu tempo na Universidade de Brasília.

Também agradeço aos integrantes do Núcleo de Dados do Tribunal de Contas da União que ajudaram na realização deste trabalho. Agradecimento especial para meu orientador de estágio, Tiago Marafante Lins de Souza, por prover os dados que possibilitaram a existência do estudo aqui apresentado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

O método mais comum utilizado para documentar transações monetárias no Brasil é a emissão de nota fiscal eletrônica (NF-e). A auditoria de notas fiscais eletrônicas é essencial, o que pode ser aprimorado com o uso de soluções de mineração de dados, como agrupamento e detecção de anomalias. No entanto, aplicá-los não é uma tarefa simples, pois os dados da NF-e contêm milhões de registros com campos ruidosos e documentos fora do padrão. Além desses desafios, é custoso extrair informações de textos curtos para identificar vestígios de má gestão, desfalque, fraude comercial ou evasão fiscal. Soluções eficientes para agrupamento de dados com características semelhantes às NF-es até onde sabemos não foram propostas na literatura. Este trabalho desenvolveu o ELINAC, um serviço para agrupamento de dados de texto curto em NF-es que utiliza um autoencoder para agrupar dados. O ELINAC auxilia na auditoria de transações documentadas em NF-e, agrupando dados semelhantes por descrições de texto curto e facilitando a detecção de anomalias em campos numéricos. Para isso, ELINAC explora como modelar o autoencoder sem aumentar muito os custos de cálculo para suprimir um grande número de dados de texto curto. Na pior das hipóteses, os resultados mostram que o ELINAC agrupa os dados de forma eficiente enquanto executa três vezes mais rápido do que as soluções adotadas na literatura.

Palavras-chave: Autoencoder, Agrupamento, Redes Neurais, Notas Fiscais Eletrônicas, Textos curtos

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Auditar como os recursos são gerenciados é um desafio que as agências reguladoras de todo o mundo sempre tiveram [1, 2]. Isso pode ser ampliado quando se considera um país com um sistema econômico complexo. Por exemplo, no Brasil, a quantidade de dados que precisa ser auditada é de grande magnitude. Com o aumento exponencial da informação digital produzida nas últimas décadas, os órgãos públicos buscam constantemente novas soluções que melhorem a eficiência e os resultados dos processos que lidam com *big data* [3, 4, 5, 6], tais como classificação, agrupamento e detecção de anomalias em transações monetárias. Regular as transações que os agentes públicos realizam com dinheiro do governo é uma das principais prioridades do processo de auditoria, em que a má gestão, o desfalque, a fraude e a corrupção podem ter um impacto devastador no desenvolvimento do país e de seus cidadãos [7, 8, 9, 10, 11]. Essas transações são documentadas em documentos fiscais, como notas fiscais eletrônicas.

Os documentos fiscais estão disponíveis em diversos formatos no Brasil, a saber: recibos, notas fiscais de serviços, notas fiscais de consumidores e nota fiscal eletrônica, que é o mais comum. A Nota Fiscal Eletrônica Brasileira, denominada NF-e, é um documento que registra as transações *Business to Business*. As NF-es são armazenadas como arquivos XML e divididas em vários campos a serem preenchidos. Desde 7 de outubro de 2021, 29,847 bilhões de NF-es foram emitidas por 2,013 milhões de entidades diferentes [12]. Nos últimos anos, as NF-es têm apresentado uma tendência de crescimento contínuo, em que a quantidade de NF-es emitidas de 2016 a 2018 aumentou em mais de 100 milhões a cada ano.

A emissão de NF-es exige a verificação do tipo das entradas, garantindo que as informações preenchidas correspondem ao campo correto. Mesmo após as validações iniciais, esses documentos ainda podem apresentar informações incorretas, tais como categoria de produto errada, pontos e vírgulas mal colocados para preços e erros ortográficos na descrição do produto. Portanto, soluções eficientes de mineração de dados para esses problemas

são essenciais para uma melhor auditoria das notas fiscais eletrônicas. Embora muitos métodos diferentes estejam disponíveis para dados de texto curto, os resultados não são ótimos quando aplicados a conjuntos de dados de NF-e. Os campos baseados na entrada de texto têm documentação muito ruidosa e fora do padrão. Além desses desafios, extrair informações de textos curtos para identificar vestígios de má gestão, desfalque, negócios que conduzem operações fraudulentas ou evasão fiscal é um processo custoso.

Lidar com esse tipo de dados não é simples, pois encontrar padrões significativos é uma tarefa complexa, mesmo após a limpeza, devido às suas características desbalanceadas. Além disso, os métodos padrões de mineração de dados ainda têm algumas limitações [13, 14, 15, 16]. Por exemplo, é difícil perceber variações nos campos numéricos quando os dados processados contêm registros sobre produtos diferentes. Isso ocorre principalmente porque as variações entre os valores de registro não podem fornecer um padrão confiável no qual uma divergência significativa possa ser notada [17, 18].

Estratégias que agrupam elementos baseados em suas descrições podem fornecer melhores informações sobre o produto para lidar com esses desafios. Com isso, variações significativas em um campo diferente, tais como preço e quantidade, podem ser melhor analisadas para *outliers* [19]. Algoritmos de agrupamento podem ser usados para este propósito [20, 21]. Para isso, um conjunto de dados é dividido em agrupamentos menores com base em uma similaridade observada, fornecendo dados mais fáceis de aplicar diferentes análises, como métodos de detecção de anomalias.

Vale ressaltar que um dos principais problemas quando se trata da maioria dos algoritmos de agrupamento é a complexidade de caso médio, que em alguns casos, como o algoritmo Hierárquico Aglomerativo, chega a $O(N^2)$ [22]. Por outro lado, soluções mais otimizadas também estão disponíveis, em que DBSCAN, que tem complexidade de caso médio $O(N \log_2 N)$ [23], pode agrupar dados mais rapidamente do que outros algoritmos convencionais. No entanto, mesmo considerando abordagens rápidas, uma vez que o valor de N pode ser muito grande, o tempo ainda pode ser um problema [24]. Com isso em mente, alternativas como as Redes Neurais (RNs) podem fornecer soluções promissoras.

Vários trabalhos na literatura atual visam melhorar os resultados de agrupamento [25, 26, 27, 28, 29, 30]. Muitos deles usam RNs para pré-processar os dados antes de alimentar os dados em um algoritmo de agrupamento [28, 29, 30], em que a maioria deles depende de algoritmos, como K-means, que não são eficientes em termos de tempo. Outros também implementam o aprendizado profundo diretamente no processo de agrupamento, pois fornece flexibilidade [25, 26, 27]. Ainda assim, eles estão mais focados em melhorar o agrupamento em vez de fornecer soluções que possam melhorar o desempenho em relação a grandes volumes de dados. Outros trabalhos utilizam o autoencoder como solução para detecção de anomalias [31, 32, 33]. No entanto, essas soluções complexas não se destinam

aos resultados de agrupamento mais rápidos para dados de texto curto.

1.1 Objetivos

Com o objetivo de fornecer uma nova solução no tratamento de dados de NF-e, foi proposto um serviço de agrupamento de dados para textos curtos em NF-es que utiliza um autoencoder para agrupar os dados, denominado ELINAC. O objetivo do ELINAC é auxiliar na auditoria de transações documentadas em NF-e, agrupando dados semelhantes por meio de descrições de texto curto e facilitando a detecção de anomalias em campos numéricos. Para isso, o ELINAC transforma os dados e cria *clusters* com base em um modelo de autoencoder. Ao fazer isso, o ELINAC gera resultados que os auditores podem usar para analisar NF-es ou dados semelhantes de maneira eficiente. Os resultados mostram que o ELINAC pode agrupar dados de forma eficiente enquanto executa três vezes mais rápido do que os algoritmos tradicionais no pior caso. Além disso, a transformação de dados fornece melhores resultados para o autoencoder e todos os outros algoritmos de agrupamento usados nos experimentos. Este trabalho possui as seguintes contribuições:

- Nosso trabalho agrupa dados de texto curto em um curto espaço de tempo baseado em uma rede de autoencoder;
- Contamos com a possibilidade de ajustar os parâmetros de agrupamento sem retreinamento;
- Nosso modelo de autoencoder implementado no ELINAC é mais rápido que os modelos tradicionais e hierárquicos; e
- Identificamos *outliers* com base nos valores de erro de reconstrução produzidos pelo nosso modelo.

1.2 Publicação

Os principais resultados obtidos deste trabalho foram publicados na revista *Applied Sciences*, Qualis A3, Fator de Impacto 2.679 e pode ser acessado em [34]:

- SCHULTE, Johannes P. et al. ELINAC: Autoencoder Approach for Electronic Invoices Data Clustering. Applied Sciences, v. 12, n. 6, p. 3008, 2022.

1.3 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 discute e analisa trabalhos relacionados ao que está sendo proposto no ELINAC. O Capítulo 3 apresenta o serviço proposto, descrevendo o conjunto de dados usado para experimentos, como os dados são transformados, o modelo de autoencoder usado e o algoritmo usado para agrupamento. O Capítulo 4 apresenta os resultados e análises dos experimentos realizados para avaliar o ELINAC. Por fim, O Capítulo 5 apresenta uma retrospectiva do que foi alcançado com esta pesquisa e discute trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Este capítulo apresenta trabalhos que usam o autoencoder como uma solução de mineração de dados para identificar vestígios de má gestão, desfalque, negócios que conduzem operações fraudulentas ou evasão fiscal. Além disso, trabalhos que exploram autoencoders em conjunto com outros algoritmos de agrupamento serão apresentados para retratar o que vem sendo pesquisado na área. Vale a pena notar que, apesar de muitos estudos propondo soluções baseadas em autoencoder, até agora não encontramos uma solução que modele o autoencoder para dados de texto curto com uma abordagem eficiente em termos de tempo.

Técnicas de mineração de dados aplicadas em documentos fiscais, *eg* agrupamento, classificação e detecção de anomalias, são amplamente utilizadas para detecção de fraudes em transações [31, 35, 36, 37] e operações financeiras em geral [38, 39, 40]. No trabalho de Paula et al. [31], foi proposto um modelo para detecção de fraudes em documentos fiscais brasileiros, como a nota fiscal eletrônica. O modelo foi baseado em um autoencoder profundo responsável por gerar o erro de reconstrução utilizado para a detecção de fraudes. O trabalho visa detectar casos reais de fraude nos dados com base na saída do modelo. Kieckbusch et al. [41] desenvolveram um sistema de classificação de notas fiscais eletrônicas baseado em Redes Neurais Convolucionais (CNN). O treinamento da CNN é baseado em uma descrição de texto curto do produto e no código de categoria do produto (NCM) usado como rótulo. O objetivo é melhorar o processamento das faturas, identificando o código de produto correto para as faturas. Ambas as soluções não tratam de dados desequilibrados e distintos em notas fiscais. Além disso, embora o modelo proposto no [41] tenha um bom desempenho para sua tarefa, o NCM é um código geral que não é adequado para análise de grupos mais específicos. Técnicas de agrupamento também foram usadas para faturas eletrônicas no trabalho de Tang et al. [42]. Neste caso, um método de análise de fusão profunda é proposto com base em K-means e *skip-gram*, onde faturas eletrônicas anormais foram usadas para análise de associação entre empresas e

usuários. Embora os estudos acima mencionados tenham tratado de dados de faturas eletrônicas, nenhum deles buscou uma análise mais específica produto a produto. Além disso, os trabalhos exploraram seus resultados sobre o tema da análise e não sobre o desempenho do método proposto.

Outras frentes de pesquisa investigam soluções de agrupamento com autoencoders [43, 25, 26, 27]. Yang et al. [25] apresentaram um *framework* de agrupamento espectral profundo baseado em uma rede *dual autoencoder*. Esta rede cria dados mais robustos ao ruído em sua representação latente enquanto usa estimativa de informações mútuas para informações mais específicas. O trabalho proposto visa melhorar os resultados de agrupamento de abordagens atuais do estado da arte. Um agrupamento profundo com um método de autoencoder variacional foi proposto no trabalho de Lim et al. [26]. O estudo é baseado em um autoencoder modificado, onde cada ponto presente no espaço latente é realinhado artificialmente para a classe do vizinho mais próximo durante o treinamento. Esta abordagem visa melhorar os resultados de estudos anteriores utilizando a mesma técnica. Outra abordagem de agrupamento através do autoencoder é apresentada por Mrabah et al. [27]. O estudo aborda o problema de capturar variações significativas no aprendizado não supervisionado, propondo um autoencoder dinâmico. Seguindo esta abordagem, o método visa alcançar resultados do estado da arte. Esses trabalhos validam a ideia de usar autoencoders para agrupamento, mas nenhum deles foi proposto para dados que apresentem textos curtos, como ocorre nas NF-es. Além disso, o tempo de agrupamento é um fator essencial que na maioria dos estudos não recebeu tanta atenção quanto explorada nesta pesquisa.

Outros trabalhos tratam apenas do problema de categorização de textos curtos [15, 16]. Enamoto et al. [16] desenvolveu uma estrutura genérica para categorização de textos curtos multilíngue baseada na CNN, intitulada GM-ShorT. Os resultados mostram que o GM-ShorT pode ser usado eficientemente para categorização de textos curtos multilíngues. Em Schmitz et al. [15], foi proposta uma solução para categorizar textos curtos derivados de múltiplas fontes de informação para retratar a situação atual do mercado financeiro, denominado GOOSE. O GOOSE foi modelado com base em um Bi-LSTM e GloVe Embeddings para auxiliar na confiabilidade na classificação de textos curtos. No entanto, os trabalhos citados focam apenas na categorização de textos curtos, incapazes de agrupar dados de textos curtos. Em outras palavras, os trabalhos não exploram o uso do autoencoder para agrupamento de dados como o ELINAC.

Autoencoders também podem ser usados em conjunto com outros métodos. Muitos estudos exploraram o uso de soluções de aprendizado profundo para pré-processamento de dados em agrupamento e detecção de anomalias [28, 29, 30]. Yang et al. [28] utiliza técnicas de aprendizado profundo para pré-processamento de dados para criar representações

mais adequadas para K-means. O método proposto valida o uso de redes neurais profundas para uma melhor representação dos dados para agrupamento e valida os resultados usando conjuntos de dados de texto e imagem. O objetivo é melhorar os resultados de agrupamento e recuperar uma representação latente dos dados mais amigável ao processo de agrupamento. Embora utilize dados de texto, baseia-se principalmente na extração de características de palavras que não são adequada para dados de NF-e. Além disso, como ele depende de K-means para o agrupamento final, o modelo não é orientado ao desempenho.

Uma abordagem diferente para redes neurais profundas e K-means é apresentada por Fard et al. [29]. O problema de gerar representações de aprendizagem adequadas para algoritmos de agrupamento é estudado para melhor desempenho de agrupamento. As redes de autoencoder são utilizadas para melhor representação de dados de alta dimensão usando apenas atualizações de gradiente para aprendizado. Este estudo também demonstrou melhores resultados ao usar o aprendizado profundo antes de outras técnicas de agrupamento em todos os conjuntos de dados, mas não explora o uso de agrupamento com eficiência de tempo.

Kim et al. [30] apresentam uma solução baseada em autoencoder e K-means. Para isso, o autoencoder é utilizado para a redução da dimensionalidade dos dados, que são então agrupados usando K-means. Os dados resultantes são usados para um modelo de aprendizado profundo para detectar anomalias. Enquanto o trabalho apresenta uma solução utilizando autoencoder e agrupamento para melhor detecção de anomalias, a agrupamento depende do algoritmo K-means, que não é eficiente em termos de tempo, diferente da técnica apresentada nesta pesquisa.

Estudos anteriores em agrupamento e detecção de anomalias conduziram extensas pesquisas apresentando resultados interessantes para vários tipos de dados. Os estudos apresentados formam uma base sólida para nosso trabalho, apresentando soluções bem-sucedidas para análise de documentos fiscais, autoencoders para agrupamento e pré-processamento de dados com autoencoders para melhor representação de dados para outros algoritmos de agrupamento. No entanto, deve-se notar que até o momento não foram encontradas soluções baseadas em autoencoders para agrupar dados de texto curto como a descrição apresentada nas NF-es. Além disso, os autoencoders têm sido usados principalmente para agrupamento de resultados em otimização e não em abordagens de eficiência de tempo, como será apresentado a seguir.

Capítulo 3

Uma Abordagem para Agrupar Dados de NF-e com base em Autoencoder

Este Capítulo apresenta o ELINAC[1], um serviço para agrupamento de dados de texto curto em NF-es que usa um autoencoder para resultados mais rápidos. O objetivo do ELINAC é auxiliar na auditoria de transações documentadas em NF-e, agrupando dados semelhantes por descrições de texto curto e facilitando a detecção de anomalias em campos numéricos. Para isso, o ELINAC transforma os dados e cria *clusters* com base em um modelo de autoencoder. Portanto, o ELINAC gera resultados que os auditores podem usar para analisar NF-es ou dados semelhantes de maneira eficiente em termos de tempo.

Para melhor compreensão do serviço proposto, a Figura 3.1 apresenta uma visão geral de como o ELINAC funciona, desde a emissão da NF-e até a auditoria. Após a extração dos dados de uma agência reguladora, o conjunto de dados da NF-e passa pelo processo de remoção de informações desnecessárias do campo de descrição (Rótulo A, Figura 3.1), que é preenchido com entrada humana e contém uma quantidade significativa de ruído e é muito desbalanceado [44, 45, 46, 47]. Os dados agora são mais adequados para a próxima etapa (Rótulo B, Figura 3.1), em que o treinamento do autoencoder gera um valor de reconstrução com base em quão semelhante a descrição de texto curto é aos recursos aprendidos. A saída de erro de reconstrução do autoencoder tem variações significativas para representar dados diferentes, e essa variação é o limite que define um novo *cluster*.

[1] `https://github.com/johpetsc/ELINAC`

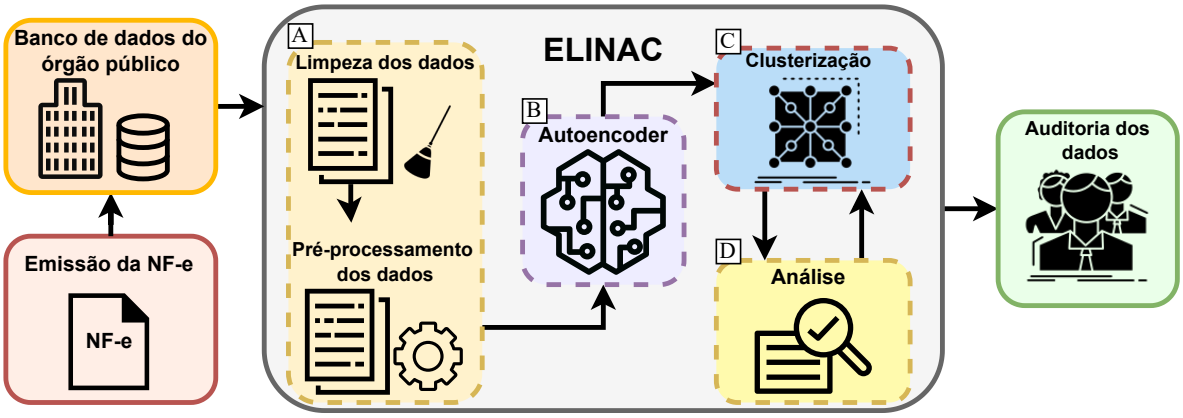


Figura 3.1: Cenário operacional do ELINAC.

A fase de agrupamento do ELINAC (Rótulo C, Figura 3.1) é realizada por um algoritmo que busca o valor limite ideal que representa as variações necessárias para o número de agrupamentos solicitados como parâmetro. O agrupamento é realizado em conjunto com a fase de análise (Rótulo D, Figura 3.1), pois pode ser repetido mais vezes dependendo dos agrupamentos gerados. Uma das vantagens do ELINAC é que quando o número de *clusters* é desconhecido, o agrupamento pode ser realizado várias vezes em um curto espaço de tempo, pois os dados não precisam ser treinados novamente no modelo de autoencoder. Dessa forma, se a análise apresentar *clusters* muito específicos ou muito amplos, o algoritmo de agrupamento pode receber um número diferente de *clusters* e gerar o novo resultado quase que imediatamente. Cada etapa no ELINAC é essencial para resultados adequados: o autoencoder não pode gerar valores de erro de reconstrução semelhantes se os dados não forem padronizados corretamente, e se o autoencoder não produzir valores com variância perceptível, a fase de agrupamento e análise não gerará agrupamentos corretos.

3.1 Descrição do dataset

O conjunto de dados utilizado para esta pesquisa foi fornecido pelo Tribunal de Contas da União (TCU - Brasil), contendo todas as NF-es emitidas no estado da Paraíba, entre março e julho de 2020, com mais de 929 mil registros. A NF-e foi implementada para substituir as notas fiscais físicas e dar validade judicial nas transações para o fisco. O conjunto de dados utilizado contém informações do banco de dados do Tribunal protegido pela Lei Geral de Proteção de Dados Pessoais (LGPD) do Brasil. A LGPD fornece orientações sobre como coletar e proteger dados pessoais no Brasil.

Tabela 3.1: Amostra do conjunto de dados usado no ELINAC. Legenda: **NCM** - Identificador de Categoria de Produto; **DESC** - Descrição do Produto, **QT** - Quantidade Comprada, **VAL** - Preço, **TIP** - Medida do Produto (por exemplo, unidade, caixa ou galão), **DATE** - Data de compra.

NCM	**DESC**	**QT**	**VAL**	TIP	**DATA**	**MUNICÍPIO**
30049099	#$ACETO.TRIA+SULFNEO+GRAM+NIST POM G LE	1	20	UN	2020-07-07	Nova Cruz
30049069	ZYXEM GTS 5MG20ML UCB Lote: 19I19 Vcto: 08/21	1	58.72	CX	2020-06-30	BELEM
30059090	ALGODAO HIDROFILO NEVOA 500GR	3000	8.35	RL	2020-04-23	MOSSORO
30049099	DICLOFENACO POTASSIO 50MG GEO GEOLAB	6000	0.07	CP	2020-05-14	Campina Grande
90183929	SCALP CANULA 21G C/DISP. SEG. WILTEX	15000	0.4	UN	2020-04-22	MOSSORO

A Tabela 3.1 apresenta uma amostra do conjunto de dados usado no ELINAC. Cada linha do conjunto de dados contém informações sobre uma compra realizada por um município do estado da Paraíba. As colunas consistem em identificador de categoria de produto (NCM), descrição do produto (DESC), preço (VAL), quantidade comprada (QT), data de compra (DATA), medida do produto (unidade, caixa, galão, etc.) (TIP), e o município que operou (MUNICÍPIO). O recurso DESC tem a descrição do produto que é usada para agrupamento. Consiste em poucas palavras e valores relacionados ao produto e não forma uma frase significativa. Como o ELINAC é baseado na similaridade de texto, a limpeza dos dados é essencial. Embora a remoção de caracteres especiais ou capitalização possa afetar os *clusters* resultantes, ainda há informações irrelevantes para o agrupamento, como o fabricante do produto ou abreviações desnecessárias. Por isso o ELINAC realiza um pré-processamento que filtra os dados para padronizar o texto.

3.2 Extraindo informações da NF-es

A característica DESC da NF-e tem o seguinte padrão: a primeira ou as duas primeiras palavras referem-se ao nome do produto, seguidas de alguma descrição extra (por exemplo, se o medicamento está em forma de gotas ou pomada, quantos comprimidos ou miligramas), e por fim, o fabricante e algumas meta-informações como o lote do produto (Tabela 3.1). Para nossa proposta, as únicas informações relevantes que precisamos utilizar são o nome do produto e os valores numéricos da descrição. Isto porque são as informações que mais influenciam nas variações em outros campos da NF-e. Para isso, desenvolvemos um mecanismo de filtro *ad hoc* que recebe a descrição original e retorna apenas as informações que ajudam a agrupar os mesmos produtos. Tal processo é descrito a seguir:

- Limpa os dados removendo caracteres especiais, capitalizando o texto e filtrando palavras irrelevantes para remover artigos e preposições.
- Procura a primeira palavra na *string*, ignorando todos os caracteres anteriores.
- Armazena a primeira palavra e pesquisa se outra palavra segue: caso negativo, a primeira palavra é o nome do produto; caso positivo, armazena-o junto com a primeira palavra, e ambos se tornam o nome do produto.
- Remove todos os caracteres do restante da *string* que não sejam um número ou o nome do produto.

Embora manter todos os números após o nome do produto ainda possa conter ruídos, esses valores são fundamentais para identificar anomalias nos valores da fatura no agrupamento formado. Por exemplo, o produto A com caixa de 30 não deve ser agrupado com o produto A com caixa de 60 porque o preço seria mais alto e, portanto, seria rotulado como uma anomalia. Além disso, as metainformações como o número do lote e a data de validade foram removidas na etapa de limpeza, portanto, os caracteres numéricos devem se referir ao produto.

Tabela 3.2: Extraindo informações após usar o mecanismo de filtro em uma amostra de dados. *Dados Originais* mostra um medicamento (VIMOVO) com sua dosagem correspondente (500+20), quantidade de comprimidos (20) e vários fabricantes. *Dados Novos* mostra o texto correspondente apenas com as informações relevantes sobre o medicamento e as quantidades.

Dados Originais	**Dados Novos**
VIMOVO 500 20MG 20CPR ASTRAZENECA	VIMOVO 500 20 20
VIMOVO 500 20MG CX 20 COMP REV	VIMOVO 500 20 20
VIMOVO 500+20 MG 20COMP	VIMOVO 500 20 20

A Tabela 3.2 apresenta a descrição DESC da NF-e ao usar o mecanismo de filtro. Observa-se que tal mecanismo retorna apenas as informações que ajudam a agrupar os mesmos produtos. Por exemplo, na Tabela 3.2, três DESCs diferentes são apresentados, em que o mesmo produto tinha fabricantes diferentes, e sua dosagem e quantidade de comprimidos foram escritas de formas diferentes. Após utilizar o mecanismo proposto, obtemos todas as informações que descrevem o produto em si, descartando todo o resto. Essa técnica torna o campo de descrição mais padronizado e mais adequado para agrupamento, especialmente para a abordagem de autoencoder, que é baseada principalmente na similaridade de texto.

Figura 3.2: *Word cloud* dos dados extraídos.

Avaliar os resultados do agrupamento não é uma tarefa trivial, especialmente ao usar dados não rotulados. Desta forma, foi realizada uma rotulação manual de uma amostra da base de dados, a qual será utilizada para calcular as métricas de agrupamento durante os experimentos. Para isso, utilizamos o campo NCM para filtrar os dados com mais ocorrências, "30049099", composto por medicamentos, e campo TIP igual a "UN", significando que o produto adquirido é unitário. A base de dados rotulada possui cerca de 21 mil registros. A nuvem de palavras ilustrada na Figura 3.2 apresenta as palavras mais comuns nos dados rotulados, em que a maioria se refere ao nome ou tipo de medicamento.

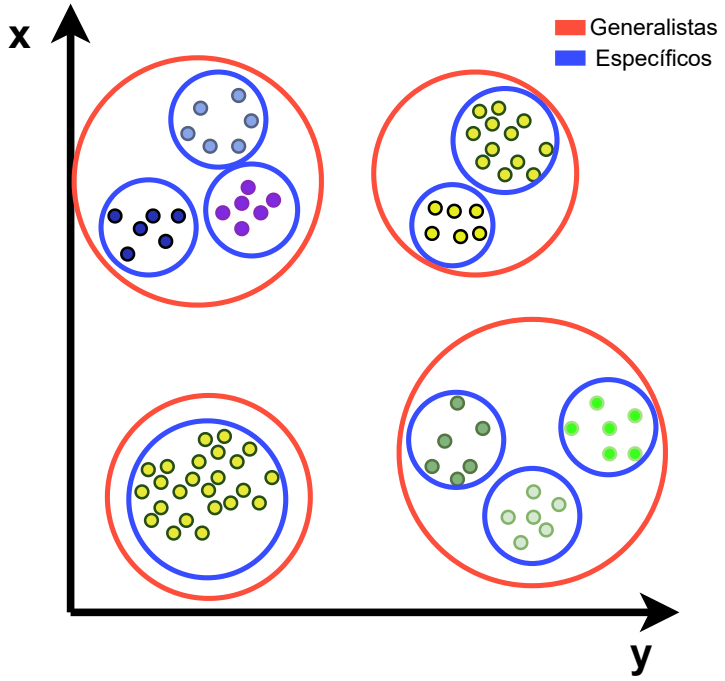


Figura 3.3: Representação dos rótulos utilizados para *clusters*.

Foram atribuídos dois rótulos distintos para cada amostra do conjunto de dados rotulados, representando grupos mais específicos e mais generalistas. A Figura 3.3 apresenta uma representação visual de como os rótulos foram atribuídos. O rótulo específico descreve medicamentos com a mesma descrição, ou seja, possuem o mesmo nome, dosagem, quantidade de comprimidos e outros. Esta rotulação resultou em 3.173 *clusters* diferentes, o que significa que o tamanho médio do *cluster* é próximo de 7. Por outro lado, o rótulo generalista ignora informações adicionais relacionadas ao medicamento, ou seja, agrupa-os apenas pelo nome, resultando em 1564 *clusters* com tamanho médio de 13.

A rotulação do o conjunto de dados com essa abordagem foi realizado principalmente por dois motivos: o rótulo específico testa a capacidade do método de agrupar por similaridade de texto, uma vez que as descrições são próximas de serem as mesmas, principalmente após o pré-processamento dos dados. O rótulo generalista tem como objetivo testar se uma abordagem de autoencoder para agrupamento pode generalizar os dados e não depender apenas da similaridade literal do texto.

3.3 Autoencoder utilizado no ELINAC

Autoencoders são RNs que podem ser utilizadas para detectar anomalias devido à sua capacidade de redução de dimensionalidade para dados multidimensionais [48]. No ELINAC, o autoencoder modelado segue a arquitetura padrão (Figura 3.4), que consiste em camadas ocultas com menos nós que a camada de entrada, e suas saídas são representações reduzidas dos dados de entrada. A rede é simétrica e pode ser dividida em dois [48]: o codificador, que aprende a função de codificação ϕ, e o decodificador, a função de decodificação ψ.

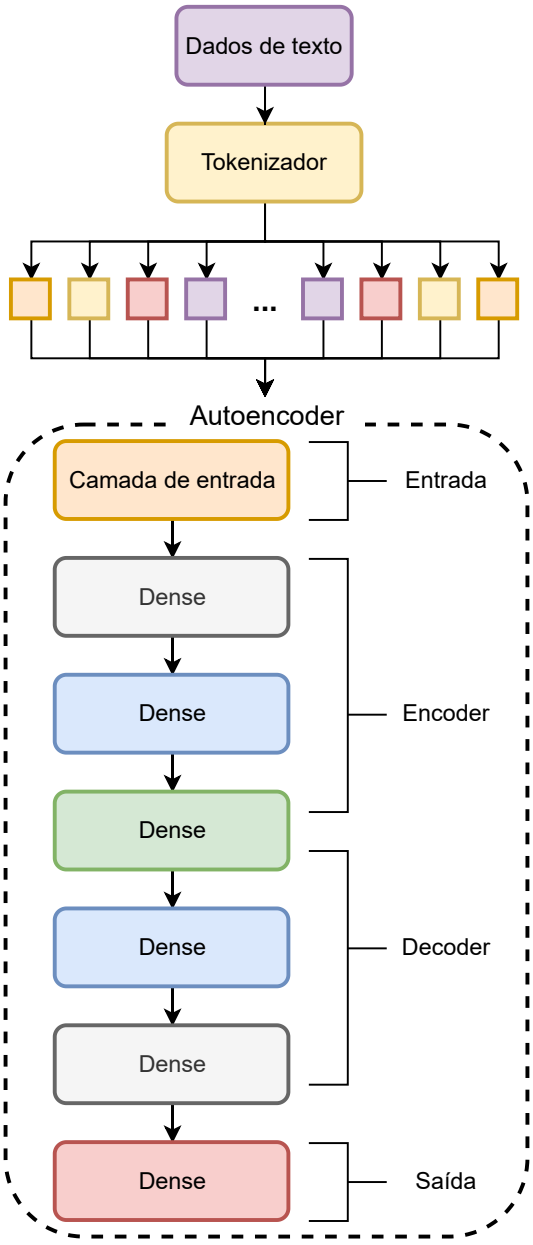


Figura 3.4: Modelo autoencoder e entrada de dados.

O autoencoder modelado funciona da seguinte forma: os dados de entrada D são aplicados à função codificadora $\phi(D)$ e, após passarem pela redução de dimensionalidade, os dados resultantes são aplicados à função decodificadora $D' = (\ phi \circ \psi)(D)$, que gera os dados reconstruídos [48]. Quando camadas ocultas suficientes são utilizadas, os dados de entrada podem ser representados em uma forma de baixa dimensão. O erro de reconstrução é então calculado por $D - D'$ [48].

A definição usada para o autoencoder do ELINAC mostra como um *cluster* pode ser formado usando o mesmo princípio de como uma anomalia pode ser detectada. Ou seja, no ELINAC, a rede comprime os dados de entrada e depois os reconstrói na função

decodificadora, em que o resultado compreende os dados de entrada reconstruídos. Ao fazer isso, o ELINAC retorna um valor que mostra a diferença entre um registro de dados e o conjunto de dados, fornecendo um valor que pode ser usado para agrupamento. Em outras palavras, os valores de saída do ELINAC representam o quão semelhante os dados são comparados ao que o autoencoder aprendeu com a entrada.

Conforme apresentada na Figura 3.4, a entrada do autoencoder é criada tokenizando o campo de descrição, dividindo o texto na mesma quantidade de *tokens* que os nós da camada de entrada. Nossa arquitetura de modelo de autoencoder é simétrica e consiste em um codificador e um decodificador com três camadas ocultas cada, onde a camada de entrada é composta por 16 nós e a camada de gargalo é composta por 4 nós. As camadas da rede são ativadas pela função *relu*, e o modelo é compilado usando o otimizador *Adam* [49].

3.4 Clusterização do ELINAC

Após o autoencoder processar os dados de entrada no ELINAC, temos nossos dados de saída, onde cada amostra com uma descrição possui um valor correspondente ao seu erro de reconstrução. O próximo passo é aplicar a hipótese de que descrições semelhantes já estão agrupadas por ter erros de reconstrução semelhantes, e tudo o que resta a fazer é encontrar esses agrupamentos e rotulá-los.

A Figura 3.5 apresenta como o erro de reconstrução aumenta para as primeiras 150 amostras classificando o valor de reconstrução do menor para o maior valor. Observa-se que os dados se apresentam em passos. Alguns deles são longos, o que significa que existem muitas descrições semelhantes, enquanto outros são curtos, mas também muito altos quando apenas algumas descrições semelhantes diferem mais das outras. Para ser considerado o início de um novo *cluster*, o aumento do erro de reconstrução entre duas linhas deve ser maior que um valor limite usado para comparação entre cada linha.

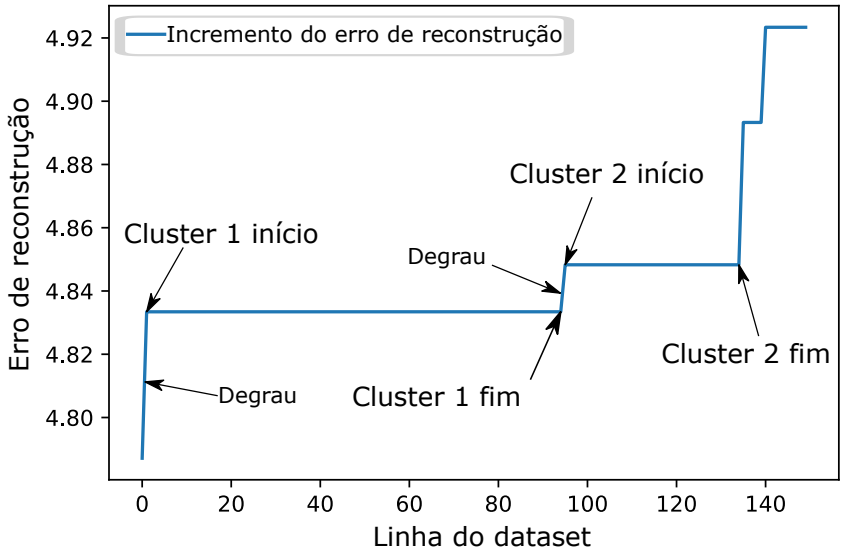


Figura 3.5: Visualização dos *clusters* em uma amostra ordenada pelo erro de reconstrução.

O processo usado para pesquisar e rotular os *clusters* com base na saída do autoencoder no ELINAC é descrito no Algoritmo 1. Tendo em mente que o tempo eficiente é um dos fatores desejáveis para nosso agrupamento, nosso algoritmo foi baseado no algoritmo de busca binária devido à sua complexidade de caso médio de $O(\log N)$, adaptado para buscar um valor de *threshold* em um intervalo de possíveis valores. A complexidade logarítmica é aplicada na faixa de limite possível T, significando que o *loop* principal tem uma complexidade de $O(\log T)$. Para cada iteração, o algoritmo percorre todas as amostras do conjunto de dados (linhas 9 a 17), adicionando uma complexidade $O(N)$ extra para cada repetição. Portanto, a complexidade final do nosso algoritmo chega a $O(I{*}N)$, onde I representa as iterações do *loop while* (número médio de iterações ao aplicar $O(\log T)$), e N é o tamanho do *dataframe*. O valor médio de I em nossos experimentos foi de $10,5$. Mesmo ao alterar os valores de *den* (valor do denominador usado para pesquisar o limite que varia de 100 a 10.000), o número de *clusters* e o tamanho do conjunto de dados nunca ultrapassaram 20 iterações. Portanto, o tamanho de I não escala com o valor de N, o que tornaria a complexidade final do algoritmo $O(N)$.

Algorithm 1 Algoritmo para agrupamento de NF-es

Require: $target \geq 0$ ▷ Quantidade alvo de*clusters*
Ensure: $df \neq NULL$ ▷ *Dataframe*
1: $den \leftarrow 4000$ ▷ Denominador para o threshold
2: $flag \leftarrow 0$ ▷ *Flag* de parada
3: $clusters \leftarrow 0$ ▷ *Clusters* encontrados
4: $max \leftarrow 0$ ▷ Denominador máximo
5: $min \leftarrow 0$ ▷ Denominador mínimo
6: **while** $clusters \neq target$ **and** $den \neq flag$ **do**
7: $clusters \leftarrow 0$
8: $flag \leftarrow den$
9: **for** row **in** df **do**
10: $row.shifted \leftarrow row.error.shift(1)$
11: **if** $row.error \geq row.shifted * 1 + (1/den)$ **then**
12: $row.cluster \leftarrow 1$
13: $clusters \leftarrow clusters + 1$
14: **else**
15: $row.cluster \leftarrow 0$
16: **end if**
17: **end for**
18: **if** $clusters > target$ **then**
19: $max \leftarrow den$
20: $den \leftarrow (max + min)/2$
21: **else if** $max \neq 0$ **then**
22: $min \leftarrow den$
23: $den \leftarrow (max + min)/2$
24: **else**
25: $den \leftarrow den * 2$
26: **end if**
27: **end while**

A principal diferença entre o algoritmo de busca binária e o ELINAC é que não temos o valor inicial e final usados na busca de uma lista. Para resolver este problema, o Algoritmo 1 inicializa com um valor denominador (*den*) (linha 1), que é o valor usado para encontrar o limite. Com base no valor inicial do denominador e no mínimo (*min*) como 0, o algoritmo executa sua primeira iteração para verificar se o número de *clusters* encontrados é maior ou menor que nosso alvo (linha 18). Se maior, atribui o máximo inicial como denominador na linha 19 e, se menor, dobra o valor do denominador na linha

25 até que a condição na linha 18 passe e o valor máximo (max) seja atualizado na linha 19. Agora que conhecemos os limites máximo e mínimo para o nosso denominador, o algoritmo procura o meio. Se for maior, atualize o limite mínimo (linha 22) e o valor médio (linha 23); se for menor, atualize o limite máximo (linha 19) e o valor médio (linha 20). Em seguida, repete o *loop* na linha 6 até encontrar o limite correto.

Tabela 3.3: Valor médio de iterações para a variável de denominador.

Valor de denominador	**Iterações**
100	16.7
2000	12.2
4000	10.5
6000	12.6
8000	13.0
10000	13.4

O limite calculado na linha 11 do Algoritmo 1 é um número muito baixo, na maioria das vezes próximo a $0,00025$, razão pela qual o algoritmo aumenta ou diminui o denominador para seu cálculo ($1/den$), e o valor inicial de den é 4000. Esses valores foram escolhidos com base no número médio de iterações necessárias para encontrar o valor limite, onde 4000 teve o melhor resultado, que é mostrado na Tabela 3.3. Em seguida, comparamos o erro de reconstrução de cada linha com o seguinte. Se o próximo valor for maior que o valor atual vezes o limite, é marcado como um novo início de *cluster*. O agrupamento é concluída se o número de *clusters* atribuídos for igual ao nosso destino. O algoritmo usa uma variável de *flag* de parada que interrompe o *loop* quando o número alvo de *clusters* não pode ser alcançado devido à falta de variação nos valores de saída ou dados menores do que a quantidade desejada de *clusters*. Por fim, é necessário enumerar cada *cluster*. Para isso, o algoritmo passa por todas as linhas atribuindo um rótulo. Sempre que $row.cluster$ é marcado como 1, o valor do *cluster* aumenta.

Em termos matemáticos, o algoritmo recebe um *array* R de tamanho N (tamanho do conjunto de dados), que consiste nos valores para o erro de reconstrução R_0, R_1,...,R_{N-1} atribuído a cada descrição de texto curto, onde para cada elemento i no *array*, $R_i \leq R_{i+1}$. O valor de destino T_a é o número de *clusters* C_0, C_1,...,C_{Ta-1} sendo pesquisados e T_i o número de *clusters* encontrados em cada iteração. Existe um valor de *threshold* T_h calculado por $1 + (1/den)$ que o número de pares onde $\forall x \in N, R_x \geq R_{x-1} + T_h$ é verdadeiro é igual a T_a. Para procurar T_h, assumimos um *array* V de possíveis valores racionais $1 \geq T_h \geq 2$, sabendo que valores menores que 1 não são possíveis, e valores

maiores que 2 são muito divergentes para serem considerados um *cluster*. O algoritmo de busca binária é aplicado a V, para buscar por T_h em seu intervalo de valores possíveis. Para começar, o valor de *den* é definido como 4000, que é o valor que, em média, produz o menor número de iterações por meio de V. Os valores máximos de V_{max} e mínimos de V_{min} são iniciados como 0 e são atualizados de acordo com a seguinte função:

$$f(V_{max}, V_{min}, T_i, T_a) = \begin{cases} V_{max} = den, (V_{max} + V_{min})/2 & \text{se } T_i > T_a \\ V_{min} = den, (V_{max} + V_{min})/2 & \text{se } T_i \leq T_a \text{ e } V_{max} \neq 0 \\ den = den * 2 & \text{caso contrário} \end{cases}$$

Capítulo 4

Avaliação de desempenho

Este capítulo apresenta a metodologia utilizada para validar o ELINAC. Para isso, o ELINAC foi avaliado em duas etapas. Na primeira etapa, é avaliada a capacidade do ELINAC em agrupar dados de texto curto, apresentada na Seção 4.2. Em seguida, na segunda etapa, é realizada a avaliação de desempenho do ELINAC para agrupar dados de texto curto, comparando-o com outras abordagens utilizadas na literatura, como apresentada na Seção 4.3.

4.1 Configuração do cenário

Os experimentos foram realizados em uma CPU i7-4770k (*base clock*), GPU GTX 1660 6GB e 16GB de RAM rodando Python versão 3.7.9 [50]. O ELINAC foi implementado usando o *framework* Tensorflow [51] em conjunto com a biblioteca scikit-learn [52]. Para validar o ELINAC, os seguintes algoritmos da literatura foram utilizados para comparação: (i) K-means [53, 54]; (ii) Agrupamento Hierárquico [55]; e (iii) DBSCAN [23]. A Tabela 4.1 apresenta o conjunto de parâmetros utilizados nos algoritmos.

Tabela 4.1: Parâmetros utilizados para cada método.

Método	**Parâmetros**
K-means	$n\_init = 2$, $max\_iter = 5$
Hierárquico	$linkage = average$, $affinity = cosine$
DBSCAN	$min\_samples = 1$ e $eps = 0.513/0.968$
ELINAC	$optimizer = Adam(0.001)$, $loss = mse$, $activation = relu$

Como o objetivo é avaliar a eficiência do ELINAC no agrupamento, foram utilizadas as seguintes medidas de desempenho:

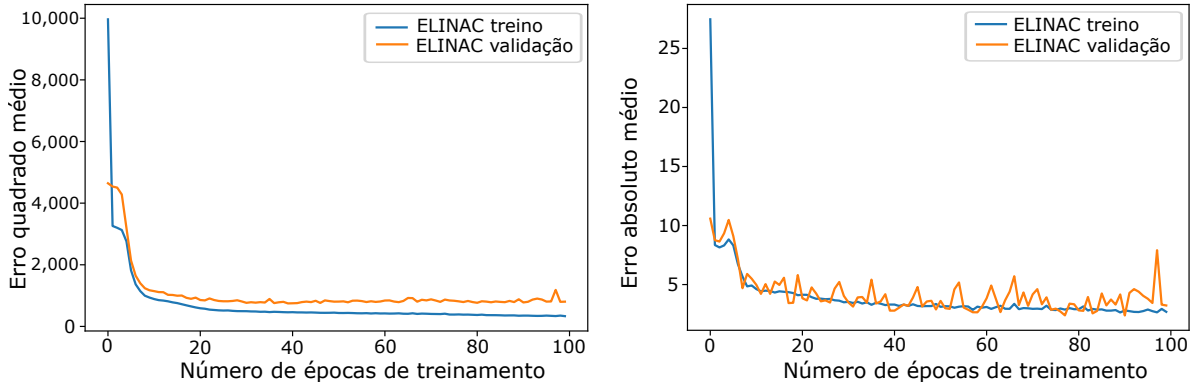
- *Tempo para agrupamento*: Mede a eficiência do método de agrupamento, começando quando a entrada de dados é alimentada a um método de agrupamento e terminando quando todos os *clusters* são atribuídos aos dados de entrada.
- *Homogeneidade*: Avalia se cada *cluster* contém apenas membros de uma única classe.
- *Completude*: Avalia se todos os membros de uma determinada classe estão atribuídos ao mesmo *cluster*.
- *Informação Mútua*: Mede a concordância de duas atribuições, ignorando permutações.
- 2×2 *Matriz de Confusão por Pares*: A matriz compara dois resultados de agrupamento, considerando todas as amostras e contando pares atribuídos ao mesmo agrupamento ou em agrupamentos diferentes. Os elementos da matriz contêm informações sobre quatro tipos de pares, C_{00} (verdadeiros negativos), C_{10} (falsos positivos), C_{01} (falsos negativos) e C_{11} (Verdadeiros Positivos).
- *Fowlkes-Mallows*: Calcula a pontuação de Fowlkes-Mallows FMI, definida como a média geométrica da precisão e *recall* dos pares. Sua fórmula é apresentada na Equação (4.1), em que TP é o número de Verdadeiros Positivos, FP é o número de Falsos Positivos e FN é o número de Falsos Negativos.

$$FMI = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}} \tag{4.1}$$

4.2 Avaliação de desempenho do ELINAC no agrupamento de dados de texto curto

A rede autoencoder do ELINAC foi treinada utilizando uma divisão de treinamento 80/20 para verificar as curvas de treino e validação. Embora o número de épocas para agrupamento possa variar, o modelo foi treinado por 100 épocas para avaliar a progressão das métricas de treinamento para nosso conjunto de dados de NF-e. O número de épocas necessárias para o agrupamento ideal é discutido mais adiante nesta seção. As camadas ocultas da rede usam a função de ativação do retificador (ou seja, *relu*), que aplica a unidade linear retificada à entrada do nó. O modelo é otimizado pelo algoritmo *Adam* [49], um método estocástico de gradiente descendente baseado na estimativa adaptativa de momentos de primeira e segunda ordem que é computacionalmente eficiente e adequado para problemas com grandes dados [49], com uma taxa de aprendizado padrão de $0,001$.

A primeira etapa para validar o ELINAC é avaliar seu processo de treinamento usando métricas convencionais de autoencoder e, em seguida, compará-lo com o comportamento das métricas de *cluster* ao treinar os mesmos dados. A Figura 4.1 apresenta os valores para o Erro Médio Absoluto (MAE, definido na Equação (4.2)) e o Erro Médio Quadrado (MSE, definido na Equação (4.3)) ao longo de épocas. Os resultados apresentam um comportamento típico para treinamento de autoencoder, onde a curva começa com uma queda acentuada até retificar e convergir. Isso significa que o ELINAC possui um comportamento de treinamento convencional para os dados utilizados para esta pesquisa.



(a) MSE.

(b) MAE.

Figura 4.1: Avaliação das curvas de treinamento do ELINAC para MSE e MAE.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}| \tag{4.2}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 \tag{4.3}$$

O erro de reconstrução resultante também avalia o processo de agrupamento de dados de texto curto do ELINAC para identificar valores discrepantes. A Figura 4.2 apresenta o aumento do valor do erro de reconstrução em cada amostra do conjunto de dados quando ordenado do menor para o maior. Em uma abordagem convencional de detecção de *outliers*, os registros de aumento acentuado no final seriam rotulados como *outliers*. Portanto, o erro de reconstrução resultante segue o comportamento esperado para identificar *outliers*.

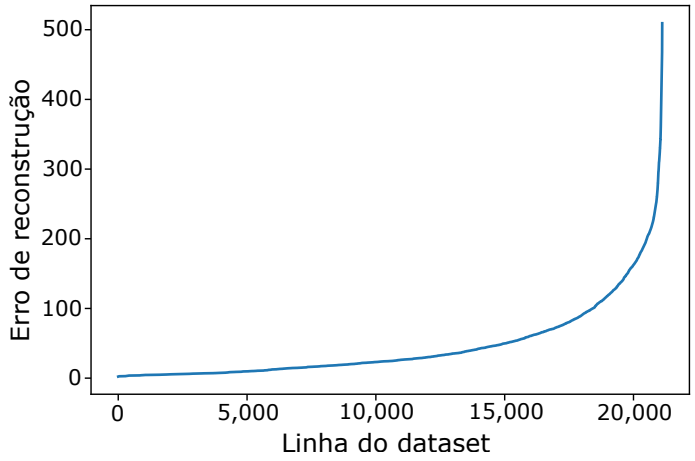
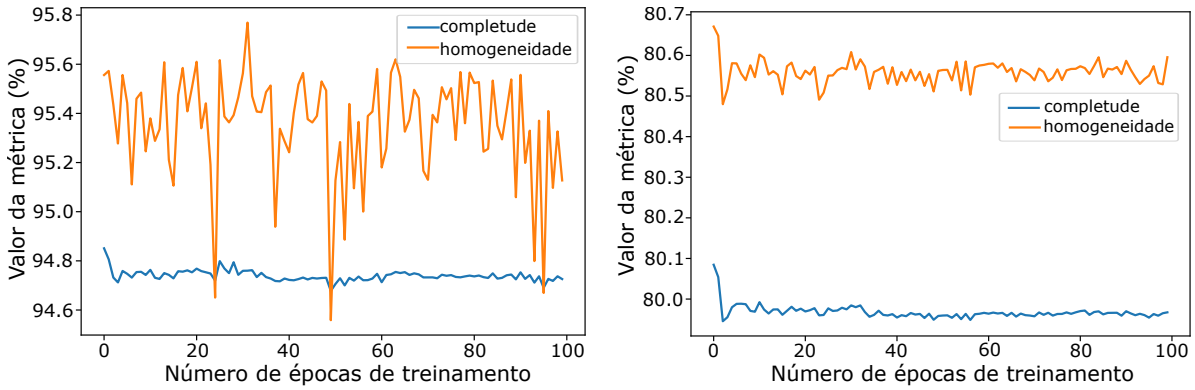


Figura 4.2: Distribuição dos erros de reconstrução ao ordenar o conjunto de dados do menor para o maior valor.

Os resultados de treinamento apresentados nas Figuras 4.1 e 4.2 mostram que o ELINAC possui uma curva de treinamento típica e produz o valor do erro de reconstrução com o mesmo padrão usado puramente para detecção de *outliers*. Isso faz sentido, pois o ELINAC foi modelado com base no autoencoder. A próxima etapa para validar o ELINAC é avaliar o comportamento das métricas de *cluster* ao treinar o modelo com os mesmos dados de entrada anteriores.

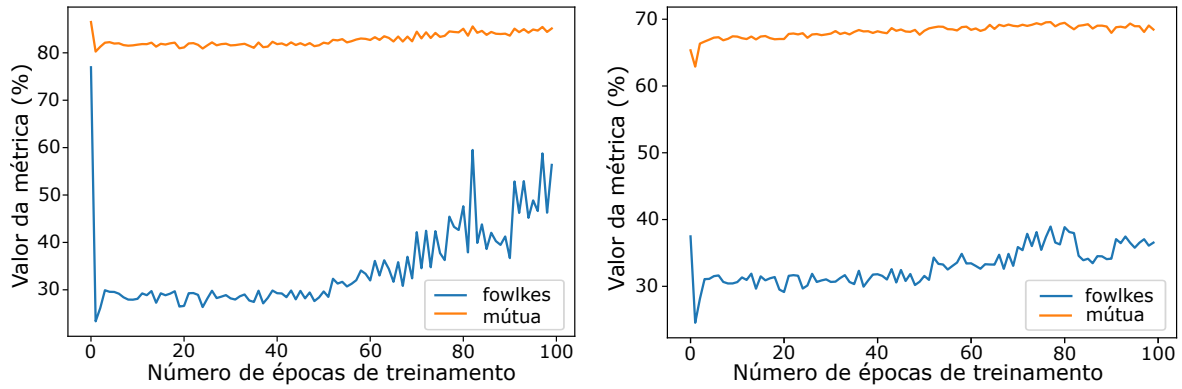
A Figura 4.3 apresenta a variação das métricas de Completude e Homogeneidade em relação ao treinamento de épocas, ilustrando uma análise comparativa com *clusters* específicos (Figura 4.3(a)) e *clusters* generalistas (Figura 4.3(b)). À medida que o número de épocas aumenta, observa-se que há estabilidade para a métrica de completude e dispersão para a métrica de homogeneidade, conforme apresentado na Figura 4.3(a). Isso ocorre porque *clusters* específicos têm um tamanho médio pequeno, o que significa que a homogeneidade de um *cluster* é mais propensa a alterações quando os registros são atribuídos incorretamente. Deve-se notar que para *clusters* generalistas (Figura 4.3(b)), há estabilidade na precisão independente da métrica utilizada. Isso faz sentido porque *clusters* maiores são menos propensos a variações.



(a) Clusters específicos. (b) Clusters generalistas.

Figura 4.3: Avaliação de desempenho do ELINAC para métricas Completude e Homogeneidade.

A variação das métricas Informação Mútua e Fowlkes Mallows em relação ao treinamento de épocas é mostrada na Figura 4.4, comparando o comportamento para *clusters* específicos (Figura 4.4(a)) e *clusters* generalistas (Figura 4.4(b)). A métrica de Informação Mútua apresenta resultados muito consistentes com um pequeno aumento no final, mas não o suficiente para justificar a perda em outras métricas. Para a métrica de Fowlkes Mallows, é possível notar um declínio acentuado após a primeira época, principalmente para *clusters* específicos. Esses valores confirmam que o ELINAC não precisa de muitas épocas para agrupar este conjunto de dados, apresentando resultados satisfatórios no processo de identificação de *outliers*.



(a) Clusters específicos. (b) Clusters generalistas.

Figura 4.4: Avaliação de desempenho do ELINAC para métricas Informação Mútua e Fowlkes Mallows.

Esses resultados revelam que o ELINAC não precisa de muitas épocas para atingir as métricas de agrupamento ideais e o conjunto de dados rotulado usado para os experimen-

tos. Embora o conjunto de dados rotulado leve apenas algumas épocas para atingir os resultados ideais, observa-se que o ELINAC pode precisar de mais épocas de treinamento para obter resultados ideais para diferentes conjuntos de dados. Isso acontece porque o objetivo do ELINAC é gerar um valor para cada registro para que valores semelhantes possam agrupar textos semelhantes. Portanto, o ELINAC provou ser eficiente para agrupar dados de texto curto.

4.3 Avaliação de desempenho do ELINAC comparado a outras abordagens da literatura

Para validar o ELINAC, foi realizada uma análise comparativa com outras abordagens da literatura, conforme apresentado na Tabela 4.2. Os resultados são mostrados para ambos os tipos de *cluster* (ou seja, específico e generalistaa), e os valores das métricas de *cluster* são comparados para as abordagens de *cluster* escolhidas. Para *clusters* específicos, os resultados para a maioria das métricas são relativamente próximos, onde o ELINAC tem uma Completude de 95% enquanto outros variam entre 96% e 97%, e para Homogeneidade, ELINAC tem um desempenho melhor que DBSCAN e é 2% menor que o melhor método, o algoritmo Hierárquico. Para informações mútuas, o K-means se destaca, sendo quase 2% melhor que o segundo colocado. Fowlkes Mallows é a métrica que apresenta mais variação. O K-means tem o maior valor, onde ELINAC e o Hierárquico apresentam resultados muito próximos, ambos próximos de 78% e DBSCAN quase 3% menores. O ELINAC apresenta queda de desempenho para *clusters* generalistas, mas o mesmo ocorre para DBSCAN e K-means. Os métodos ELINAC e K-means apresentam consistência, enquanto DBSCAN tem Completude muito alta, mas também baixa Homogeneidade e apenas 19% para Fowlkes Mallows.

Tabela 4.2: Avaliação de desempenho do ELINAC em comparação com outras abordagens da literatura.

Método	**Clusters**	**Completude**	**Homogeneidade**	**Informação Mútua**	**Fowlkes Mallows**
ELINAC	Específico	95.051%	95.708%	87.070%	77.675%
K-means	Específico	96.630%	**97.228%**	**91.416%**	**82.712%**
DBSCAN	Específico	**96.945%**	94.883%	89.088%	74.909%
Hierárquico	Específico	96.928%	95.578%	89.883%	78.547%
ELINAC	Generalista	80.885%	86.699%	68.903%	40.518%
K-means	Generalista	88.383%	96.906%	85.134%	55.684%
DBSCAN	Generalista	**97.223%**	76.523%	76.938%	19.109%
Hierárquico	Generalista	95.731%	**97.204%**	**93.653%**	**84.831%**

A Figura 4.5 apresenta o gráfico do tempo de agrupamento para cada método. O eixo y consiste na escala logarítmica do tempo em segundos para melhor visualização da dispersão em cada caixa. O método de melhor desempenho é o ELINAC, que foi capaz de obter um tempo médio de agrupamento de 2 segundos para *clusters* específicos e, ainda melhor, 1,8 segundos para *clusters* generalistas. ELINAC apresenta maior dispersão nos resultados do que DBSCAN e Hierárquico, o que se deve ao fato de que a maior parte do tempo é gasto durante a fase de treinamento do autoencoder. O tempo de agrupamento do ELINAC é consideravelmente melhor quando comparado aos algoritmos da literatura. Seu agrupamento apresenta desempenho superior, ou seja, três vezes mais rápido no pior caso. O DBSCAN executa o segundo melhor atrás apenas do ELINAC e produz tempos muito consistentes, com uma média de tempo de *cluster* de 6 segundos para ambos os tipos de *cluster*. O K-means se destaca como o método com maior variação no tempo, o que é causado pelo número de *clusters* que escalam sua complexidade. O tempo médio do algoritmo é de 13,2 segundos para *clusters* generalistas e 22,1 segundos para *clusters* específicos. Causado por sua complexidade quadrática, o método de pior desempenho é o algoritmo Hierárquico, que agrupa consistentemente *clusters* generalistas e específicos em 527,3 e 526,3 segundos, respectivamente.

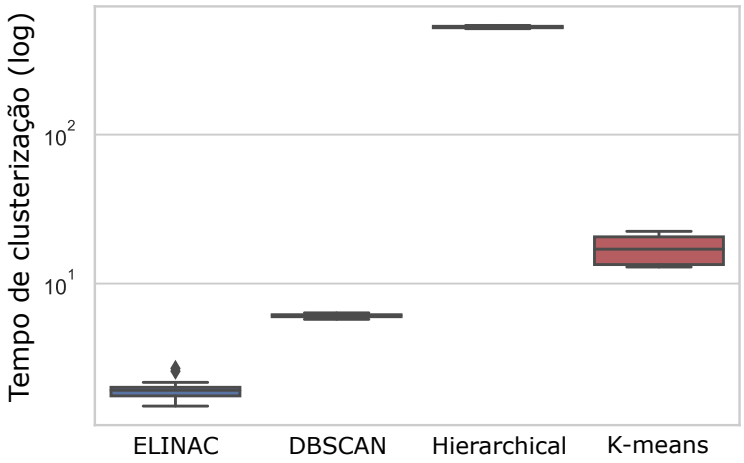


Figura 4.5: Dispersão do tempo de agrupamento em escala logarítmica para *clusters* generalistas e específicos.

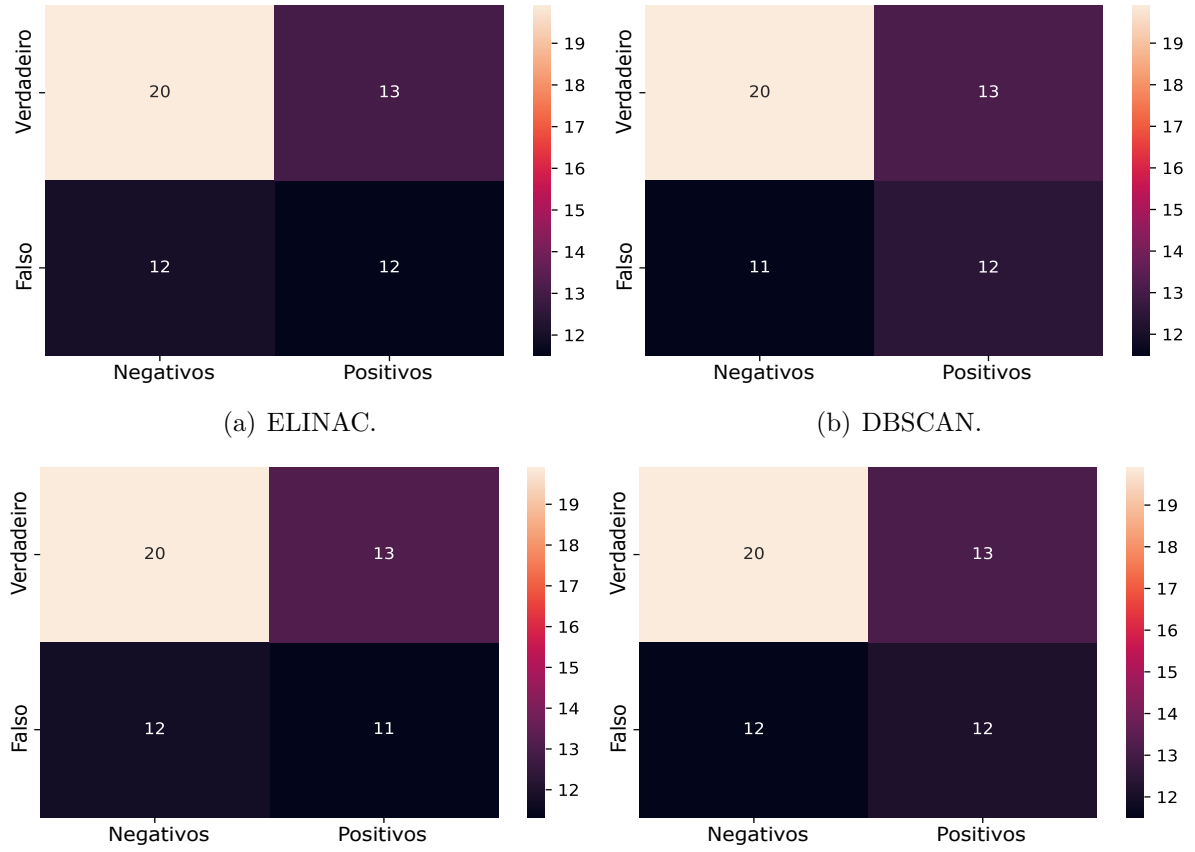
Para melhor avaliar o tempo de agrupamento em grandes volumes de dados, foi utilizado um conjunto de dados de 1 milhão de registros de NF-e. A Tabela 4.3 mostra o tempo necessário para o agrupamento, onde apenas dois métodos conseguiram terminar em um tempo razoável. Como o experimento foi baseado em dados com as mesmas características do conjunto de dados rotulado, pode-se supor que os resultados de agrupamento

seriam próximos aos experimentos rotulados usando os mesmos parâmetros. O DBSCAN terminou seu agrupamento em 2,7 horas (10.211 segundos), enquanto o ELINAC leva 207 segundos por época, o que significa que o ELINAC pode realizar 49 épocas de treinamento e ainda ser mais rápido que o DBSCAN. A julgar pela complexidade do algoritmo, pode-se supor que o K-means acabaria por terminar e teria resultados próximos aos do conjunto de dados rotulado. Para o algoritmo Hierárquico, ao analisar quanto tempo levou para 21 mil registros, podemos concluir que, seguindo sua complexidade quadrática, não terminaria em tempo admissível para este projeto. Com base nesses resultados, K-means e Hierárquico não são adequados para o objetivo de agrupamento extensivo de dados de NF-e. Observa-se que ELINAC é a abordagem mais adequada para identificar *outliers*, superando significativamente o segundo melhor algoritmo (ou seja, DBSCAN) no tempo de agrupamento.

Tabela 4.3: Tempos para agrupar o conjunto de dados completo com 1 milhão de amostras para NF-es.

Método	**Tempo**
ELINAC	206.6 s/epoch
K-means	-
DBSCAN	10210.979 s
Hierárquico	-

As Figuras 4.6 e 4.7 apresentam a matriz de confusão de pares para cada algoritmo comparando com *clusters* específicos e generalistas, respectivamente. Os valores são apresentados na escala logarítmica para melhor visualização da escala de cores.



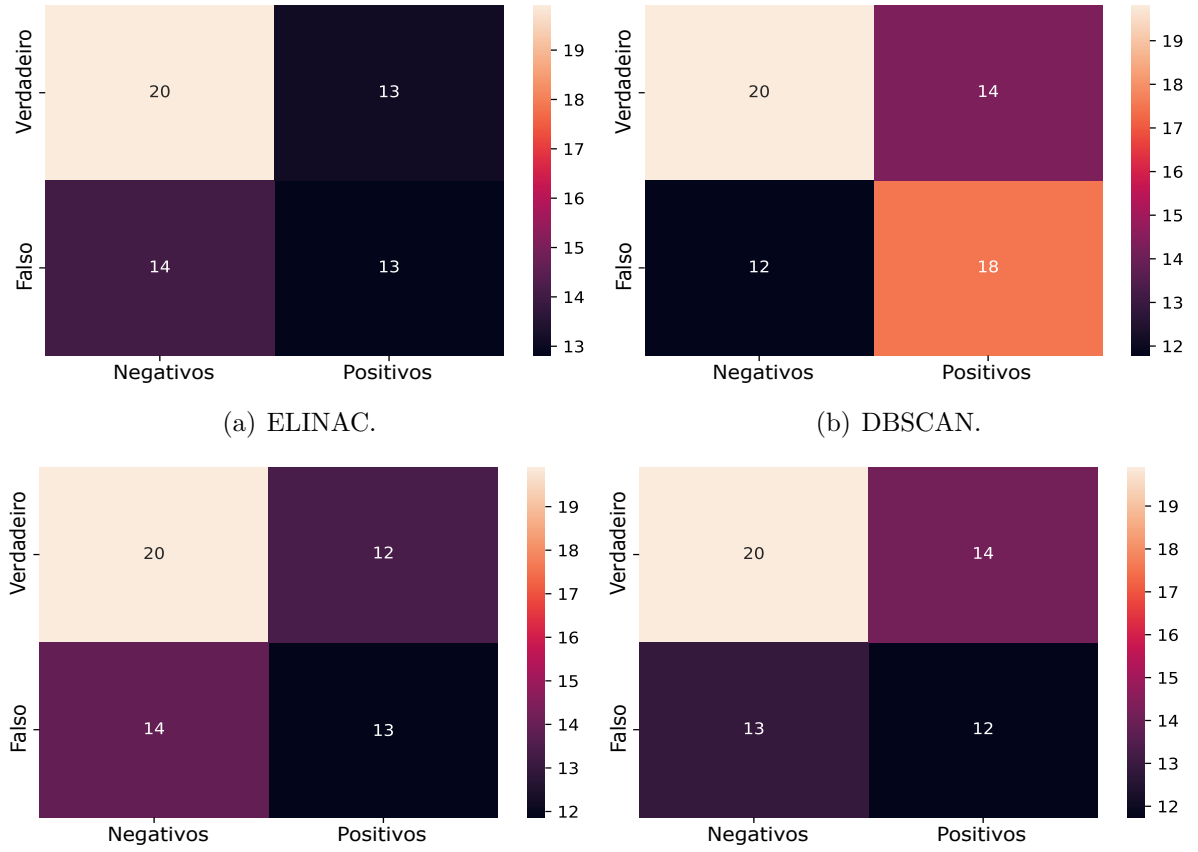
(a) ELINAC.

(b) DBSCAN.

(c) K-means.

(d) Hierárquico.

Figura 4.6: Avaliação do desempenho de cada método ao atribuir registros a *clusters* específicos.



(a) ELINAC. (b) DBSCAN.

(c) K-means. (d) Hierárquico.

Figura 4.7: Avaliação do desempenho de cada método ao atribuir registros a *clusters* generalistas.

Analisando a Figura 4.6, todos os quatro métodos são muito semelhantes, com variações mais perceptíveis em falsos negativos e falsos positivos. O K-means teve o melhor resultado para os pares de falsos positivos e verdadeiros negativos, enquanto o DBSCAN teve o melhor resultado para falsos negativos e verdadeiros positivos. Os resultados do ELINAC estão próximos do K-means, com um pouco mais de falsos negativos. Isso significa que o ELINAC é muito consistente quando comparado aos outros algoritmos. Embora o algoritmo Hierárquico tenha valores semelhantes, há mais falsos positivos e menos verdadeiros negativos quando comparado ao ELINAC.

A Figura 4.7 mostra mais variação nos resultados das matrizes, onde a diferença mais notável é entre ELINAC e DBSCAN, que apresentaram muito mais falsos positivos. Embora o ELINAC tenha um desempenho pior para *clusters* generalistas em comparação com seus resultados para *clusters* específicos, ele ainda conseguiu ser muito consistente e confiável, além de ser mais rápido no tempo de agrupamento (Figura 4.5). Ao contrário disso, o DBSCAN mostrou muita inconsistência, tendo muito mais falsos positivos do que verdadeiros positivos. Para os outros métodos, K-means e Hierárquico tiveram desempe-

nho semelhante para verdadeiros negativos e falsos positivos, mas muito diferentes para verdadeiros positivos e falsos negativos.

4.4 Discussão

Os experimentos realizados apresentaram resultados valiosos, que não divergiram muito das expectativas iniciais. Analisando os resultados, houve uma distinção significativa entre ELINAC e DBSCAN, que são mais eficientes em termos de tempo, e os algoritmos K-means e Hierárquico, que são mais demorados. Embora o ELINAC e o DBSCAN tenham semelhanças nas métricas de agrupamento, eles também tiveram um desempenho melhor ao considerar o tempo de agrupamento. Para K-means e Hierárquico, as métricas de agrupamento mostraram resultados significativamente melhores com a penalidade de demorar mais, especialmente para o algoritmo Hierárquico.

O algoritmo Hierárquico, que utilizou a abordagem aglomerativa, teve uma desvantagem significativa no desempenho, ao contrário do ELINAC. No entanto, foi bastante consistente para as métricas de agrupamento, principalmente para agrupamentos mais generalistas. Embora essa abordagem hierárquica possa não ser viável para agrupar dados extensivos de NF-e, ela apresentou vantagens claras na busca de *clusters* com mais variação.

O K-means foi uma das abordagens mais consistentes, sendo semelhante ao Hierárquico na maioria das métricas e mantendo resultados próximos para *clusters* generalistas. Embora tenha um desempenho significativamente mais rápido que o Hierárquico, sua complexidade de tempo mostrou ser uma barreira para conjuntos de dados mais extensos, em oposição ao ELINAC.

O DBSCAN é uma escolha comum para soluções de *cluster* com eficiência de tempo devido à sua complexidade. O algoritmo apresentou-se como uma solução balanceada, fornecendo bons resultados de agrupamento sem consumir muito tempo. A principal desvantagem do método foi a inconsistência observada nas métricas de Homogeneidade e Fowlkes Mallows, principalmente ao formar *clusters* homogêneos para o rótulo generalista.

A ELINAC cumpriu o que foi proposto para ele: agrupar os dados de forma rápida e eficiente. Os resultados do tempo de agrupamento mostraram que não só a abordagem é eficaz em termos de tempo, mas também conseguiu superar o segundo algoritmo mais rápido (ou seja, DBSCAN) em quase três vezes no tempo de agrupamento. Portanto, o ELINAC foi capaz de agrupar de forma semelhante a outras abordagens conhecidas, ao mesmo tempo em que o fazia consideravelmente mais rápido, evidenciando seu avanço para o estado da arte.

Capítulo 5

Conclusão

Uma sociedade transparente, em que as transações com dinheiro público sejam realizadas com integridade demonstrável e pela necessidade das pessoas, ainda está longe da realidade. Com o avanço da tecnologia, especialmente na área de ciência de dados e aprendizado de máquina, passos estão sendo dados na direção certa. Este trabalho foi feito para aprimorar o processo de auditoria dos gastos públicos, fornecendo uma solução para os órgãos responsáveis pela fiscalização do dinheiro gasto no Brasil. O ELINAC foi desenvolvido para agrupar os dados de NF-e com base em sua descrição de texto curto enquanto realiza o processo o mais rápido possível. Para isso, o ELINAC é composto por vários componentes que ajudam a atingir seu objetivo: pré-processar a descrição, transformar os dados de texto curto através da redução da dimensionalidade do autoencoder e, finalmente, agrupar os dados resultantes com um algoritmo otimizado.

Para avaliar o ELINAC, uma série de experimentos foram conduzidos para validar os resultados em comparação com outras abordagens da literatura para dois tipos diferentes de *clusters*. As métricas escolhidas representam a capacidade de cada método de criar *clusters* precisos e o custo computacional necessário para isso. Os resultados mostram que o ELINAC é eficiente no agrupamento de dados de texto curto, mostrando superioridade no tempo de agrupamento. Além disso, o ELINAC conseguiu finalizar seu processo de agrupamento três vezes mais rápido que o segundo método mais rápido, validando seu propósito para este trabalho. As principais contribuições do ELINAC são: (i) Agrupamento de dados de texto curto em um curto espaço de tempo baseado em uma rede de autoencoder; (ii) Permitir que os parâmetros de agrupamento sejam ajustados sem retreinamento; (iii) Desempenho mais rápido do que os modelos de agrupamento tradicionais; e (iv) Unificação do processo de treinamento para agrupamento e detecção de *outliers*.

Como trabalho futuro, pretendemos analisar o impacto do ELINAC nos processos de auditoria em tempo real. Além disso, pretendemos propor um serviço de coleta de texto espaço-temporal para melhorar o agrupamento de dados de texto curto no ELINAC.

Pretendemos também desenvolver uma ferramenta baseada no ELINAC que seja capaz de detectar anomalias, independentemente do domínio ou estrutura adotada.

Referências

[1] Rezaee, Zabihollah, Ahmad Sharbatoghlie, Rick Elam e Peter L McMickle: *Continuous auditing: Building automated auditing capability.* Auditing: A Journal of Practice & Theory, 21(1):147–163, 2002. 1

[2] Ozgediz, Selcuk e Paramjit Sachdeva: *Managing the public service in developing countries.* World Bank Washington, DC, 1983. 1

[3] Mergel, Ines, R Karl Rethemeyer e Kimberley Isett: *Big data in public affairs.* Public Administration Review, 76(6):928–937, 2016. 1

[4] Klievink, Bram, Bart Jan Romijn, Scott Cunningham e Hans de Bruijn: *Big data in the public sector: Uncertainties and readiness.* Information systems frontiers, 19(2):267–283, 2017. 1

[5] Munné, Ricard: *Big data in the public sector.* Em *New Horizons for a Data-Driven Economy*, páginas 195–208. Springer, Cham, 2016. 1

[6] Weigang, Li, Liriam Michi Enamoto, Denise Leyi Li e Geraldo Pereira Rocha Filho: *New directions for artificial intelligence: human, machine, biological, and quantum intelligence.* Frontiers of Information Technology & Electronic Engineering, páginas 1–7, 2021. 1

[7] Hanf, Matthieu, Astrid Van-Melle, Florence Fraisse, Amaury Roger, Bernard Carme e Mathieu Nacher: *Corruption kills: estimating the global impact of corruption on children deaths.* PLoS One, 6(11):e26990, 2011. 1

[8] Bentzen, Jeanet Sinding: *How bad is corruption? cross-country evidence of the impact of corruption on economic prosperity.* Review of Development Economics, 16(1):167–184, 2012. 1

[9] Levi, Michael e John Burrows: *Measuring the impact of fraud in the uk: A conceptual and empirical journey.* The British Journal of Criminology, 48(3):293–318, 2008. 1

[10] Fantaye, Dawit Kiros: *Fighting corruption and embezzlement in third world countries.* The Journal of criminal law, 68(2):170–176, 2004. 1

[11] Andrews, Rhys, George A Boyne e Gareth Enticott: *Performance failure in the public sector: misfortune or mismanagement?* Public Management Review, 8(2):273–296, 2006. 1

[12] *Brazilian electronic invoices.* `https://www.nfe.fazenda.gov.br/portal/infoEstatisticas.aspx`. Accessed: 2021-03-02. 1

[13] Chalapathy, Raghavendra e Sanjay Chawla: *Deep learning for anomaly detection: A survey.* arXiv preprint arXiv:1901.03407, 2019. 2

[14] Tan, Pang Ning, Michael Steinbach e Vipin Kumar: *Data mining cluster analysis: basic concepts and algorithms.* Introduction to data mining, páginas 487–533, 2013. 2

[15] Schmitz, Matheus, Roger Immich, Gustavo Pessin e Geraldo Pereira Rocha Filho: *Towards the categorization of brazilian financial market headlines.* IEEE Latin America Transactions, 20(2):344–351, 2021. 2, 6

[16] Enamoto, Liriam, Li Weigang *et al.*: *Generic framework for multilingual short text categorization using convolutional neural network.* Multimedia Tools and Applications, 80(9):13475–13490, 2021. 2, 6

[17] Ahmed, Mohiuddin, Abdun Naser Mahmood e Md Rafiqul Islam: *A survey of anomaly detection techniques in financial domain.* Future Generation Computer Systems, 55:278–288, 2016. 2

[18] Wang, Ruoying, Kexin Nie, Tie Wang, Yang Yang e Bo Long: *Deep learning for anomaly detection.* Em *Proceedings of the 13th International Conference on Web Search and Data Mining*, páginas 894–896, 2020. 2

[19] Agrawal, Shikha e Jitendra Agrawal: *Survey on anomaly detection using data mining techniques.* Procedia Computer Science, 60:708–713, 2015. 2

[20] Chang, Yunpeng, Zhigang Tu, Wei Xie e Junsong Yuan: *Clustering driven deep autoencoder for video anomaly detection.* Em *European Conference on Computer Vision*, páginas 329–345. Springer, 2020. 2

[21] Markovitz, Amir, Gilad Sharir, Itamar Friedman, Lihi Zelnik-Manor e Shai Avidan: *Graph embedded pose clustering for anomaly detection.* Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, páginas 10539–10547, 2020. 2

[22] Jain, Anil K e Richard C Dubes: *Algorithms for clustering data.* Prentice-Hall, Inc., 1988. 2

[23] Ester, Martin, Hans Peter Kriegel, Jörg Sander, Xiaowei Xu *et al.*: *A density-based algorithm for discovering clusters in large spatial databases with noise.* Em *kdd*, volume 96, páginas 226–231, 1996. 2, 20

[24] Shirkhorshidi, Ali Seyed, Saeed Aghabozorgi, Teh Ying Wah e Tutut Herawan: *Big data clustering: a review.* Em *International conference on computational science and its applications*, páginas 707–720. Springer, 2014. 2

[25] Yang, Xu, Cheng Deng, Feng Zheng, Junchi Yan e Wei Liu: *Deep spectral clustering using dual autoencoder network.* Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, páginas 4066–4075, 2019. 2, 6

[26] Lim, Kart Leong, Xudong Jiang e Chenyu Yi: *Deep clustering with variational autoencoder.* IEEE Signal Processing Letters, 27:231–235, 2020. 2, 6

[27] Mrabah, Nairouz, Naimul Mefraz Khan, Riadh Ksantini e Zied Lachiri: *Deep clustering with a dynamic autoencoder: From reconstruction towards centroids construction.* Neural Networks, 130:206–228, 2020. 2, 6

[28] Yang, Bo, Xiao Fu, Nicholas D Sidiropoulos e Mingyi Hong: *Towards k-means-friendly spaces: Simultaneous deep learning and clustering.* Em *international conference on machine learning*, páginas 3861–3870. PMLR, 2017. 2, 6

[29] Fard, Maziar Moradi, Thibaut Thonet e Eric Gaussier: *Deep k-means: Jointly clustering with k-means and learning representations.* Pattern Recognition Letters, 138:185–192, 2020. 2, 6, 7

[30] Kim, Younghwan e Huy Kang Kim: *Cluster-based deep one-class classification model for anomaly detection.* Journal of Internet Technology, 22(4):903–911, 2021. 2, 6, 7

[31] Paula, Ebberth L, Marcelo Ladeira, Rommel N Carvalho e Thiago Marzagao: *Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering.* Em *2016 15th ieee international conference on machine learning and applications (icmla)*, páginas 954–960. IEEE, 2016. 2, 5

[32] Borghesi, Andrea, Andrea Bartolini, Michele Lombardi, Michela Milano e Luca Benini: *A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems.* Engineering Applications of Artificial Intelligence, 85:634–644, 2019. 2

[33] Zhou, Chong e Randy C Paffenroth: *Anomaly detection with robust deep autoencoders.* Em *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, páginas 665–674, 2017. 2

[34] Schulte, Johannes P, Felipe T Giuntini, Renato A Nobre, Khalil C do Nascimento, Rodolfo I Meneguette, Weigang Li, Vinícius P Gonçalves e Geraldo P Rocha Filho: *Elinac: Autoencoder approach for electronic invoices data clustering.* Applied Sciences, 12(6):3008, 2022. 3

[35] Pourhabibi, Tahereh, Kok Leong Ong, Booi H Kam e Yee Ling Boo: *Fraud detection: A systematic literature review of graph-based anomaly detection approaches.* Decision Support Systems, 133:113303, 2020. 5

[36] Huang, Dongxu, Dejun Mu, Libin Yang e Xiaoyan Cai: *Codetect: Financial fraud detection with anomaly feature detection.* IEEE Access, 6:19161–19174, 2018. 5

[37] Nian, Ke, Haofan Zhang, Aditya Tayal, Thomas Coleman e Yuying Li: *Auto insurance fraud detection using unsupervised spectral ranking for anomaly.* The Journal of Finance and Data Science, 2(1):58–75, 2016. 5

[38] Ahmed, Mohiuddin, Nazim Choudhury e Shahadat Uddin: *Anomaly detection on big data in financial markets*. Em *2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, páginas 998–1001. IEEE, 2017. 5

[39] Elliott, Andrew, Mihai Cucuringu, Milton Martinez Luaces, Paul Reidy e Gesine Reinert: *Anomaly detection in networks with application to financial transaction networks*. arXiv preprint arXiv:1901.00402, 2019. 5

[40] Bezerra, Fábio, Jacques Wainer e Wil MP van der Aalst: *Anomaly detection using process mining*. Em *Enterprise, business-process and information systems modeling*, páginas 149–161. Springer, 2009. 5

[41] Kieckbusch., Diego, Geraldo Filho., Vinicius Di Oliveira. e Li Weigang.: *Scan-nf: A cnn-based system for the classification of electronic invoices through short-text product description*. Em *Proceedings of the 17th International Conference on Web Information Systems and Technologies - WEBIST,*, páginas 501–508. INSTICC, SciTePress, 2021, ISBN 978-989-758-536-4. 5

[42] Tang, Peng, Weidong Qiu, Min Yan, Zheng Huang, Shuang Chen e Huijuan Lian: *Association analysis of abnormal behavior of electronic invoice based on k-means and skip-gram*. Em *2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC)*, páginas 300–305. IEEE, 2019. 5

[43] Song, Chunfeng, Feng Liu, Yongzhen Huang, Liang Wang e Tieniu Tan: *Auto-encoder based data clustering*. Em *Iberoamerican congress on pattern recognition*, páginas 117–124. Springer, 2013. 6

[44] Angluin, Dana e Philip Laird: *Learning from noisy examples*. Machine Learning, 2(4):343–370, 1988. 8

[45] Gamberger, Dragan, Nada Lavrac e Saso Dzeroski: *Noise detection and elimination in data preprocessing: experiments in medical domains*. Applied artificial intelligence, 14(2):205–223, 2000. 8

[46] García, Vicente, Roberto Alejo, José Salvador Sánchez, José Martínez Sotoca e Ramón Alberto Mollineda: *Combined effects of class imbalance and class overlap on instance-based classification*. Em *International Conference on Intelligent Data Engineering and Automated Learning*, páginas 371–378. Springer, 2006. 8

[47] Hernández, Mauricio A e Salvatore J Stolfo: *Real-world data is dirty: Data cleansing and the merge/purge problem*. Data mining and knowledge discovery, 2(1):9–37, 1998. 8

[48] Aggarwal, Charu C.: *Outlier Analysis*. Springer, 2013. `https://doi.org/10.1007/978-1-4614-6396-2`. 13, 14

[49] Kingma, Diederik P e Jimmy Ba: *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980, 2014. 15, 21

[50] *Python software foundation. python language reference, version 3.7.9.* `https://www.python.org`. Accessed: 2021-03-02. 20

[51] Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu e Xiaoqiang Zheng: *TensorFlow: Large-scale machine learning on heterogeneous systems.* `https://www.tensorflow.org/`, 2015. Software available from tensorflow.org. 20

[52] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay: *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, 12:2825–2830, 2011. 20

[53] Learn, Scikit: *K-Means clustering.* `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`. Accessed: 2021-10-10. 20

[54] Lloyd, Stuart: *Least squares quantization in pcm.* IEEE transactions on information theory, 28(2):129–137, 1982. 20

[55] Day, William HE e Herbert Edelsbrunner: *Efficient algorithms for agglomerative hierarchical clustering methods.* Journal of classification, 1(1):7–24, 1984. 20