Trabalho de Conclusão de Curso

Classificação de produtos em Notas Fiscais Eletrônicas usando Descrições Textuais não estruturadas

Maria Tatiane Medeiros dos Santos

Orientador:

Prof. Dr. Thales Miranda de Almeida Vieira

Maceió, Janeiro de 2023

Maria Tatiane Medeiros dos Santos

Classificação de produtos em Notas Fiscais Eletrônicas usando Descrições Textuais não estruturadas

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. Thales Miranda de Almeida Vieira

Maceió, Janeiro de 2023

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Taciana Sousa dos Santos – CRB-4 – 2062

S237c Santos, Maria Tatiane Medeiros dos.
Classificação de produtos em Notas Fiscais Eletrônicas usando descrições textuais não estruturadas / Maria Tatiane Medeiros dos Santos. – 2023.
39 f. : il. color.

Orientador: Thales Miranda de Almeida Vieira.
Monografia (Trabalho de Conclusão de Curso em Engenharia da Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2022.

Bibliografia: f. 38-39.

1. Aprendizagem de máquina. 2. Processamento de linguagem natural. 3. Redes neurais recorrentes. I. Título.

CDU: 004.8

Agradecimentos

A Deus, por todas as bençãos diárias;

A minha família, aos meus pais, Efigênia e Janaildo, e minhas irmãs, Pâmela e Tainá, por todo o apoio nessa jornada;

Aos meus professores, coordenadores, colegas, e a todos que de alguma forma contribuíram para minha trajetória acadêmica.

Resumo

Este trabalho apresenta uma análise acerca da viabilidade do uso de algoritmos de aprendizagem de máquina para a classificação de itens de produtos de Nota Fiscal Eletrônica (NFE), se baseando nas descrições textuais dos produtos para os atribuir às unidades comerciais correspondentes. A base de dados selecionada foi a base de dados de NFE disponibilizada pelo Ministério Público do Estado da Paraíba (MPPB). Esta pesquisa tem o objetivo de contribuir com a busca de soluções para o desafio de definir exatamente qual produto está sendo descrito e qual a quantidade que foi comercializada em uma NFE, e assim auxiliar em uma possível fiscalização automatizada dos preços. Para realizar a classificação, primeiro foi realizada uma rotulação manual de uma amostra dos dados, classificando-as quanto à unidade de medida. Essa amostra foi utilizada no treinamento de dois modelos para realizar a classificação da descrição do produto: uma solução tradicional a nível de palavras, adotado como *baseline*; e uma solução mais a nível do estado da arte baseado em redes neurais recorrentes, a nível de caracteres. Realizamos uma comparação do desempenho dessas duas abordagens.

Palavras-chave: Aprendizagem de Máquina; Processamento de Linguagem Natural; Redes Neurais Recorrentes

Abstract

This work presents an analysis about the feasibility of using machine learning algorithms to product item classification in Electronic Invoices (NFE), based on the textual descriptions of the products to associate them with the corresponding commercial units. The database used in this work was extracted from a NFE database made available by Public Ministry of Paraiba (MPPB). This research has as purpose to contribute to the challenge of defining exactly which product is being described and what quantity was sold in an NFE, and thus help in a possible automated price inspection. To the classification, first was performed a manual labeling of a data sample, classifying them with the matching unit of measurement. This sample was used in the training of two models to perform a product description classification: a traditional word-based solution, adopted as baseline; and a state-of-the-art character-based solution based on recurrent neural networks. We perform a comparison of the performance of these two approaches.

Keywords: Machine Learning; Natural Language Processing; Recurrent Neural Network

Lista de Figuras

2.1 Representação de uma ANN multicamada com camada oculta única. 16
2.2 Estrutura de uma GRU (ZHANG et al., 2021) 18
2.3 Estrutura de uma Bi-GRU (ZHANG et al., 2021) 19
2.4 Codificador do *tweet2vec* para texto de rede social (DHINGRA et al., 2016). . . . 21
3.1 Abreviações de algumas unidades de medida. 24
3.2 Exemplos de descrição de produto. 25
4.1 Matriz de confusão do classificador com SVM linear. 31
4.2 Itens que deveriam ser classificados como "indefinido". 33
4.3 Matriz de confusão do classificador *tweet2vec* (1º experimento). 34
4.4 Matriz de confusão do classificador *tweet2vec* (2º experimento). 35

Lista de Abreviaturas e Siglas

AI	*Artificial Inteligence*
ANN	*Artificial Neural Network*
Bi-GRU	*Bidirectional Gated Recurrent Unit*
CFOP	Código Fiscal de Operações e Prestações
CPU	*Central Processing Unit*
CSV	*Comma-Separated Values*
GPU	*Graphical Processor Unit*
GRU	*Gated Recurrent Unit*
GTIN	*Global Trade Item Number*
LSTM	*Long short-term memory*
MPPB	Ministério Público da Paraíba
ML	*Machine Learning*
NFE	Nota Fiscal Eletrônica
NCM	Nomenclatura Comum do Mercosul
PLN	Processamento de Linguagem Natural
RNN	*Recurrent Neural Network*
SPED	Sistema Público de Escrituração Digital
SVM	*Support Vector Machine*
SBC	Sociedade Brasileira de Computação
TF-IDF	*Term Frequency–Inverse Document Frequency*

Sumário

Lista de Figuras vi
Lista de Abreviaturas e Siglas vii

1 Introdução **10**
1.1 Contextualização 10
1.2 Motivação 11
1.3 Objetivos 13

2 Referencial Teórico **14**
2.1 Processamento de Linguagem Natural 14
2.1.1 *Term Frequency–Inverse Document Frequency* (TF-IDF) 14
2.2 Aprendizagem de Máquina 15
2.2.1 Classificação 15
2.3 Máquina de Vetores de Suporte (SVM) 15
2.4 Redes Neurais Artificiais (ANN) 16
2.5 Redes Neurais Recorrentes (RNN) 17
2.5.1 *Long short-term memory* (LSTM) 17
2.5.2 *Gated Recurrent Unit* (GRU) 18
2.5.3 *Bidirectional Gated Recurrent Unit* (Bi-GRU) 19
2.6 Métricas de Avaliação 20
2.7 Tweet2Vec 21

3 Metodologia **23**
3.1 Os Dados 23
3.1.1 Amostra e Rotulação 24
3.2 Pré-processamento 25
3.2.1 Limpeza de texto 26
3.2.2 Remoção de palavras de parada 26
3.3 Modelos de Classificação 26
3.3.1 Classificação com TF-IDF e SVM 27
3.3.2 Classificação com *tweet2vec* 28
3.3.3 Avaliação 29

4 Resultados e Discussões **30**
4.1 Classificação com TF-IDF + SVM Linear 30
4.2 Classificação com *tweet2vec* 31
4.2.1 Configuração dos Experimentos 32
4.2.2 1° Experimento: 32
4.2.3 2º Experimento 34

5 Conclusão **36**

Referências bibliográficas **39**

1

Introdução

1.1 Contextualização

O Brasil é um dos maiores emissores mundiais de documentos fiscais eletrônicos (BARREIX et al., 2018), sendo registrada uma quantidade em torno de 35 bilhões de documentos autorizados no Portal Nacional da Nota Fiscal Eletrônica até o ano de 2022 (Receita Federal do Brasil, 2022).

Essa trajetória se iniciou em 2007, quando o Governo do Brasil, com o objetivo de modernizar a escrituração contábil e fiscal das empresas no Brasil, criou o Sistema Público de Escrituração Digital (SPED), o qual unificou as atividades de recepção, validação, armazenamento e autenticação de livros e documentos contábeis e fiscais, mediante fluxo único e computadorizado das informações.

O SPED teve como uma de suas principais vertentes a implantação de uma Nota Fiscal Eletrônica (NFE) no Brasil. Sendo a substituição da nota fiscal em papel por um documento eletrônico com validade jurídica para todos os fins, e de existência apenas digital, uma das principais quebras de paradigma do Projeto NFE, que contribuiu com os esforços de transparência tributária, digitalização das administrações tributárias e intercâmbio de informações, aumentando o potencial de combate à sonegação fiscal e a outros atos ilícitos relacionados (MELLO, 2014).

A NFE é um documento emitido e armazenado exclusivamente de forma digital, com o intuito de documentar uma operação de circulação de mercadorias ou uma prestação de serviços, ocorrida entre as partes contratantes. Sua validade jurídica é garantida pela assinatura digital do remetente e a autorização de uso fornecida pela administração tributária da esfera do contribuinte, antes da ocorrência do fato gerador, que garantem a autoria, integridade e irrefutabilidade dos arquivos de notas fiscais (Receita Federal do Brasil, 2022).

A NFE possui as seguintes características (NETO, 2018):

- A NFE detalha os produtos ou serviços fornecidos por um vendedor a um comprador;
- A NFE é um modelo nacional, com *layout* de campos definidos por uma legislação específica;
- O arquivo de uma NFE é considerado dado público e seu arquivo pode ser consultado e descarregado através dos endereços eletrônicos estaduais e federais de agências tributárias. Sendo necessário apenas o código numérico (chave de acesso) do documento fiscal, que consiste numa sequência de 44 dígitos, obtido por meio de algoritmo fornecido pela administração tributária, juntamente com o número do cadastro nacional de pessoa jurídica (CNPJ) do emitente e do número da NFE.

Assim, as informações de determinada NFE podem ser acessadas nos portais das agências tributárias estaduais e federal, uma vez que se tenha as chaves de acessos destes documentos, e extraídas diversas informações, como por exemplo: dados cadastrais do emitente, dados cadastrais do destinatário, descrição e quantidade dos produtos ou serviços, valores monetários envolvidos na transação, dados sobre o transporte dos produtos, dentre outros dados.

Essa grande quantidade de dados armazenada na base de NFE é de grande interesse analítico para as agências de fiscalização tributária que, devido ao elevado volume de notas fiscais emitidas, historicamente realizam cruzamento de dados a fim de evitar tanto a omissão de vendas quanto a inclusão de falsas compras, os quais podem envolver sonegação fiscal e diversos tipos de fraudes. Com isso, podem monitorar e cobrar o imposto devido (OECD, 2022).

Recentemente, os dados armazenados na base de NFE também têm atraído o interesse analítico de outras agências governamentais responsáveis pela fiscalização, como controladorias, auditorias internas, Tribunais de Contas e Ministérios Públicos. Esses órgãos de controle tem acessado os dados de NFE visando desenvolver soluções de tecnologia para apoiar sua áreas finalísticas, tais como a formação de banco de preços de aquisições públicas e malhas eletrônicas de fiscalização, visando a detecção de possíveis irregularidades (vendas superfaturadas, fraudes em contratações, simulação de operações comerciais e outras), bem como promover uma maior transparências dos gastos incorridos pela Administração Pública com aquisições de bens e serviços (ENCCLA, 2018)(CGU, 2022).

1.2 Motivação

Apesar do potencial de utilidade dos dados da base de NFE para as agências governamentais de controle e fiscalização, surge um desafio comum na utilização destes dados: o problema

de agrupamento e classificação de produtos quanto à sua natureza e similaridade, que busca definir exatamente qual o produto que está sendo descrito e qual a quantidade que foi comercializada.

A NFE possui diversos campos de dados que são estruturados, e seguem padrões estabelecidos de acordo com especificação do *layout* da NFE em vigor, tais como os campos relacionados a dados cadastrais do fornecedor e comprador e códigos de descrição dos produtos como o NCM (Nomenclatura Comum do Mercosul), CFOP (Código Fiscal de Operações e Prestações) e GTIN (*Global Trade Item Number*).

Entretanto, muitos campos são de livre preenchimento pelo fornecedor, como é o caso do campo "Descrição do Produto ou Serviço" e o campo "Unidade de Comercialização do Produto". Para estes campos não estruturados, de livre preenchimento, cada contribuinte pode adotar uma forma própria para descrever um mesmo produto, resultando em uma grande variedade de formas de descrever um mesmo produto no documento fiscal emitido por diversos fornecedores. Problemas comuns podem envolver divergências de descritores textuais, como erros ortográficos, abreviações, prefixação, super especificação e subespecificação. Isto dificulta a comparabilidade de um mesmo produto entre diversos fornecedores ou compradores, e inviabiliza o processamento deste conteúdo em verificações automáticas realizadas por malhas eletrônicas de órgãos governamentais, limitando as análises aos campos estruturados.

Adicionalmente, uma empresa fornecedora pode manipular as informações de campos estruturados e não estruturados para não representar adequadamente um produto ou a sua quantidade comercializada, dificultando a verificação por ferramentas eletrônicas de cruzamento de dados das agências governamentais, o que remete para a importância também de se padronizar a variável quantidade de produtos comercializados.

Embora a NFE possua campos específicos para se informar a quantidade da mercadoria (campo numérico) e a unidade de comercialização adotada (campo texto de livre preenchimento), é comum a inserção de descrição de unidades de medidas de forma agrupada no campo da descrição do produto, que podem não condizer com os valores descritos em outros campos.

Assim, as descrições dos produtos, apesar de expressas em textos desestruturados, são uma alternativa para enriquecer a informação sobre a quantidade de produtos comercializados, por possuírem informações adicionais sobre as unidades de comercialização adotadas.

Entretanto, esta análise necessita do uso de estratégias mais sofisticadas de identificação da similaridade. Assim, seria possível o uso de algoritmos de aprendizagem de máquina para classificar os itens de produtos de acordo com as unidades de medidas corretas, independente do dado existente no campo específico de unidade de medida na base de dados da NFE?

1.3 Objetivos

Este trabalho tem como objetivo geral avaliar a viabilidade de uso de algoritmos de aprendizagem de máquina para classificar os itens de produtos de Nota Fiscal Eletrônica (NFE), atribuindo-os às unidades comerciais correspondentes, com base nas descrições textuais dos produtos.

Para possibilitar a realização do objetivo geral, podemos enumerar os seguintes objetivos específicos:

- Pesquisar técnicas de processamento de texto que possibilitem lidar com grandes volumes de dados de modo a torná-los adequados para o treinamento de classificadores;
- Avaliar técnicas de classificação de texto que sejam eficazes para a utilização em classificadores com múltiplas classes;
- Realizar o pré-processamento de dados, como limpeza, seleção e integração de dados, para que os algoritmos selecionados realizem a análise e o processamento dos dados;
- Implementar dois algoritmos classificadores de texto, um a nível de palavras e outro a nível de caracteres, para classificar as descrições textuais de itens de produtos, associando-os a unidades comerciais adequadas;
- Testar os classificadores sobre dados reais de transações obtidas a partir do banco de dados de NFE de aquisições públicas fornecido pelo MPPB;
- Analisar e comparar os resultados dos classificadores implementados utilizando métricas adequadas para avaliação.

2

Referencial Teórico

2.1 Processamento de Linguagem Natural

De acordo com Anchiêta et al. (2021), Processamento de Linguagem Natural (PLN) é uma vertente da Inteligência Artificial (IA) que ajuda computadores a entender, interpretar e manipular a linguagem humana.

A Comissão Especial de Processamento de Linguagem Natural (CE-PLN) da Sociedade Brasileira de Computação (SBC), estabelece que a área de PLN, também denominada Linguística Computacional, busca investigar, propor e desenvolver formalismos, modelos, técnicas e sistemas computacionais para resolver problemas relacionados à automação da interpretação e da geração da língua humana.

2.1.1 *Term Frequency–Inverse Document Frequency* (TF-IDF)

A medida chamada TF-IDF nos permite identificar palavras em uma coleção de documentos que são úteis para diferenciar cada documento. De acordo com Rajaraman e Ullman (2011), a medida chamada TD-IDF representa o quanto a ocorrência de uma determinada palavra é concentrada em relativamente poucos documentos. Uma palavra terá um alto valor TF-IDF em um documento se ela aparece em relativamente poucos documentos, mas aparece neste, e tende a aparecer muitas vezes.

Para calcular essa medida, suponha que f_{ij} seja a frequência de determinado termo (palavra) i em um documento j. Essa frequência é normalizada dividindo pelo número máximo de ocorrências de cada termo no documento j. O valor da frequência do termo TF_{ij} é dado por:

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \tag{2.1}$$

Para calcular a frequência inversa do documento (IDF), suponha que um termo i aparece em n_i documentos de um total de N documentos. Essa frequência será dada por:

$$IDF_i = \log_2 \frac{N}{n_i} \tag{2.2}$$

Assim, o valor TF-IDF de um termo i em um documento j será o produto TF_{ij} x IDF_i.

Os termos com valor TF-IDF mais altos geralmente são os termos que melhor caracterizam o tópico de determinado documento.

2.2 Aprendizagem de Máquina

Um algoritmo de aprendizagem de máquina (ML) é um algoritmo que é capaz de aprender através de um conjunto de dados e melhorar seu desempenho. Esse aprendizado é feito através da observação dos dados, instruções explícitas ou experiências. Para Rajaraman e Ullman (2011), o aprendizado de máquina é uma boa abordagem quando não se conhece muito sobre o que se está buscando nos dados.

Em geral, algoritmos de ML se dividem em:

- Aprendizagem Supervisionada: busca encontrar um modelo que relacione as observações de dados com valores previamente associados a estas, de forma a categorizar as observações atuais e prever as respostas para observações futuras. Um exemplo desse tipo de aprendizagem são os algoritmos de classificação;
- Não-supervisionado: utiliza dados de observações sem nenhum valor associado. A aprendizagem ocorre de acordo com o entendimento das relações entre tais observações e seus respectivos valores, que podem não ser claras para o entendimento humano. Um exemplo de aprendizagem não supervisionada são algoritmos de agrupamento.

2.2.1 Classificação

Em aprendizagem de máquina, a classificação é um tipo de problema que busca identificar um conjunto de categorias baseadas em um conjunto de treinamento de dados contendo atributos ou instâncias cuja associação à categoria pode ou não ser conhecida.

2.3 Máquina de Vetores de Suporte (SVM)

SVM é um conjunto de métodos de aprendizado supervisionado avançado que podem ser usados para resolver problemas de regressão e classificação, embora seja mais favorável para

a classificação.

Os itens ou registros do conjunto de dados são representados como pontos em um espaço n-dimensional, separados em classes por uma margem conhecida como hiperplano. Para realizar a classificação, nesse mesmo espaço n-dimensional, os itens de dados são mapeados para obter a previsão da categoria a que pertencem, de acordo com o lado do hiperplano em que estiverem (SEN; HAJRA; GHOSH, 2019).

É possível aplicar o SVM em conjuntos de dados que não são separados por funções lineares utilizando o truque do *kernel*, que utiliza núcleos que transformam o espaço de entrada de baixa dimensão em um espaço de dimensão superior, ou mesmo dimensão infinita.

2.4 Redes Neurais Artificiais (ANN)

Baseado nos estudos sobre o funcionamento do cérebro humano, as Redes Neurais Artificiais (ANN) buscam construir uma estrutura artificial que funcione de forma similar, tendo capacidades de processamento, de associação de padrões e de aprendizado.

Para Osorio e Bittencourt (2000), as ANNs são constituídas por um grafo orientado e ponderado, formado por um conjunto de unidades elementares de processamento de informações fortemente conectadas, os neurônios artificiais. Estes neurônios artificiais são autômatos simples, que formam através de suas conexões um autômato mais complexo, a rede neural, também conhecida como rede conexionista.

Cada unidade dessa rede é dotada de um estado interno, o qual é chamado de estado de ativação. As unidades podem propagar seu estado de ativação para as outras unidades do grafo por ligações ponderadas, chamadas de ligações sinápticas ou pesos sinápticos. A regra que determina a ativação de um neurônio em função da influência vinda de suas entradas, ponderadas pelos seus respectivos pesos, se chama regra de ativação ou função de ativação. Mudanças nos valores dos pesos sinápticos entre unidades de uma rede são responsáveis pelas alterações no comportamento de ativação desta, e isso é o que permite a rede aprender um novo comportamento.

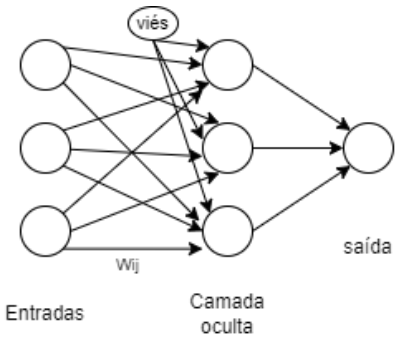

Figura 2.1: Representação de uma ANN multicamada com camada oculta única.

Uma ANN pode ser composta de três tipos de camadas:

- A camada de entrada, formada pelos nós de entrada;
- As camadas intermediárias, chamadas de camadas ocultas;
- A camada de saída, que representa o modelo resultante.

Cada nó de uma camada é conectado a todos nós da camada seguinte por uma ligação ponderada, com um peso W_{ij}. Na Figura 2.1 vemos uma representação da arquitetura de uma ANN de uma camada oculta (RIPLEY, 1996). Uma rede neural onde os nós de uma camada conectam-se entre si é chamada de rede recorrente.

2.5 Redes Neurais Recorrentes (RNN)

As Redes Neurais Recorrentes são um tipo especializado de rede neural, focadas em processar uma sequência de valores. As RNNs permitem trabalhar com dados sequenciais, como um texto (ANCHIÊTA et al., 2021).

Esse tipo de rede é bastante utilizado em algoritmos de PLN, que utilizam sequências de texto como entrada. Uma rede recorrente é qualquer estrutura em que o valor de uma unidade dependa de saídas anteriores (dos elementos anteriores da sequência) como entrada, seja direta ou indiretamente. Entre os vários tipos de RNN, temos dois que se destacam: *Long Short-Term Memory* (LSTM) e *Gated Recurrent Unit* (GRU), que foram criadas para solucionar o problemas das RNNs convencionais de "esquecer" informações ao longo do processamento, assim como os problemas com dissipação ou explosão do gradiente.

2.5.1 *Long short-term memory* (LSTM)

A rede LSTM é um tipo de Rede Neural Recorrente que possui a habilidade de aprender dependências de longo prazo. Ela foi proposta por Hochreiter e Schmidhuber (1997) para solucionar os problemas de dissipação ou de explosão do gradiente.

Gradientes são valores usados para atualizar os pesos das redes neurais. O problema da dissipação do gradiente ocorre quando esse gradiente diminui à medida que se propaga ao longo do tempo. Assim, nas redes neurais recorrentes, as camadas que recebem uma pequena atualização gradiente param de aprender. Dessa forma, as RNNs podem esquecer o que foi visto em sequências mais longas, tendo assim uma memória de curto prazo.

LSTMs têm sido utilizadas em algoritmos de reconhecimento de fala, análise emocional e análise de texto, pois possui memória própria e pode fazer previsões relativamente precisas (LU et al., 2020).

2.5.2 *Gated Recurrent Unit* (GRU)

Visando solucionar esses mesmos problemas de gradiente, uma Unidade Recorrente Fechada (GRU) foi proposta por Cho et al. (2014) para fazer com que cada unidade de uma camada oculta da rede capture de forma adaptativa as dependências de diferentes escalas de tempo. Redes com GRU possuem estrutura similar as redes LSTM, porém um pouco mais simplificada. Assim como a unidade LSTM, a GRU possui unidades fechadas que modulam o fluxo de informações dentro da unidade, porém, sem possuir células de memória separadas (CHUNG et al., 2014).

A GRU é composta pela entrada atual x_t, o estado oculto anterior h_{t-1}, o *update gate* z_t, o *reset gate* r_t e o novo estado oculto h_t. No processo a seguir, a entrada de GRU é x_t e h_{t-1}, e a saída é o novo estado oculto h_t, como mostra a Figura 2.2 (ZHANG et al., 2021).

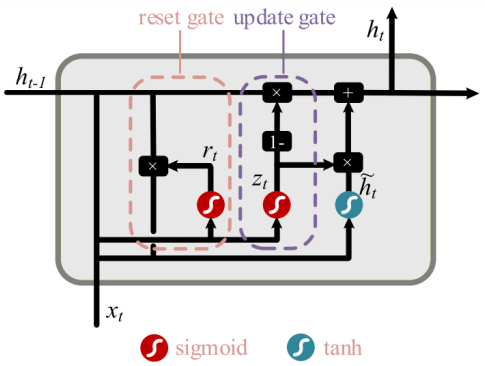

Figura 2.2: Estrutura de uma GRU (ZHANG et al., 2021)

- *Update gate* (z_t): é usado para controlar o grau em que a informação do momento anterior é trazida para o estado atual. Quanto maior o valor do *update gate*, mais informações são trazidas do momento anterior. Pode ser calculado por:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \tag{2.3}$$

  onde W_z e b_z são a matriz de peso e viés da porta de atualização, respectivamente. $\sigma(x) = 1/[1 + e^x]$ é a função de ativação *sigmoid*, pela qual os dados são mapeados no intervalo [0, 1] (sinal de controle da porta).

- *Reset gate* (r_t): é usado para controlar quanto das informações da camada oculta do momento anterior precisam ser esquecidas. Pode ser calculado por:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r), \tag{2.4}$$

  onde W_r e b_r são a matriz de peso e viés da porta de atualização, respectivamente. O valor dessa função será 0 para apagar a informação de estado oculto do momento anterior.

- *Candidate output state* ($\tilde{h}_t$):

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h), \tag{2.5}$$

onde W_h e b_h são a matriz de peso e viés do estado *candidate output*, respectivamente. A função *tanh* ativa a função para dimensionar os dados para o intervalo de -1 a 1. O produto de Hadamard, representado por $\odot$, é a multiplicação dos elementos correspondentes na matriz.

- Novo estado da camada oculta (h_t): A saída de GRU é o novo estado da camada oculta, que possui a seguinte expressão matemática

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h_t. \tag{2.6}$$

2.5.3 *Bidirectional Gated Recurrent Unit* (Bi-GRU)

Diferente do GRU tradicional, que só pode prever a saída do próximo momento com base nas informações do momento anterior, o Bi-GRU combina as informações da sequência de entrada nas direções para frente e para trás (*forward* e *backward*) (ZHANG et al., 2021). Sua estrutura é mostrada na Figura 2.3.

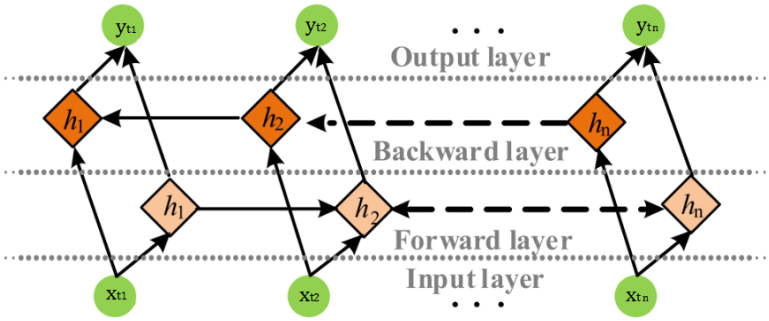

Figura 2.3: Estrutura de uma Bi-GRU (ZHANG et al., 2021)

A estrutura bidirecional pode ser considerada como duas camadas ocultas com direções opostas de transferência de informações e unidades GRU. No momento t, a entrada x_t fornece as camadas ocultas em duas direções opostas simultaneamente. A saída y_t no momento t é determinada conjuntamente por essas duas camadas ocultas unidirecionais. A camada *forward* de GRU possui a informação do momento t e o momento anterior na sequência de entrada, enquanto a camada *backward* de GRU possui a informação do momento t e o momento subsequente na sequência de entrada.

Uma estrutura de camada oculta cíclica bidirecional pode capturar as dependências *forward* e *backward* no tempo, e pode efetivamente melhorar a capacidade de processamento do modelo para dados não lineares.

2.6 Métricas de Avaliação

Existem várias métricas que são usadas para avaliar o treinamento e o teste de um algoritmo classificador:

- Acurácia: consiste na razão entre o número de predições verdadeiras e o número total de amostras. Indica a performance do modelo, então quanto maior a acurácia, mais as predições deste modelo se aproximam dos valores de referência.

- Matriz de confusão: Para Anchiêta et al. (2021), a matriz de confusão é utilizada para identificar a performance do modelo para cada classe. A matriz fornece quantas amostras de cada classe foram classificadas corretamente.

  A matriz de confusão segue a seguinte forma:

  $$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix},$$

  onde:

  - TN (Verdadeiro Negativo): a quantidade de rótulos da classe negativa que o modelo previu corretamente como negativo;
  - FP (Falso Positivo): a quantidade de rótulos que originalmente pertencem a classe negativa, mas o modelo previu como positivo;
  - FN (Falso Negativo): a quantidade de rótulos que originalmente pertencem a classe positiva, mas o modelo previu como negativo;
  - TP (Verdadeiro Positivo): a quantidade de rótulos da classe positiva que o modelo previu corretamente como positivo.

  Uma matriz de confusão com valores mais altos na diagonal (TN e TP) representa uma alta taxa de acerto do modelo.

- Precisão (P): o número de amostras de determinada classe que realmente pertencem a esta classe:

  $$P = \frac{TP}{TP + FP}. \tag{2.7}$$

  Um valor alto de precisão é desejado quando se quer evitar falsos positivos.

- Cobertura (*Recall*): o número de predições corretas de uma determinada classe:

$$R = \frac{TP}{TP + FN}. \tag{2.8}$$

Um valor alto de *recall* é desejado quando se quer evitar falsos negativos.

2.7 Tweet2Vec

Textos de rede social possuem muitos desafios que podem tornar as abordagens tradicionais de PLN pouco eficientes, como erros ortográficos, escrita informal, abreviações e símbolos, levando a um vocabulário desnecessariamente grande para abordagens em nível de palavra.

Dhingra et al. (2016) propôs um modelo a nível de caracteres, o *tweet2vec*, que encontra representações em espaço vetorial de tweets inteiros pelo aprendizado de dependências complexas e não locais nas sequências de caracteres.

A proposta do *tweet2vec* é uma rede neural bi-direcional com *Gated Recurrent Unit* (Bi-GRU) para codificar *tweets*. O modelo processa as sequências de caracteres em *forward* e *backward*, e os estados GRU finais são combinados linearmente para obter a associação do *tweet*. A sua estrutura é mostrada na figura 2.4.

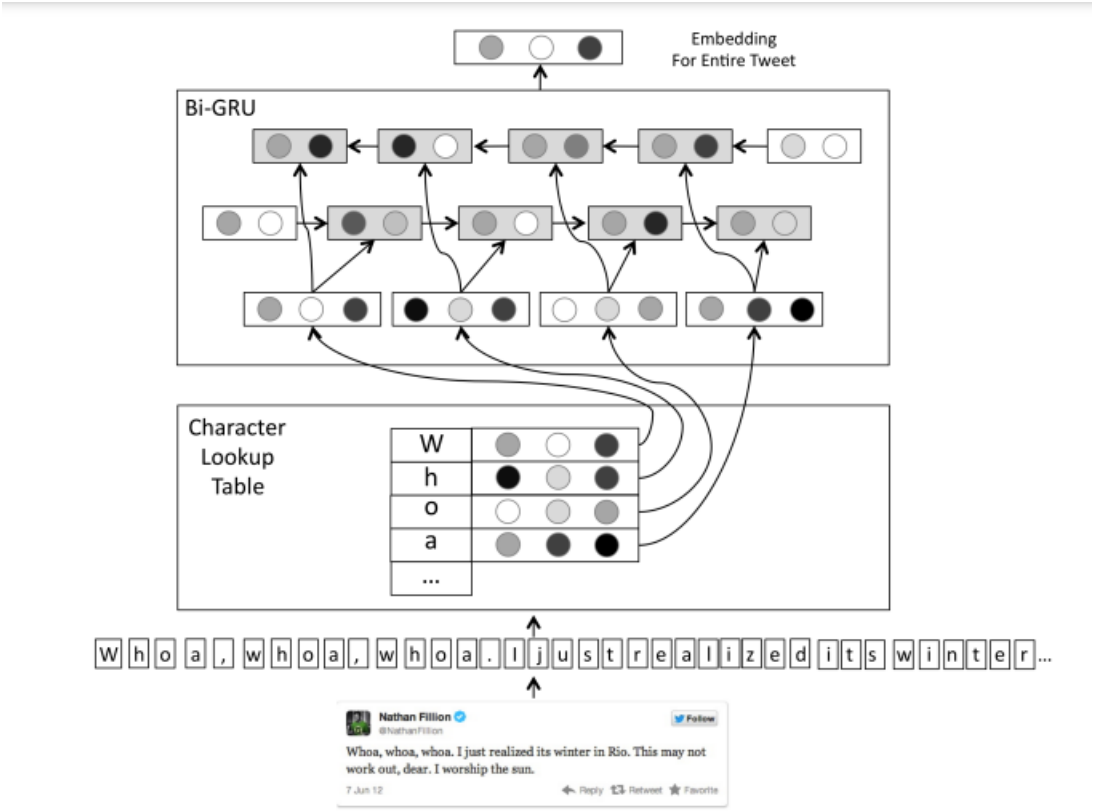

Figura 2.4: Codificador do *tweet2vec* para texto de rede social (DHINGRA et al., 2016).

Este trabalho teve como estudo base o *word2vec*, um modelo que treina uma rede neu-

ral recorrente utilizando palavras como unidade básica, o que gerava vetores de palavras muito extensos. O modelo proposto pelo *tweet2vec* superou o *baseline* implementado em nível de palavra na previsão de *hashtags* associadas às postagens, sendo significativamente mais preciso quando a entrada contém muitas palavras fora do vocabulário ou sequências de caracteres incomuns: enquanto o *baseline* obteve precisão de 20,4% para o conjunto de palavras raras, o modelo proposto alcançou uma precisão de 32,9%.

3

Metodologia

Neste capítulo será descrita a metodologia adotada para alcançar os objetivos descritos na seção 1.3.

3.1 Os Dados

Neste trabalho foi utilizada uma amostra de dados de notas fiscais eletrônicas disponibilizada pelo Ministério Público da Paraíba, que consiste em um arquivo no formato CSV contendo dados de 30 mil NFEs emitidas por empresas fornecedoras para órgãos da administração pública.

Essas NFEs contam com 62 campos de dados, entre os quais se destacam pela relevância para a determinação do produto e do valor do item:

I. NCM_prod: a Nomenclatura Comum do Mercosul é uma convenção de categorização de mercadorias adotada desde 1995 pelo Uruguai, Paraguai, Brasil e Argentina, sendo uma categoria representada por um código numérico que pode agrupar vários produtos;

II. Descricao_do_Produto_ou_servicos: descrição do item vendido, de livre preenchimento;

III. Quant_prod: a quantidade vendida, depende da unidade de medida;

IV. Valor_total_da_nota: valor total da nota fiscal, contabilizando taxas e descontos;

V. Valor_unit_prod: o valor unitário do produto;

VI. Valor_total_prod: o valor total do produto;

VII. Unid_prod: unidade de medida do produto.

3.1.1 Amostra e Rotulação

Os algoritmos de classificação de texto utilizados neste trabalho utilizam como entrada um conjunto de pares <texto, classe>. Para classificar nossas descrições de produtos, foi feita uma rotulação manual de uma amostra dos dados, classificando-as quanto à unidade de medida.

A base de notas fiscais disponibilizada conta com um campo de unidades de medida, o qual nem sempre condiz com a unidade de medida expressa na descrição. Sabendo disso, foi criado um *script* para gerar uma amostra que contivesse todos os valores de unidades de medidas encontradas. Inicialmente, o objetivo era obter aproximadamente 2 mil notas, por isso a seleção ocorreu da seguinte forma: para cada valor único da coluna 'Unid_prod' foram selecionadas aleatoriamente até 40 notas, e foi gerado um arquivo contendo apenas os campos unidade de medida e texto de descrição dos produtos. A amostra obtida continha 1963 NFEs.

Analisando as unidades de medidas descritas nas NFEs da amostra, constatou-se que uma mesma unidade de medida pode ser escrita de várias formas, como podemos ver na Figura 3.1:

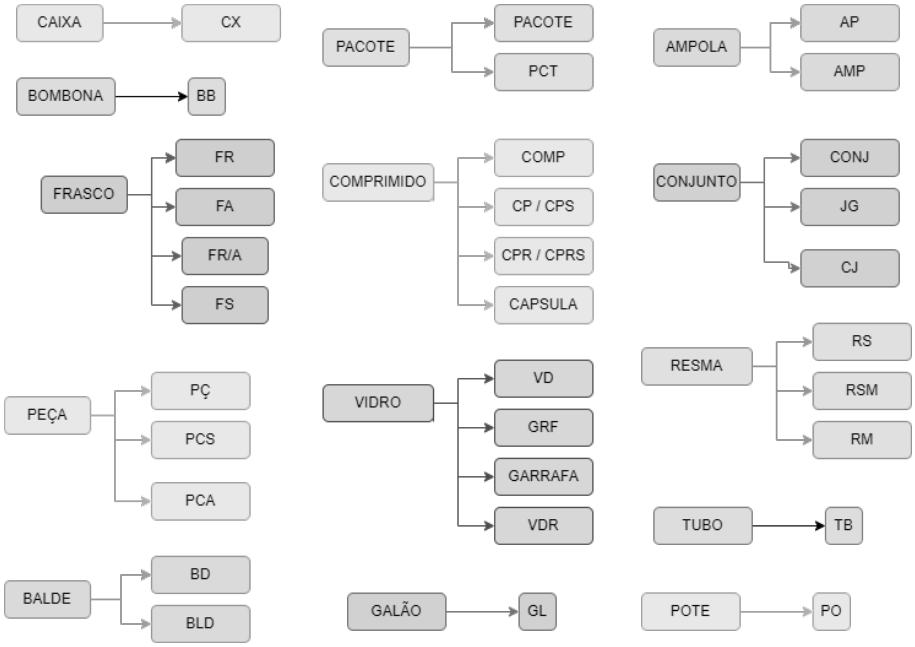

Figura 3.1: Abreviações de algumas unidades de medida.

Assim, adotou-se uma padronização da escrita para cada unidade de medida, que foi aplicada durante a rotulação dos dados.

Na amostra, também foi observado que algumas das descrições dos produtos eram complexas, tendo múltiplos valores de quantidade e unidade de medida, como mostra a Figura 3.2.

Figura 3.2: Exemplos de descrição de produto.

Uma caixa contendo 12 unidades de um produto deve ter um preço aproximado a 12 vezes o valor unitário do mesmo produto. Com isso, durante a rotulação, sempre que possível a unidade mais complexa foi dividida, sendo selecionada apenas a menor unidade presente na descrição. Então, utilizando-se desses critérios, foi realizada a rotulação manual das unidades de medida dessa amostra a partir do texto de descrição do produto. A rotulação manual foi realizada por meio de um editor de planilhas, e das 1963 NFEs da amostra foi possível classificar as unidades de medida a partir da leitura do campo de descrição em 1038, o que representa cerca de 52,8% da amostra.

As quantidades de itens de cada classe de unidade de medida da amostra rotulada, ordenadas em ordem decrescente, são mostradas na Tabela 3.1.

Classe	**Quantidade**
indefinido	925
ml	167
comp	139
grama	132
unidade	120
litro	120
kg	74
metro	59
m3	44
caixa	39
ampola	34
frasco	31
conjunto	27
kit	26
rolo	9
cilindro	9
galao	8

Tabela 3.1: Distribuição das classes na amostra de descrições de produtos.

3.2 Pré-processamento

Como a descrição dos produtos nas notas é de livre preenchimento, podendo conter erros e divergências de representação, é necessário realizar um pré-processamento desses textos antes de realizar uma modelagem, a fim de melhorar a qualidade dos dados. Para tratar os dados dessa amostra, foram realizadas as etapas descritas nas seções seguintes.

3.2.1 Limpeza de texto

Trata os caracteres que podem atrapalhar o processamento do texto. Possui as seguintes etapas:

I. Remoção de acentos nas letras;

II. Converter todas as letras em minúsculas;

III. Remoção de caracteres especiais e pontuação;

IV. Separar números de letras;

V. Conversão de todos os números para um único símbolo.

Como a presença de números é uma informação útil para a determinação de algumas unidades de medida, essa informação foi mantida, e ao invés de remover os números as "cadeias de dígitos" encontradas foram convertidas em um único caractere: "0" na função para a classificação a nível de palavras, e "9" para a classificação a nível de caracteres.

Por exemplo, a função de limpeza de texto para o algoritmo a nível de palavras converteu a descrição "ATADURA DE CREPOM 30CM X 1,8 13FIOS C/12" para "atadura de crepom 0 cm x 0 0 fios c 0".

3.2.2 Remoção de palavras de parada

A biblioteca do python NLTK (BIRD; LOPER; KLEIN, 2009) fornece as palavras de parada (*stop words*) da língua portuguesa, porém algumas dessas teriam impacto negativo se removidas nos textos de descrição devido a algumas abreviações, motivo pelo qual optou-se por utilizar palavras de parada personalizadas. A biblioteca *scikit-learn* (PEDREGOSA et al., 2011) aplica essa etapa no momento de calcular o vetor TF-IDF.

3.3 Modelos de Classificação

Foram treinados dois modelos para realizar a classificação do texto. O primeiro é uma solução tradicional a nível de palavras, adotado como *baseline*; e o segundo é uma solução mais a nível do estado da arte baseado em redes neurais profundas, a nível de caracteres. Os classificadores utilizam um conjunto de entrada contendo textos de descrição de produtos e uma classe que é a unidade de medida presente nesse texto, e com isso devem ser treinados para classificar a unidade de medida de um produto a partir de sua descrição textual.

3.3.1 Classificação com TF-IDF e SVM

O primeiro modelo testado foi uma abordagem baseada em representação vetorial TF-IDF com classificador *Support Vector Machine* (SVM) linear, a nível de palavras. Esta abordagem utiliza o TF-IDF para seleção de *features* e SVM linear para treinamento e predição do modelo. SVM é um dos algoritmos de classificação mais usados na categorização de texto. Ele pode lidar com a classificação de dados lineares e não lineares com a ajuda da função *kernel*, cujo objetivo é mapear os vetores de recursos de um espaço de *features* para um espaço de alta dimensão, buscando a separabilidade linear (FEI; ZHANG, 2020).

Cada valor de unidade de medida representa uma classe. Nessa abordagem, cada classe de unidade foi representada por um número: as classes que tiveram ao menos 9 ocorrências foram convertidas em valores numéricos de 1 a 16, e o restante, assim como as com valor "indefinido", receberam o valor de classe 0.

Como se trata de um problema de múltiplas classes, foi utilizado o classificador com SVM linear da a biblioteca *scikit-learn* (PEDREGOSA et al., 2011), o qual implementa a estratégia *"one-vs-rest"*, a qual divide o problema de classificação com várias classes em um subproblema binário para cada classe. Como dados de entrada para esse classificador utilizou-se um arquivo CSV contendo as amostras previamente rotuladas, mapeadas para categorias numéricas.

Detalhes de implementação

Essa abordagem de classificador foi executada utilizando a versão 3.9 do python, e sua implementação utilizou funções das bibliotecas pandas (The pandas development team, 2020), *scikit-learn* e NLTK.

A metodologia conta com as etapas:

I. Limpeza do texto, onde foi feito o pré processamento dos campos de descrição;

II. Gerar o vocabulário;

III. Gerar o vetor TF-IDF para o conjunto de entrada;

IV. Separar os dados de entrada em conjuntos de treino (60% da amostra) e de teste (40% da amostra);

V. Treinar o SVM;

VI. Validação cruzada: SVMs não fornecem estimativas de probabilidade diretamente, elas são calculadas usando uma validação cruzada. Foi aplicada uma validação cruzada estratificada para que nenhuma classe ficasse de fora, e assim verificar as taxas de acerto do modelo;

VII. Testar o modelo: testamos os classificadores com o mesmo conjunto de validação, a fim de fazer uma comparação dos resultados.

3.3.2 Classificação com *tweet2vec*

É um classificador a nível de caracteres que utiliza um treinamento não supervisionado usando redes neurais recorrentes para classificação, conforme detalhado na Seção 2.7. Esse classificador foi feito para um problema que envolve lidar com textos de redes sociais, que possuem problemas semelhantes aos textos de descrições de produtos. Sua implementação utiliza as bibliotecas python Theano (Theano Development Team, 2016) e Lasagne (DIELEMAN et al., 2015) para gerar e treinar uma rede neural recorrente com Bi-GRU.

Os dados de entrada para o modelo foram obtidos com os seguintes passos:

I. Foram selecionadas as classes de unidade de medida que tinham ao menos 9 ocorrências na amostra, o que representa um total de 16 classes. O restante das descrições foram classificadas como "indefinido";

II. Foi realizado um pré-processamento dos textos, aplicando as etapas da seção 3.2.1;

III. A amostra foi dividida aleatoriamente em conjuntos de treino e teste, a uma proporção de 60% e 40%, respectivamente;

IV. Os conjuntos foram salvos em arquivos de texto, tendo em cada linha o valor de unidade rotulado e o texto pré-processado, de acordo com a entrada esperada para o algoritmo do *tweet2vec*.

Treinamento

O treinamento e teste desse modelo foi executado utilizando a versão 2.7 do python. Embora as bibliotecas solicitem utilizar GPU para otimização do processamento, não foi possível realizar essa configuração, e o treino foi executado em CPU. Foram experimentadas duas abordagens de treinamento distintas: uma utilizando apenas itens rotulados com um valor válido de unidade de medida, e outra utilizando toda a amostra previamente rotulada.

Codificação

Com o modelo treinado, o conjunto de validação foi codificado com esse classificador. O *tweet2vec* originalmente classificava textos com múltiplas classes, por isso esse modelo gera uma lista de possíveis classes, onde a primeira é a classe calculada com maior probabilidade de ser a correta para determinada entrada. Assim, para o nosso problema, foi realizada uma modificação para o resultado da classificação ser apenas a classe de maior probabilidade para cada entrada.

3.3.3 Avaliação

Foi criado um *script* em python para testar os classificadores no conjunto de validação e obter os valores de acurácia, precisão e *recall*, e formar a matriz de confusão. O conjunto de validação consiste em uma amostra aleatória com 786 itens de descrição de NFEs rotuladas quanto à unidade de medida. A saída foi um relatório em formato de texto com as estatísticas do modelo e o gráfico da matriz de confusão.

4

Resultados e Discussões

Neste capítulo serão apresentados os resultados da aplicação de dois classificadores ao problema da classificação de produtos quanto à unidades de medida, bem como as discussões e comentários sobre os mesmos.

4.1 Classificação com TF-IDF + SVM Linear

Esse classificador foi validado através do método validação cruzada estratificada (*stratified cross-validation*), configurado com 5-*folds*, pois não é um conjunto muito grande. Essa validação mostrou uma acurácia média de 78,25% para o modelo. Testando o modelo no conjunto de validação, a acurácia foi 85,35%, e a precisão média foi de 78,96%.

Para esse teste obtemos a matriz de confusão normalizada, conforme a Figura 4.1.

Assim, obtemos as diferentes taxas de erro e acerto para as várias classes de unidade de medida. Podemos fazer as seguintes observações:

I. As classes que possuem uma escrita mais padronizada tendem a ter uma taxa de acerto maior;

II. A lista de unidades estava ordenada em ordem decrescente de número de ocorrências na amostra (conforme a Tabela 3.1), o que mostra que itens com muitas poucas ocorrências no conjunto de treino não foram classificados corretamente, exceto quando muito padronizados;

III. Classes de unidades que costumam estar incluídas em unidades complexas, como grama, litro e unidade, que podem fazer parte de pacotes, caixas ou fardos, tiveram uma precisão menor de acertos. Por exemplo, o produto "nylon mon. preto 3-0 c/ AG. 3/8 3cm cx c/24" deveria ser classificado em unidade, mas foi classificado como indefinido;

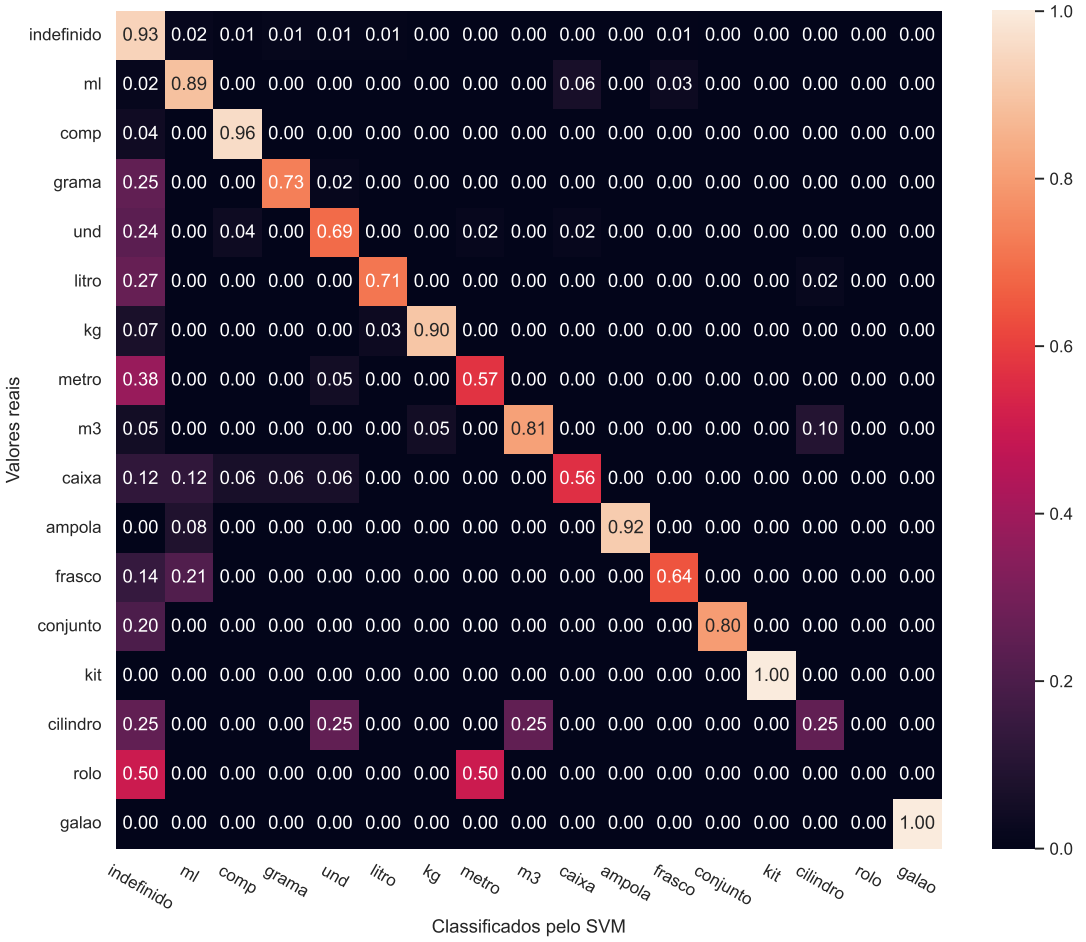

Figura 4.1: Matriz de confusão do classificador com SVM linear.

IV. Produtos com as unidades “rolo” e “cilindro” sendo classificados como “metro” e “m^3” mostram que o padrão encontrado pelo algoritmo foi quanto ao produto, e não à unidade de medida. Isso ocorre pois na base de dados usada os itens com essas unidades de medida, além de aparecerem em menor quantidade, descreviam produtos parecidos. Por Exemplo, "rolo de lã 9cm atlas” possui a unidade "rolo” na descrição, mas foi classificado como metro.

4.2 Classificação com *tweet2vec*

Para esse classificador, foram experimentadas duas abordagens de treinamento distintas: uma utilizando apenas itens rotulados com um valor válido de unidade de medida, e outra utilizando toda a amostra previamente rotulada.

4.2.1 Configuração dos Experimentos

1° Experimento: O modelo foi treinado apenas com os itens de descrição que possuem valor válido de unidade de medida, o que representa 1038 itens. Essa decisão se deu pelo fato de que o algoritmo desse classificador prevê que algumas entradas não podem ser atribuídas a nenhuma classe. Para treinar esse modelo, os conjuntos de treino e teste foram separados a uma proporção de 60% e 40% respectivamente. A esse conjunto de treino, que contém 623 itens, vamos chamar C1. Esse treino foi executado em CPU e levou 20 épocas para chegar ao modelo mais preciso.

2° Experimento: Treinamos o modelo com toda a amostra previamente rotulada. Para treinar esse modelo, os conjuntos de treino e teste foram separados a uma proporção de 60% e 40%, respectivamente. A esse conjunto de treino, que contém 1178 itens, vamos chamar C2. Esse treino foi executado em CPU e levou 15 épocas para chegar ao modelo mais preciso.

O *tweet2vec* conta com a função *evaluate*, que calcula a precisão e o *recall* do modelo gerado. A Tabela 4.1 mostra um comparativo dos modelos gerados com os dois conjuntos de dados:

Conjunto de treino	**Tempo de treino/ época**	**Máxima precisão**	**Recall**
C1	127,4s	54,21%	96,48%
C2	236,64s	69,04%	98,08%

Tabela 4.1: Resultados dos experimentos com o *tweet2vec*.

4.2.2 1° Experimento:

O modelo foi treinado com o conjunto C1. Avaliando esse modelo com o conjunto de validação, a precisão máxima da predição do modelo foi de 54,21%. Testando esse modelo com o conjunto de validação, a acurácia foi de 34,1%, com precisão média de 40,3%.

Isso significa que, usando este modelo para tentar prever unidades de medidas em textos de descrição onde não é possível a classificação, ou seja, aqueles previamente classificados como “indefinido”, mostrou que o algoritmo os classifica com “falsos positivos”, o que significa que ele falha nesses casos. A Figura 4.2 mostra alguns exemplos disso.

unidade classificada	unidade real	descrição
unidade	indefinido	escova de robson
litro	indefinido	sifao univ ssum br 9 astra
comp	indefinido	formula didatico polipac
litro	indefinido	tampa tanque comb rsc int
litro	indefinido	suco de fruta

Figura 4.2: Itens que deveriam ser classificados como “indefinido”.

Embora em sua implementação o algoritmo preveja entradas onde não se pode atribuir classes, na prática isso ocorreu em apenas 1% dos casos no nosso teste, e o modelo se comportou de forma tendenciosa, classificando segundo padrões que modelos a nível de palavras não encontram.

Para o modelo obtido neste experimento, obtemos a seguinte matriz de confusão:

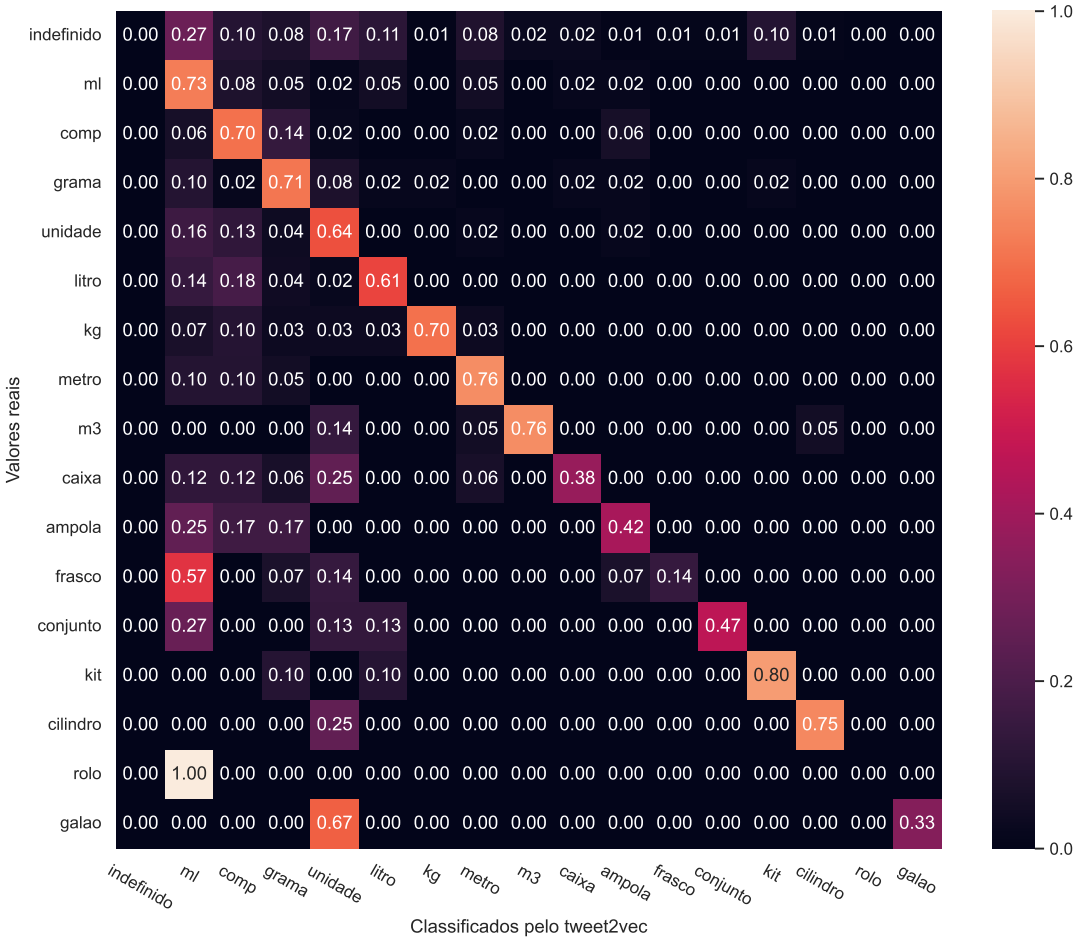

Figura 4.3: Matriz de confusão do classificador *tweet2vec* (1º experimento).

4.2.3 2º Experimento

Experimentamos treinar o modelo com o conjunto C2. Avaliando esse modelo com o conjunto de validação, a precisão média máxima foi 59,44%. Um teste feito com os dados de validação obtiveram acurácia de 69,04%. Para o modelo obtido no segundo experimento, obtemos a matriz de confusão, conforme a Figura 4.4.

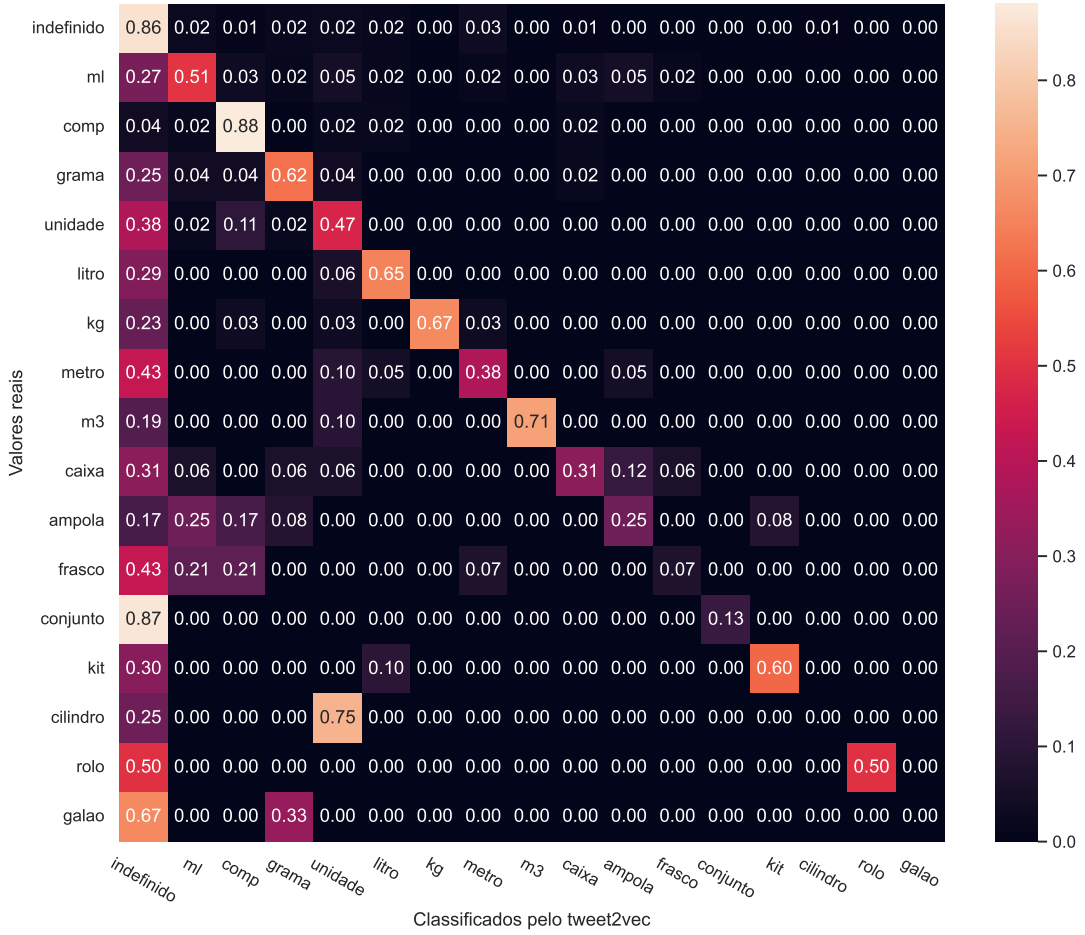

Figura 4.4: Matriz de confusão do classificador *tweet2vec* (2º experimento).

Com isso, podemos observar que esse classificador confunde classes similares com mais frequência, e muitos itens não foram classificados com um valor válido. Ao adicionar a classe “indefinido” ao conjunto de treino do modelo, a classificação passou a contar com muitos “falsos negativos” para outras classes. Em comparação ao primeiro experimento, as taxas de acerto aumentam para algumas classes e diminuem para outras.

5

Conclusão

Assim, o presente trabalho alcançou o objetivo de avaliar técnicas para classificar os itens de produtos de Nota Fiscal Eletrônica (NFE), atribuindo-os às unidades de medidas correspondentes, com base nas descrições textuais dos produtos. Para isto, foram pesquisados e treinados dois modelos para realizar esta classificação: o baseado em representação vetorial TF-IDF com classificador SVM linear, com uma solução tradicional a nível de palavras, e o modelo proposto pelo *tweet2vec*, com uma solução baseada em redes neurais profundas, a nível de caracteres.

O conjunto de dados analisados se limitou a uma amostra de NFE de fornecimento de produtos à administração pública, disponibilizada pelo Ministério Público do Estado da Paraíba (MPPB), da qual foi realizada uma seleção e um pré-processamento adequado para possibilitar a análise pelos algoritmos classificadores de texto mencionados.

Ao testarmos estes modelos para amostra dos dados fornecidos, o classificador a nível de palavras (SVM linear) obteve uma acurácia máxima de 85,35%, e superou as taxas de acerto do classificador a nível de caracteres (modelo *tweet2vec*) para a maioria das classes, enquanto este, configurado para o problema proposto, obteve sua maior acurácia no segundo experimento, alcançando uma acurácia de 69,04%.

Os resultados para o modelo a nível de caracteres nos dois experimentos realizados mostraram o impacto da classe unidade de medida "indefinida" no modelo. Para muitas classes de unidade de medida, o modelo treinado sem essa classe de primeiro experimento obteve taxas de acerto maiores do que no segundo experimento. Como a proposta original do *tweet2vec* era prever *hashtags* em textos, sua implementação provavelmente não previa essa situação, portanto é necessário uma maior compreensão do algoritmo para fazer a correta ponderação das classes.

De acordo com os resultados apresentados, a solução a nível de palavras proposta, em geral, obteve melhores estatísticas para a classificação de produtos. Porém, as quantidades de itens de cada classe em nossa amostra não foram proporcionais, já os dados de teste

foram extraídos de dados reais, e a proporção para a seleção da amostra foi baseada nos valores do campo referente a "unidade de medida" das NFEs, o que nem sempre condiz com o que é representado no campo "descrição do produto".

Apesar de o classificador a nível de palavras ter apresentado melhores resultados, há a necessidade de melhorar esta classificação. Recomenda-se, assim, realizar o treinamento com dados mais balanceados em questão de proporção de itens entre as diversas classes de unidades de medida.

Referências Bibliográficas

ANCHIÊTA, R. et al. PLN: Das Técnicas Tradicionais aos Modelos de Deep Learning. *Sociedade Brasileira de Computação*, 2021.

BARREIX, A. D. et al. *Electronic Invoicing in Latin America: English Summary of the Spanish Document*. [S.l.]: Inter-American Development Bank, 2018. v. 595.

BIRD, S.; LOPER, E.; KLEIN, E. *Natural Language Processing with Python*. 2009. Disponível em: <https://www.nltk.org/book/>.

CGU. *Plano Anticorrupção. Diagnóstico e Ações do Governo Federal*. 2022. Disponível em: <https://www.gov.br/cgu/pt-br/anticorrupcao/plano-anticorrupcao.pdf>. Acesso em: 6 dez. 2022.

CHO, K. et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

CHUNG, J. et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. arXiv, 2014. Disponível em: <https://arxiv.org/abs/1412.3555>.

DHINGRA, B. et al. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*, 2016.

DIELEMAN, S. et al. *Lasagne: First release*. 2015. Disponível em: <http://dx.doi.org/10.5281/zenodo.27878>.

ENCCLA. *XV Reunião Plenária da Estratégia Nacional de Combate à Corrupção e à Lavagem de Dinheiro. Ações de 2018*. 2018. Disponível em: <http://enccla.camara.leg.br/acoes/acoes-de-2018-1>. Acesso em: 6 dez. 2022.

FEI, N.; ZHANG, Y. Movie genre classification using tf-idf and svm. In: *Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City*. New York, NY, USA: Association for Computing Machinery, 2020. (ICIT 2019), p. 131–136. ISBN 9781450376631. Disponível em: <https://doi.org/10.1145/3377170.3377234>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, v. 9, p. 1735–80, 12 1997.

LU, W. et al. A CNN-LSTM-based model to forecast stock prices. In: . [S.l.]: Hindawi, 2020. v. 2020.

MELLO, N. O. de. *Manual Pratico Da NF-e Da NFC-e - Entendendo A*. IOB, 2014. ISBN 9788537922101. Disponível em: <https://books.google.com.br/books?id=tnojvgAACAAJ>.

NETO, A. B. d. S. *Extração de informações mercadológicas a partir de notas fiscais eletrônicas*. Dissertação (Programa de Pós-Graduação em Informática Aplicada) — Universidade Federal Rural de Pernambuco, Recife, 2018. Disponível em: <http://www.tede2.ufrpe.br:8080/tede2/handle/tede2/7867>. Acesso em: 6 dez. 2022.

OECD. *Tax Administration 3.0 – Action Plan Update*. 2022. Disponível em: <https://www.oecd.org/tax/forum-on-tax-administration/publications-and-products/tax-administration-3-0-action-plan-update.pdf>. Acesso em: 6 dez. 2022.

OSORIO, F.; BITTENCOURT, J. R. Sistemas inteligentes baseados em redes neurais artificiais aplicados ao processamento de imagens. 12 2000.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Disponível em: <https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html>.

RAJARAMAN, A.; ULLMAN, J. *Mining of massive datasets*. [S.l.]: Cambridge University Press, 2011.

Receita Federal do Brasil. *Portal Nacional da Nota Fiscal Eletrônica*. 2022. Disponível em: <http://www.nfe.fazenda.gov.br/portal/principal.aspx>. Acesso em: 6 dez. 2022.

RIPLEY, B. D. *Pattern Recognition and Neural Networks*. [S.l.]: Cambridge University Press, 1996.

SEN, P. C.; HAJRA, M.; GHOSH, M. Supervised classification algorithms in machine learning: A survey and review. In: MANDAL, J. K.; BHATTACHARYA, D. (Ed.). *Emerging Technology in Modelling and Graphics*. Singapore: Springer Singapore, 2019. p. 99–111. ISBN 978-981-13-7403-6.

The pandas development team. *pandas-dev/pandas: Pandas*. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, maio 2016. Disponível em: <http://arxiv.org/abs/1605.02688>.

ZHANG, Z. et al. An improved bidirectional gated recurrent unit method for accurate state-of-charge estimation. *IEEE Access*, v. 9, p. 11252–11263, 2021.