UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I
CENTRO DE CIÊNCIA E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANTONIO MARINHO DE ARAUJO NETO

O USO DE PROCESSAMENTO DE LINGUAGEM NATURAL PARA CLASSIFICAÇÃO DE PRODUTOS NO CONTEXTO DE NOTAS FISCAIS ELETRÔNICAS

CAMPINA GRANDE
2021

ANTONIO MARINHO DE ARAUJO NETO

O USO DE PROCESSAMENTO DE LINGUAGEM NATURAL PARA CLASSIFICAÇÃO DE PRODUTOS NO CONTEXTO DE NOTAS FISCAIS ELETRÔNICAS

Trabalho de Conclusão de Curso (Artigo) apresentado ao Departamento do Curso de Ciências da Computação, da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Computação.

Área de concentração: Inteligência Artificial.

Orientador: Profa. Dra. Kézia de Vasconcelos Oliveira Dantas

CAMPINA GRANDE
2021

A663u Araujo Neto, Antonio Marinho de.

O uso de processamento de linguagem natural para classificação de produtos no contexto de notas fiscais eletrônicas [manuscrito] / Antonio Marinho de Araujo Neto. - 2021.

25 p.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia , 2021.

"Orientação : Profa. Dra. Kézia de Vasconcelos Oliveira Dantas , Coordenação do Curso de Computação - CCT."

1. Processamento de linguagem natural. 2. Redes neurais artificiais. 3. Machine learning. 4. Word embedding. I. Título

21. ed. CDD 006.3

Elaborada por Lucas H. A. da Silva - CRB - 15/898 BC/UEPB

ANTÔNIO MARINHO DE ARAUJO NETO

O USO DE PROCESSAMENTO DE LINGUAGEM NATURAL PARA CLASSIFICAÇÃO DE PRODUTOS NO CONTEXTO DE NOTAS FISCAIS ELETRÔNICAS

Trabalho de Conclusão de Curso (Artigo) apresentado ao Departamento do Curso de Ciências da Computação, da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Computação.
Área de concentração: Inteligência Artificial.

Aprovada em 30 de Novembro de 2021.

Kézia de Vasconcelos Oliveira Dantas

Profa. Dra. Kézia de Vasconcelos Oliveira Dantas (DC - UEPB)
Orientador(a)

Sabrina de F. Souto.

Profa. Dra. Sabrina de Figueiredo Souto (DC - UEPB)
Examinador(a)

Paulo B

Prof. Dr. Paulo Eduardo e Silva Barbosa (DC -UEPB)
Examinador(a)

Dedico este trabalho a:
Antonio Marinho de Araújo(in memoriam)
Adair Albuquerque Marinho(in memoriam)

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CBOW	*Continuous Bag-of-Words*
CNN	Convolutional Neural Network
GTIN	*Global Trade Item Number*
LSTM	Long Short-Term Memory
NF-e	Nota Fiscal Eletrônica
NCM	Nomenclatura Comum do Mercosul
PLN	Processamento de Linguagem Natural
RNN	Recurrent Neural Network
SEFAZ-PB	Secretaria de Estado da Fazenda da Paraíba
UEPB	Universidade Estadual da Paraíba

SUMÁRIO

O USO DE PROCESSAMENTO DE LINGUAGEM NATURAL PARA CLASSIFICAÇÃO DE PRODUTOS NO CONTEXTO DE NOTAS FISCAIS ELETRÔNICAS

THE USE OF NATURAL LANGUAGE PROCESSING FOR CLASSIFICATION OF PRODUCTS IN THE CONTEXT OF ELECTRONIC INVOICES

Antonio Marinho de Araújo Neto[1]

RESUMO

A criação da Nota Fiscal Eletrônica (NF-e) facilitou tanto a prestação de tributação quanto a sua fiscalização. As tributações são cobradas de maneira automática, através de códigos presentes na nota. Contudo, nem sempre esses códigos conferem com a descrição do produto, seja por possíveis erros de digitação na ocasião do cadastro dos códigos, ou por fraude fiscal. Dada a quantidade de NF-e emitidas diariamente, é inviável uma checagem unitária. Assim, a Secretaria de Estado da Fazenda da Paraíba (SEFAZ-PB), firmou uma parceria com a Universidade Estadual da Paraíba (UEPB) para a criação de um classificador que fizesse a análise do campo de descrição dos produtos. No entanto, esse campo é de texto livre e não padronizado, sendo necessário o uso de técnicas de aprendizado de máquina em conjunto com técnicas de Processamento de Linguagem Natural (PLN), tal como a vetorização das palavras através de *word embeddings*. Após os testes, a maior acurácia foi de 97,47%, resultado obtido através do uso de uma Rede Neural artificial com LSTM e usando *word2vec* para a vetorização das palavras.

Palavras-chave: PLN. Aprendizado de máquina. Rede Neural Artificial. *Word embeddings.*

ABSTRACT

The creation of the Electronic Invoice (NF-e) facilitated both the provision of taxation and its inspection. Taxation is charged automatically, through codes present in the invoice. However, these codes do not always match a description of the product, whether due to typing errors when registering the codes, or due to tax fraud. Given the amount of NF-e issued daily, a unit check is not feasible. Thus, the State Secretariat of Finance of Paraíba (SEFAZ-PB), signed a partnership with the State University of Paraíba (UEPB) to create a classifier that analyzes the field of description of the products. However, this field is free text and not standardized, requiring the use of machine learning techniques in conjunction with Natural Language Processing (PLN) techniques, such as word vectorization through word embeddings. After the tests, the highest accuracy was 97.47%, a general result of using an artificial neural network with LSTM and using wor2vec for the vectorization of words.

Keywords: NLP. Machine learning.word embeddings. Artificial neural networks.

[1] Aluno de Computação, UEPB, Campus I. E-mail: antonio.97man@gamil.com

1 INTRODUÇÃO

Desde a invenção dos computadores eletrônicos, na segunda metade do século XX, diversas áreas foram informatizadas. Na área fiscal não é diferente e um dos exemplos é a criação da Nota Fiscal Eletrônica(NF-e). Essa digitalização das notas ajudou na declaração e na cobrança desses impostos. Ela é feita através de códigos como o de Nomenclatura Comum do Mercosul (NCM), código numérico para categorias de produtos, aceito em países do Mercosul. Entretanto, em alguns casos, as categorias são mais amplas que as cobrança de impostos.

Na NF-e existe, também, o campo para informar o GTIN (*Global Trade Item Number* - tradução livre Número Global do Item Comercial), um código numérico único para cada produto, podendo ser encontrado embaixo dos códigos de barras, dele, porém, não sendo obrigatória a declaração.

No entanto, nem sempre esses códigos conferem com a descrição do produto, o que pode ser causado por diversos motivos, seja por um erro de digitação na hora do cadastro dos códigos ou, até mesmo, uma tentativa de fraude fiscal. Logo, fazer uma análise do campo de descrição do produto pode contornar esse problema, uma vez que o campo destinado à descrição do produto é de texto livre e não padronizado e o vendedor o preenche como preferir, podendo haver diversas abreviações e siglas, além de eventuais erros de digitação.

Tais problemas são inerentes à linguagem natural pois, ao contrário dos números, os computadores, normalmente, possuem dificuldade para interpretar esse tipo de dado. Desse modo, se faz necessário o uso de técnicas de processamento de linguagem natural (PLN), na qual são desenvolvidos métodos e algoritmos com entrada e saída de textos em linguagem natural humana (GOLDBERG, 2017).

Com isso, a Secretaria de Estado da Fazenda da Paraíba (SEFAZ-PB), em parceria com a Universidade Estadual da Paraíba (UEPB), têm como objetivo desenvolver um classificador com base na descrição textual do produto. Para o desenvolvimento deste, foram testadas redes neurais artificiais, uma técnica de aprendizado de máquina baseada no funcionamento do cérebro (SCHMIDHUBER,2014). Contudo, essas técnicas não conseguem trabalhar com texto como entrada. Para isso, faz-se necessária uma representação numérica das palavras.

O presente artigo é organizado da seguinte maneira: na Seção 2, encontra-se toda a fundamentação teórica do trabalho, contendo a apresentação das técnicas utilizadas para a solução da problemática; na Seção 3, é apresentado a abordagem de desenvolvimento do classificador, sendo mostrado o fluxo e as etapas de desenvolvimento do classificador; na Seção 4, são apresentados os resultados dos testes realizados e, por fim, na Seção 6, são apresentados as conclusões e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão descritas as técnicas utilizadas para construir o classificador proposto em nosso trabalho, bem como os conceitos de aprendizado de máquina e quais as técnicas utilizadas para se trabalhar com texto.

2.1 Aprendizado de máquina

Campo dentro da inteligência artificial, o aprendizado de máquina compreende um âmbito no qual os algoritmos não são explicitamente programados para realizar uma tarefa, utilizando dados para aprender a realizá-la (LIU *et al.*, 2017). A abordagem desse aprendizado varia de acordo com a técnica empregada, podendo ser agrupadas como supervisionadas e não supervisionadas.

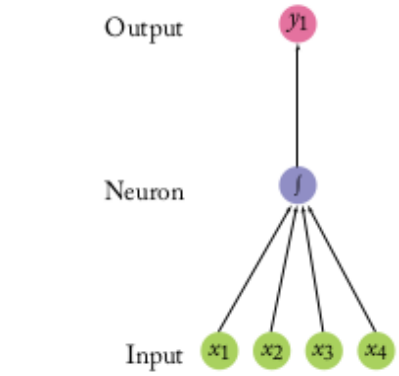
- O aprendizado de máquina supervisionado consiste em algoritmos que necessitam de um conjunto de dados que possuam entradas e saídas marcadas, ajustando valores de condicionais ou variáveis para conseguir achar valores que obtenham o melhor resultado no geral. Esse tipo de aprendizado de máquina resolve problemas que se enquadram em classificação ou regressão. Exemplos de algoritmos desse tipo são: árvore de decisão, regressão logística, regressão linear, entre outros.
- O aprendizado de máquina não-supervisionado, ao contrário do anterior, não precisa das saídas pois, neste caso, as técnicas procuram encontrar um padrão nos dados. Portanto, essa técnica é bastante usada para agrupamento de dados, sendo também utilizada para reduzir a dimensionalidade, tendo como exemplo k-means, t-SNE, PCA, mapas auto-organizados, entre outros.

Como já citado anteriormente, existem diversos algoritmos de aprendizado de máquinas. Porém, nas últimas décadas, com o crescimento de dados não estruturados, como áudios, imagens e textos, as técnicas de redes neurais obtiveram resultados melhores (LIU *et al.*, 2017). Uma vez que esse trabalho tem como objetivo classificar textos de descrições de NF-e, serão utilizadas redes neurais para alcançar esse fim.

2.1.1 Rede Neural Artificial

As Redes neurais artificiais são algoritmos inspirados no funcionamento do cérebro, tendo como base os neurônios, conforme representado na Figura 1, onde vemos as entradas (*input*) e onde cada entrada pode ser multiplicada por um número(l (denominados de peso) que são ajustados durante o aprendizado da rede. Em seguida, os valores resultantes dessa multiplicação são somados, formando os valores de entrada de uma função matemática denominada de função de ativação cujo resultado é a saída do neurônio (GOLDBERG, 2017; SCHMIDHUBER, 2014).

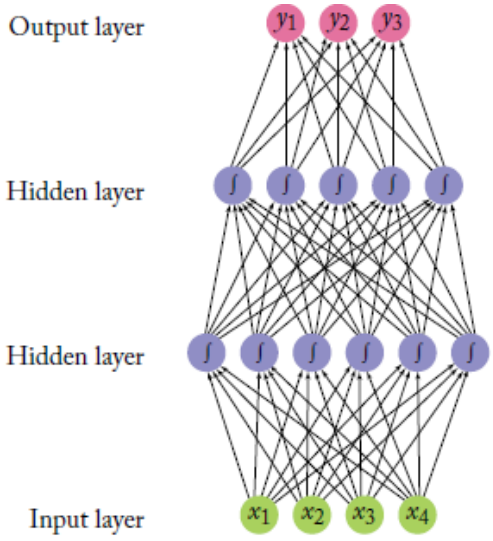
Figura 1- Modelo de neurônio com quatro entradas e uma saída



Fonte: GOLDBERG,2017.

Para formar uma rede neural, os neurônios são organizados em camadas, como mostrado na Figura 2, onde cada círculo representa um neurônio. Quando os neurônios são da mesma camada, eles não interagem entre si, sendo a entrada de uma camada a saída da anterior, com exceção da camada de entrada (*input layer*), onde os dados são fornecidos e a camada de saída (*output layer*), onde sua saída é o resultado. As outras camadas são chamadas de camadas ocultas (*hidden layer*).

Figura 2 - Exemplo de rede neural com 2 camadas ocultas



Fonte: GOLDBERG,2017.

Esse modelo de camada representado na Figura 2, onde todos os neurônios de uma camada estão ligados com os neurônios da camada anterior e da seguinte, é chamado de fully connected (GOLDBERG, 2017; LIU *et al.*, 2017). Durante o aprendizado da rede, normalmente, é utilizado o *backpropagation*, técnica na qual após a passagem de um lote de

dados de treinamento, são feitas classificações e ajustamentos dos valores dos pesos, para classificar a maior parte dos dados corretamente. Tais valores são encontrados através da gradiente (ou alguma variação dela), que é uma derivada da função mínima global, às vezes caindo em mínimos locais da função (GOLDBERG, 2017).

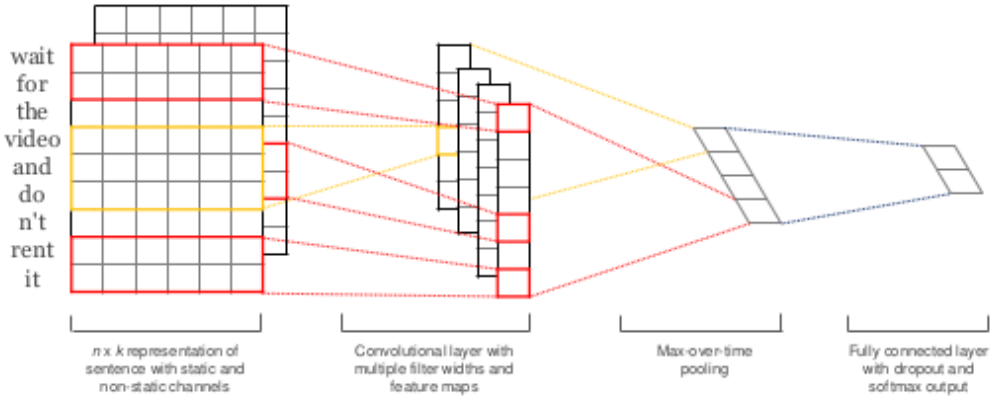
Existem diversas arquiteturas de redes neurais. No presente trabalho serão testadas duas arquiteturas, que são utilizadas em problemas de classificação com textos: as redes neurais convolucionais e as redes neurais recorrentes (GOLDBERG,2017, LIU *et al*, 2017).

2.1.1 Redes Neurais convolucionais

As redes neurais convolucionais (CNN Convolutional Neural Network) são redes cujo propósito é diminuir o tamanho dos dados para encontrar um resultado satisfatório mais rápido. Esse processo se dá porque, antes das camadas intermediárias existem as camadas de convolução, onde são aplicados filtros que retiram características, a camada de *pooling* onde há a diminuição. Depois da camada de convolução podem ficar muitos zeros, além de evidenciar as características, gerando um mapa. (Liu et al.2017; KIM,2014).

Inicialmente, as CNN foram propostas para serem usadas em redes neurais que tinham como entrada as imagens, pois elas se caracterizam por serem grandes, ainda que cada entrada seja um pixel. Desse modo, a redução das imagens de matrizes para vetores menores é de suma importância. Com a utilização de vetores de palavras, as frases ficam agrupadas em matrizes, sendo cada linha representada por uma palavra, o que pode ser observado na Figura 3, que traz o exemplo de uma CNN para texto.

Figura 3- Exemplo de uma CNN para texto.



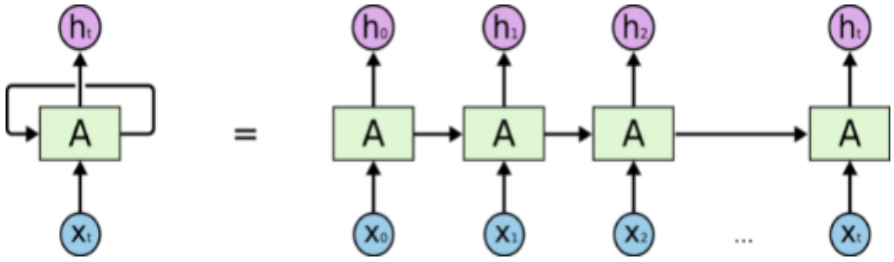
Fonte: KIM,2014.

2.1.2 Redes Neurais Recorrentes

As Redes Neurais Recorrentes (RNN *Recurrent Neural Network*), são arquiteturas de rede que levam em consideração a ordem das entradas e possuem memória, duas coisas das quais as redes neurais tradicionais não dispõem. Para isso, as RNN possuem um modelo de neurônio diferente, que é representado na figura 4. A primeira diferença é que ele faz interações recursivas entre os neurônios da mesma camada, se há frases menores que outras

entre vários 0, como a entrada dos neurônios e, também, há uma unidade de memória que armazena o valor da entrada anterior (Otter, Medina, Kalita, 2019, Colah, 2018), Na Figura 4 temos as interações. O Xt representa a interação, mudando os índices, o Ht representa o valor armazenado do estado anterior.

Figura 4 - Neurônio de uma RNN, com as interações estendidas.



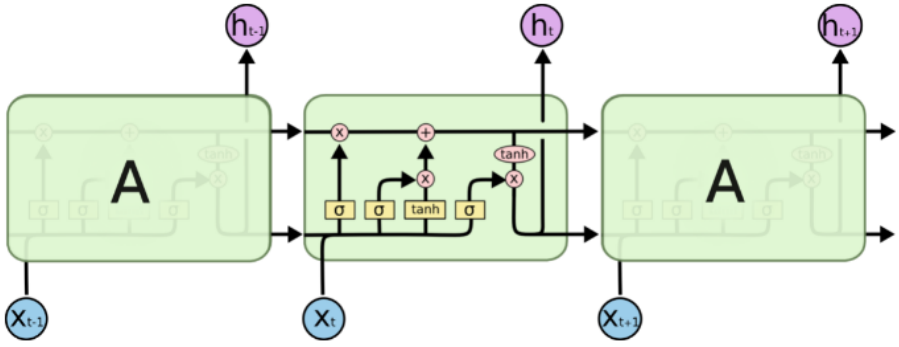
Fonte: Colah,2015.

No PNL, a ordem das palavras, frases e outros elementos faz bastante diferença no significado dos textos e também é útil ter memória dos textos anteriores. Porém, se a informação relevante estiver em frases anteriores, a unidade de memória não terá essa informação, por isso propõe-se uma RNN com memória longa, as Long Short Term Memory.

2.1.2.1 Long Short Term Memory

As RNNs resolvem alguns problemas das redes neurais tradicionais, porém outros ainda permanecem, pois o estado armazenado pode ser uma informação muito anterior e a RNN padrão armazena a informação da última entrada. Além disso, por causa do backpropagation, durante o aprendizado da rede neural, o erro das últimas camadas pode refletir nas primeiras. Este problema é chamado de vanishing gradiente. Para resolvê-lo, foi proposta a LSTM (Long short term memory) (Hochreiter and Schmidhuber, 1997).

Figura 5 - Sequência de neurônios LSTM



Fonte: Colah,2015.

Como podemos ver na Figura 5, ela possui um mecanismos para armazenar a informação que considera mais importante, através de funções *sigmóide* e tangente (*tanh*), e as operações pontuais (representadas através dos círculos cor de rosa), que vão decidir se armazenam ou descartam aquela informação. Por conta dessas características, foram utilizadas LSTM nos testes feitos durante esse trabalho.

2.2 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN), segundo Yoav Goldberg (2017) é definido como a área que desenvolve métodos e algoritmos que tem como entrada ou como resultado a linguagem natural humana, englobando reconhecimento de fala, geração automática de textos, reconhecimento de entidade mencionada, tradução automática de textos, classificação de texto, entres outras aplicações. A linguagem humana é um fenômeno complexo de ser processado, uma vez que ela pode variar, tanto sintaticamente, quanto semanticamente por diversos fatores, seja pela época em que foi escrita, por condições socioeconômicas e pelo ambiente (físico ou virtual).

Um dos pontos mais importantes no PNL quando se precisa usar aprendizado de máquina, é como representar as palavras em números, já que os métodos têm os números como entrada. Isso é feito através de vetores numéricos, pois as palavras não possuem valores mensuráveis, para poderem ser representadas em apenas um número. Uma solução são os vetores *one-hot*, que consistem em vetores do tamanho do vocabulário (o conjunto de todas as palavras únicas) no qual cada posição do vetor representa uma palavra e o valor dele é um, caso apareça na frase e, caso contrário, seu valor é zero (Goldberg,2017).

Todavia, essa técnica apresenta problemas. Não se leva em consideração a semântica, a sintaxe ou a morfologia das palavras. Outro problema é que, grande parte desse vetor é composto de zeros, ou seja, um vetor esparso. Por exemplo, para um vocabulário de 20 com a frase ”CERVEJA GARRAFA 500ML”, teríamos o vetor representado conforme o esquema do Quadro 1, no qual podemos observar que apenas 3 posições são preenchidas com o valor 1 e o restante com o valor zero. Esse problema se intensifica com o aumento da quantidade de palavras no vocabulário. Se ele possuísse 1 milhão de palavras, teríamos três posições com o valor um e o restante com zero, tendo assim um vetor grande para ser processado e armazenado, com a informação em apenas três posições.

Quadro 1 - exemplo de vetor one-hot

CEVEJA	LATA	CANA	GARRAFA	REFRIGERANTI	500ML	1L	375ML	CERVA	ARROZ	BALDE	LATAO	VINHO	VIDRO	700ML	FEIJAO
1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0

Fonte: Elaborado pelo autor,2021

Existem diversas outras soluções para o problema da representação de palavras em vetores, tais como saco-de-palavras (bag-of-word), TF-IDF (*term frequency–inverse document frequency* - frequência do termo-inverso da frequência nos documentos, em tradução livre). Porém, ainda permanecem os principais problemas, pelo fato do *one-hot* ser

um vetor esparso, não levando em consideração nem a semântica e nem a sintaxe das palavras.

2.2.1 Word Embeddings

Em 2013, foi proposto o *word2vec* (MIKOLOV *et al.*, 2013), que consiste em representar palavras dentro de um espaço vetorial de números reais, onde na formação dos vetores das palavras, são levadas em consideração tanto a semântica quanto a sintaxe das palavras, modelos que transformam palavras em vetores densos de números reais, para os quais foram utilizados modelos de redes neurais. Depois da proposta do *word2vec,* vieram outros denominados de *word embedding* (KHATTAK *t al.* 2019).

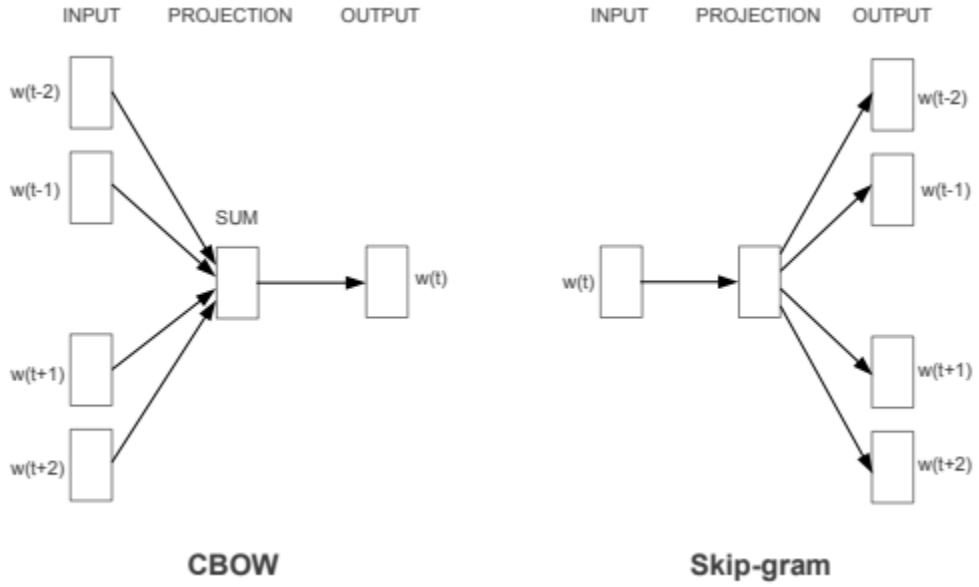
Outros modelos surgiram recentemente, tais como o ELMO e o BERT, que são baseados em redes neurais profundas, com mais camadas e neurônios. Contudo, elas demoram mais tempo para serem treinadas e precisam de mais dados para convoluir. Esses problemas são mitigados utilizando redes já treinadas e *fine-tuning* (PETERS *et al.* 2018, KHATTAK *et al*. 2019), uma técnica que ajusta a rede com novos dados.

Esses modelos novos são denominados de *contextual embeddings*, pois os modelos de *embeddings* tradicionais levam em consideração o contexto para a geração dos vetores, porém é gerado apenas um vetor por palavra, independente dos contextos que aparecerem. Já os c*ontextual embeddings* podem gerar mais de um vetor por palavra, dependendo do contexto. Como neste trabalho o contexto é específico, utilizando apenas os textos de NF-e, não há necessidade de aplicação desses diferentes vetores, além de que teríamos que possuir um modelo previamente treinado, com um contexto parecido em português, para ajustar a rede com o vocabulário usado.

2.2.1.1 Word2vec

Com o objetivo de representar as palavras em vetores densos de forma eficiente, levando em consideração tanto a sintaxe quanto a semântica das palavras, além de poder fazer operações algébricas básicas entre os vetores das palavras, em Mikolov *et al.*,(2013) foram propostas duas arquiteturas de redes neurais curtas para alcançar esses objetivos, a *Continuous Bag-of-Words*(CBOW) e o Skip-gram.

Figura 6 - Representação das arquiteturas do CBOW e Skip-gram



Fonte: MIKOLOV et al,2013

O CBOW tem como lógica de funcionamento predizer o vetor de uma palavra baseado nas palavras próximas à frase. Na Figura 6 acima, temos w(t) como a palavra pela qual será definido o vetor, tendo como entrada os vetores (w(t-1) w(t-2)) e duas posteriores (w(t+1) w(t+2)). Assim, na camada de projeção temos o cálculo do vetor na nova dimensionalidade e também são armazenados todos os vetores resultantes. Quando é calculado um novo vetor para a mesma palavra, se calcula a média entre a representação anterior e a nova (JANG; KIM; KIM, 2019; MIKOLOV *et al.*,2013).

O skip-gram é o inverso do CBOW. Ele tem como entrada a palavra e tenta prover o contexto. Na figura 6 é possível observar essa funcionalidade, sendo a entrada única e a saída múltipla. A predição do contexto está implementada na camada de projeção, onde ficam armazenadas as palavras anteriores, para prever as próximas e também ajustar os valores das palavras anteriores (JANG; KIM; KIM, 2019; MIKOLOV *et al.*,2013).

No skip-gram, como a saída é múltipla, é utilizada uma função de ativação denominada de *softmax*. Nela se calcula a probabilidade de cada saída possível. Porém, isso é bastante custoso em relação ao tempo. Para melhorar o problema, Mikolov *et al.*2013 [2] propôs um modelo alternativo que mede as probabilidades e que tem uma árvore binária, o *hierarchical softmax*, diminuindo o custo de tempo do cálculo da probabilidade para log2 (W), sendo W a quantidade de palavras do vocabulário, na implementação anterior W.

2.2.1.2 FastText

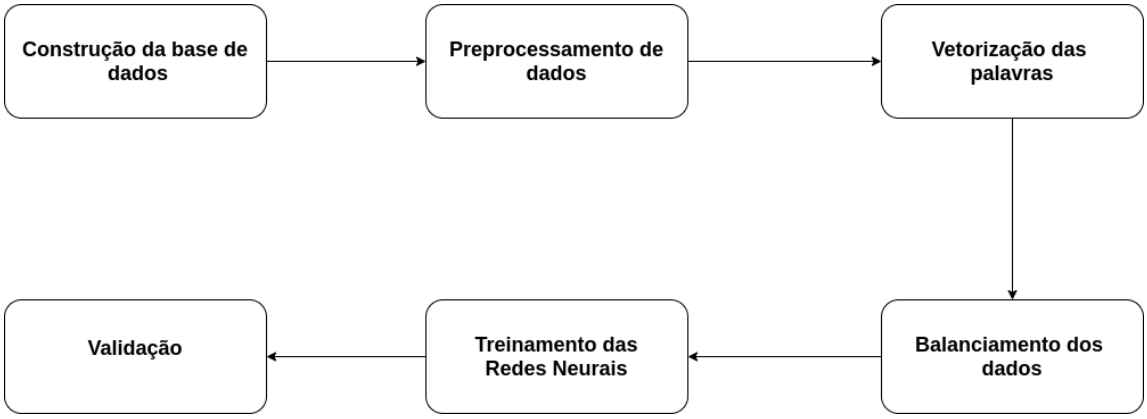
O *FastText* é um modelo inspirado no *word2vec*. Ele é baseado na limitação do word2vec de representar palavras fora do vocabulário, usadas para treinar o modelo. Logo, o

FastText consegue fazer isso, pois na construção dos vetores, para a composição morfológica das palavras (Bojanowski at el. 2016, Khattak at el. 2019), na representação vetorial, além das palavras próximas, é utilizada a composição de caracteres que permitem representar palavras fora do vocabulário, usando aproximações morfológicas, o que para nós é muito útil, pois os produtos, muitas vezes, vem com abreviações ou com erros de escrita.

3 ABORDAGEM PARA CLASSIFICAÇÃO DE PRODUTOS

Destarte, serão descritos os processos para a criação do classificador dos produtos, com base na descrição textual, tendo como etapas a construção da base de dados, o pré-processamento dos dados, a vetorização das palavras, o balanceamento dos dados, o treinamento das redes neurais e a validação, etapas representadas na figura 7 e que nas seções subsequentes serão detalhadas.

Figura 7: Fluxograma das etapas para criação do classificador



Fonte: Elaborado pelo autor,2021

3.1 Construção da base de dados

Ao longo da pesquisa, foram coletados 6,5 milhões de itens de notas fiscais, adquiridas em um parceria com a SEFAZ-PB. Algumas delas foram capturadas de acordo com o NCM nas categorias selecionadas, bem como notas de produtos aleatórios para compor a categoria outros.

Após a aquisição de dados, os produtos requereram uma classificação, pois como descrito na seção 2.2, a técnica usada para a classificação é um algoritmo supervisionado e necessita de uma base de dados previamente classificados. Classificamos, primeiramente, os dados NCMs e depois uma classificação em busca de algumas palavras chaves, que indicassem que o produto não faria parte daquela categoria.

A principal categoria que precisou dessa reclassificação foi a aguardente de cana, pois nos NCMs usados havia outros produtos como rum, whiskey e drinques com aguardente de

cana. Então, foram reclassificados na categoria de bebidas quentes os produtos que tinham subpalavras, tais como ‘whisk’, ‘montil’, ’limao’, entre outras.

Outro produto que teve que ser reclassificado foi a skol beats, que comumente possuía o NCM de cerveja, mas, em reuniões com auditores da SEFAZ-PB, foi definido que esse produto se enquadra na categoria de bebidas quentes. Então, os produtos que possuíam as ‘sense’ ou ‘beat’ foram reclassificados como bebidas quentes.

3.2 Pré Processamento

Neste trabalho, como utilizamos o texto, alguns pré-processamentos são necessários para que, quando vetorizarmos as palavras, não haja vetores diferentes para palavras iguais. Quando se trabalha com dados, mesmo que não sejam textuais, é preciso prepará-los para serem analisados ou utilizar algum algoritmo de aprendizado de máquina.

No presente trabalho, os primeiros processos foram deixar todas as letras em maiúsculo, retirar os acentos, a pontuação e os caracteres especiais(‘@’,’-’,’/’), para que quando ocorresse a vetorização das palavras (seção 3.3), não houvessem palavras iguais, mas que, sem esses processos, teriam vetores diferentes, como por exemplo, duas palavras com a grafia igual, mas uma escrita com letras maiúsculas e outra com letras minúsculas. Os algoritmos de vetorização as entenderiam como palavras diferentes, o que ocorre para todos os outros processos descritos anteriormente.

Foram retirados números e *stop words*, palavras irrelevantes para as análises e aprendizagem de máquinas, tais como artigos, preposições, o verbo ser, entre outros. Foram utilizados os *stop words* da biblioteca NLTK[2], além de ‘C/’, que é uma abreviação para com, pois eles poderiam atrapalhar na classificação, por estarem presentes em muitas das descrições dos produtos e criaram vetores sem muita utilidade.

Foi feito, também, o agrupamento de *bigrams*, a junção de duas palavras, para tentar juntar, principalmente, nomes de marcas, como por exemplo ‘COCA_COLA’, ‘SANTA_CLARA’, entre outros. Para isso, foi utilizada a implementação da biblioteca Gensim[3].

3.3 Vetorização das palavras

Os *word embeddings* são formas de vetorização das palavras, como descrito na seção 2.2.1 e, neste trabalho, foram utilizados os modelos word2vec e FastText, ambos com skip-gram. Eles foram treinados através da biblioteca *gensim*, sendo 100 o tamanho dos vetores resultantes. Para iniciar a análise dos vetores usados, temos na figura 8 os resultados da função ‘*most_similar*’. Nela é passada uma palavra para que se veja quais são as 10 palavras mais semelhantes, através da similaridade de cosseno, uma fórmula bastante utilizada para medir distâncias entre vetores.

[2] http://www.nltk.org/howto/portuguese_en.html
[3] https://radimrehurek.com/gensim/models/phrases.html

Figura 8 - Resultado das palavras mais similares com aguardente, com *word2vec* e *fasttext*

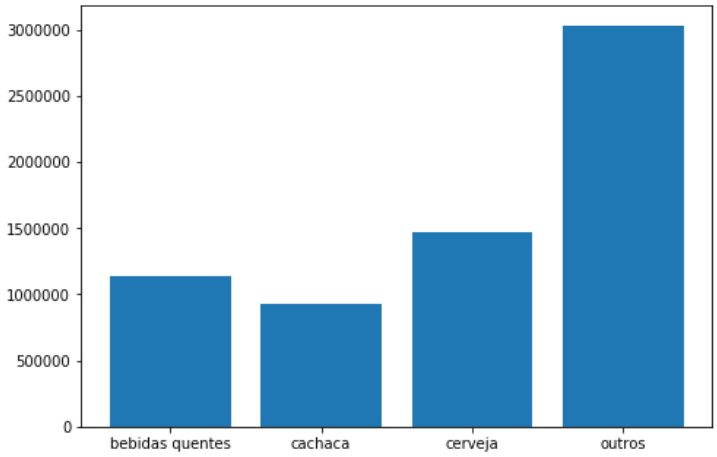
Fonte: Elaborado pelo autor, 2021

No caso mostrado na figura 8, temos as palavras similares a ‘aguardente’. Nos resultados de *fastText*, os *bigrams*, palavras unidas por um subtraço (‘_’), e abreviações, são a maioria, o que era esperado, pois eles têm como base reunir as subpalavras. As outras são sinônimos de aguardente de cana, com grafias diferentes e abreviações. Já no *word2vec*, as abreviações e *bigrams* são menos presentes, mas ele possui mais variações da palavra cachaça, que é um sinônimo de aguardente de cana.

3.4 Balanceamento dos dados

A quantidade de itens de cada categoria dos produtos ficaram desiguais, o que pode ser visto no Gráfico 1, o que ocorre por um motivo bem simples: a existência de mais produtos na categoria “outros” do que na da cerveja, por exemplo. Quando se trabalha com aprendizagem de máquina, isso pode ser um problema, pois se o algoritmo sempre, ou quase sempre, classificar o produto na classe com maior quantidade, vai ter uma taxa de acertos considerável. Isso também dificulta o algoritmo a aprender e existem alguns algoritmos próprios para dados desbalanceados, mas nenhum se enquadra no contexto desse trabalho.

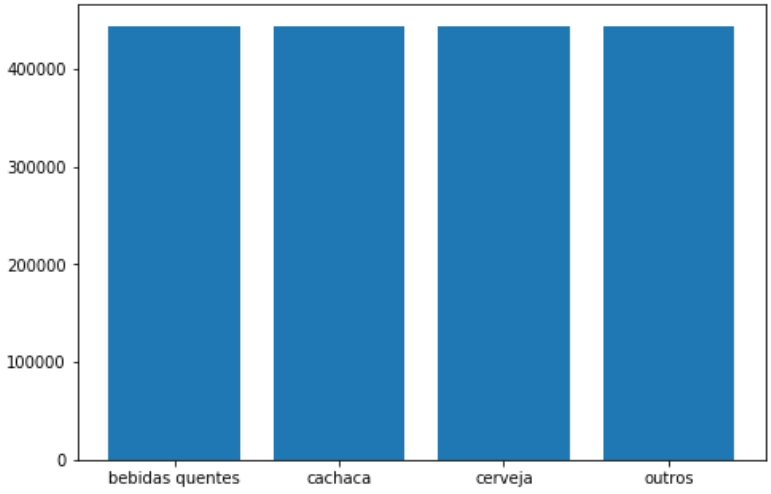
Gráfico 1 - Quantidade de itens em cada categorias



Fonte: Elaborado pelo autor,2021

Na coleta dos dados, foram feitas algumas extrações da base dos NCMs das categorias, mas, mesmo assim, o problema de desbalanceamento persistia, então foram usadas técnicas de reamostragem. Foi feita uma diminuição dos dados classificados como ‘outros’, deixando apenas as descrições únicas, restando 435638 descrições. Mesmo com essa redução, foi preciso um aumento dos dados das outras categorias, para ficarem com a mesma quantidade de produtos da categoria outros, aumento que foi feito de uma maneira semi-aleatória, usando as descrições únicas de cada categoria havendo, assim, um aumento mais uniforme com as descrições, deixando o tamanho de itens nas categorias iguais, como podemos ver no Gráfico 2.

Gráfico 2 - Quantidade de itens em cada categorias, depois do balanceamento



Fonte: Elaborado pelo autor,2021

3.5 Treinamento rede neural

Foram testadas uma CNN e uma rede neural com LSTM. Cada rede foi treinada tanto com o *word2vec* quanto com o *fastText* para além de avaliar os modelos das redes, avaliar, também, os modelos de *word embeddings.*

Para o treinamento delas, foram usadas as bibliotecas do *tensorFlow*[4] e o keras[5]. Além de toda a implementação das estruturas descritas na seção 2.2 e suas subseções, elas também possuem uma camada própria para a entrada de vetores de *word embeddings*, camada de entrada para as duas arquiteturas utilizadas, que consiste em uma matriz de vinte e duas linhas, tamanho de palavras nos produtos, por 100, tamanho dos vetores gerados nos *word embeddings.*

A camada de saída também é a mesma, tendo ela 4 neurônios que utilizam a função de ativação *softmax*, que calcula a probabilidade de cada resultado possível. Assim, cada neurônio nessa camada representa uma categoria e a função dá a probabilidade de ser cada uma delas, classificando-a, depois, como a categoria que tem a maior probabilidade.

A CNN possui filtros e *pooling* de 5, ou seja, ela vai reduzir em 5 vezes a matriz da camada de entrada. Possui, também, 3 camadas ocultas *fully connected*, a primeira e terceira com 128 neurônios e a segunda com 64 neurônios. A arquitetura e os parâmetros utilizados nela foram inspirados no artigo (kim,2014), além de alguns outros estudos, havendo testes menores utilizando menos camadas ocultas e com menos neurônios, até chegar a esses valores.

A rede neural que utilizou LSTM possui bem menos elementos, tendo apenas uma camada oculta que é uma LSTM com 128 neurônios. Porém, mesmo parecendo simples, por conta do neurônio próprio desse tipo de rede descrito na seção 2.2.2.1, elas demoram bastante tempo para serem treinadas, podendo haver testes futuros com mais neurônios.

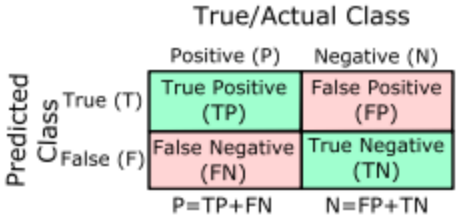
3.6 Validação

O conjunto de dados utilizados foi dividido em 80% para treino e 20% para teste, sendo essa porcentagem separada em 2 partes iguais, cada uma com 10% do conjunto total. Denominadas de validação, elas são passadas junto com o modelo no treinamento, sendo possível analisar se o modelo está com sobreajuste, o que ocorre quando o modelo está se tornando muito específico para a base de treino e está errando em muito casos gerais.

Para avaliar os modelos, foram usadas algumas métricas. Para entendê-las, o primeiro conceito é a matriz de confusão, representada na Figura 9, sendo o caso para um problema binário e, na Figura 10, para 3 categorias. Nela se observa a diagonal principal em verde, cujas classificações estão corretas e as pintadas de vermelho, com as classificações erradas.

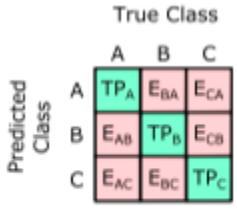
[4] https://www.tensorflow.org/
[5] https://www.tensorflow.org/guide/keras

Figura 9: Matriz de confusão para duas classes



Fonte:THARWAT, 2018.

Figura 10: Matriz de confusão para três classes



Fonte: THARWAT, 2018

A acurácia é uma métrica bastante usada. Nela é avaliada a quantidade de acertos pela quantidade total, ou seja, a soma dos positivos divididos pela quantidade total (THARWAT, 2018). A fórmula abaixo serve para o caso de classificação binária:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Fórmula 1

Precisão e revogação (*recall*) foram métricas utilizadas para avaliar cada categoria separadamente, transformando o problema de múltiplas categorias em um de categorias binárias. Por exemplo, na categoria de cerveja são avaliados os produtos classificados ou que são cervejas, independente das outras categorias. Na métrica de precisão, são avaliados os acertos entre positivos e na revogação são avaliados os positivos, com os falsos negativos. Então, com as duas métricas nas categorias, podemos avaliar onde estão os erros, se há classificações erradas para outra categoria ou se foram classificadas muitas coisas para aquela categoria.

$$PPV = \frac{TP}{FP + TP}$$

Fórmula 2

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

Fórmula 3

4 RESULTADOS

Como descrito na seção 3, foram feitos 4 testes, sendo os 2 modelos de redes neurais, cada um com um word embeddings. A primeira análise feita foi em relação a acurácia. Na base de teste e validação, cujos resultados estão apresentados na Tabela 1, podemos observar que as redes LSTM's tem uma acurácia bem maior que as CNN's em todos os casos e o *word2vec* performou melhor que o *fastText* com uma diferença de cerca de 0,2%.

Tabela 1: Resultados das acurácias no conjunto de teste e validação

	CNN Word2vec	CNN fastText	LSTM Word2vec	LSTM Fasttext
Acurácia teste	73,70%	73,50%	97,68%	97,47%
Acurácia validação	73,99%	73,79%	97,66%	97,46%

Fonte: Elaborado pelo autor,2021

Na Tabela 2 vamos as precisões e as revogações (seção 3.6) para cada categoria da base de teste. Nas redes com a LSTM's, podemos observar onde está a diferença entre o *word2vec* e o *fastText*, na precisão das categorias de 'cerveja' e 'outros'. Assim, podemos concluir que a diferença entre a acurácia do *word2vec* e do *fastText* está na classificação dessas duas categorias. Contudo, é possível observar que, mesmo com a precisão da cerveja no fasttext sendo a menor, a revogação é a mais alta, isso mostra que está havendo pouco falso positivo, o que também é observado no word2vec, mas com uma diferença menor.

Tabela 2: Precisão e Revocação para cada classe

	Precisão Word2vec	Revocação word2vec	Precisão fasttext	Revocação fasttext
Bebidas quentes	97%	96%	97%	95%
Aguardente de cana	98%	98%	98%	98%
Cerveja	96%	98%	95%	99%

Outros	100%	98%	99%	98%

Fonte: Elaborado pelo autor,2021

Um ponto importante é que o conjunto de dados, possivelmente, tem classificações equivocadas, já que, como descrito na seção 3.1, as classificações foram feitas com base no NCM, com pequenos ajustes. No entanto, erros de preenchimento dos NCM ainda refletem no treinamento e na avaliação dos modelos. Este problema foi observado, principalmente, quando analisamos as descrições e classificações de dados de testes. Como o conjunto de dados é muito grande, dificulta um processo de classificação manual em todos os produtos, mas podemos analisar formas para buscar palavras e sub-palavras chaves desses produtos com NCMs errados, para diminuir esse problema sem ter que olhar cada descrição. Todavia, ainda há muita necessidade do trabalho manual.

5 CONCLUSÃO

Apesar de inicialmente acreditarmos que o *fastText* poderia gerar melhores resultados, apesar de inicialmente acreditarmos que o fastText poderia gerar melhores resultados por conta de abreviações e erros de escrita, o word2vec apresentou resultados melhores. O melhor resultado foi encontrado com word2vec e LSTM, alcançando uma acurácia no conjunto de validação de 97,47%, conseguindo criar um classificador com base na descrição das notas fiscais. Esse classificador já está sendo testado pelo auditores fiscais e sendo integrado com o sistema de cobranças automática para ser analisado como performará no ambiente real.

Para trabalhos futuros, pretendemos analisar melhor e testar outros parâmetros de todos os modelos, desde os bigrams e os word embeddings, até as redes neurais (como otimizadores e quantidades de neurônios). Pretendemos, ainda, testar outras arquiteturas de redes neurais ou até combinações como uma LSTM como camada oculta de uma CNN, bem como testar outras formas de balanceamento de dados, buscando uma diversidade maior nas descrições e evitando um possível sobreajuste do modelo.

Outro problema que pretendemos tentar resolver é a extensão da categoria “outros”, cujo universo é muito grande. Há diversos produtos que não estão presentes no conjunto de dados, portanto um objetivo futuro é aumentar a quantidade de categorias a serem classificadas.

REFERÊNCIAS

BOJANOWSKI P.; GRAVE, E. J. A. M. T. **Enriching Word Vectors with Subword Information.** [S.l.], 2016. ArXiv preprint arXiv:1607.04606

COLAH. **Understanding LSTM Networks**, 2018. url

https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

GOLDBERG, Y. **Neural Network Methods for Natural Language Processing.** Morgan & Claypool Publishers.22 de maio de 2017

HOCHREITER, S.; SCHMIDHUBER, J. **LONG SHORT-TERM MEMORY**, 1997.

JANG, B.; KIM, I.; KIM, J. W. **Word2vec convolutional neural networks for classification of news articles and tweets.** PloS one, Public Library of Science San Francisco, CA USA, v. 14, n. 8, p. e0220976, 2019.

KHATTAK, F. K., Jeblee, S., Pou-Prom, C., Abdalla, M., Meaney, C., & Rudzicz, F. (2019). A survey of word embeddings for clinical text. Journal of Biomedical Informatics, 100, 100057.

Kim,Y.**Convolutional Neural Networks for Sentence Classification**.Computation and Language.2014.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E.**A survey of deep neural network architectures and their applications**.Neurocomputing,234, 11-26.2017

Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). **Efficient Estimation of Word Representations in Vector Space**. Em Proceedings of the International Conference on Learning Representations (ICLR 2013)

Mikolov T, I Sutskever, K Chen, GS Corrado, J Dean. **Distributed representations of words and phrases and their compositionality.**Advances in neural information processing systems, 2013

Schmidhuber, Jürgen. **Deep Learning in Neural Networks: An Overview.** Neural Networks 61, 85–117,2014.

Tharwat, A. **Classification assessment methods**, Applied Computing and Informatics, Vol. 17 No. 1, pp. 168-192. 2021