Classificação de Produtos Utilizando Técnicas de *Few-Shot Learning*

Aleson G. S. Chaves [*] **Bruno H. G. Barbosa** [**] **Danton D. Ferreira** [**]
Paulo R. Silva [***] **Sinval T. Nascimento** [***]

[*] *Programa de Pós-Graduação em Engenharia de Sistemas e Automação, Universidade Federal de Lavras (e-mail: alesongsc@gmail.com)*
[**] *Departamento de Automática, Universidade Federal de Lavras, CP 3037, 37200-900, Lavras/MG (e-mails: brunohb@ufla.br, danton@ufla.br)*
[***] *Omnilogic Inteligência S/A (e-mails: paulo.silva@omnilogic.ai, sinval@omnilogic.ai).*

Abstract: E-commerce platforms (marketplaces) receive daily thousands of products belonging to new classes that have not participated in the training process of the algorithm responsible for automating the classification of products. However, it is difficult to constantly update the system with these products, because the cost of retraining the classifiers currently in operation is high due to the large size of the databases. In this sense, the use of product classifiers that use few-shot learning algorithms is an interesting option, as they are capable of being trained using only new classes containing one or few samples per class. Therefore, the k-nearest neighbor (KNN), Matching Networks (MN) and DPGN (Distribution Propagation Graph Network) algorithms were compared in the product classification problem using a database, with 312 classes and 3120 samples, from a marketplace. The tests were performed with k-fold cross-validation, of which the matching networks presented the best result with 93.78% accuracy.

Resumo: As plataformas de comércio eletrônico (*marketplaces*) recebem diariamente milhares de produtos pertencentes a classes novas que não participaram do processo de treinamento do algoritmo responsável por automatizar a classificação de produtos. O retreinamento com estas classes novas é uma necessidade afim de evitar a categorização incorreta nos *marketplaces*. Porém, é difícil a constante atualização do sistema com estes produtos, pois o custo de retreinamento dos classificadores atualmente em operação é elevado devido à grande dimensão das bases de dados. Neste sentido, a utilização de classificadores de produtos que utilizam algoritmos do tipo *few-shot learning* é uma opção interessante, pois são capazes de serem treinados utilizando somente as classes novas contendo uma ou com poucas amostras por classes. Portanto, os algoritmos k-vizinhos mais próximos (KNN), redes *Matching Networks* (MN) e as redes DPGN (*Distribuition Propagation Graph Network*) foram comparados no problema de classificação de produtos utilizando uma base de dados com 312 classes e 3120 amostras, proveniente de um *marketplace*. Os testes foram realizados com validação cruzada do tipo *k-fold*, onde as redes *matching* apresentaram o melhor resultado com 93,78% de acurácia.

Keywords: Natural Language Processing; E-commerce; Machine Learning; Few-Shot Learning; DPGN; Matching Networks; Artificial Intelligence

Palavras-chaves: Processamento de Linguagem Natural; Comércio Eletrônico; Aprendizado de Máquina; Few-Shot Learning; DPGN; Redes Matching; Inteligência Artificial

1. INTRODUÇÃO

Com o desenvolvimento dos recursos computacionais e do acesso à internet por grande parte da população, houve um aumento significativo do comércio eletrônico (Ristoski et al., 2018). Este mercado está se tornando cada vez mais promissor devido a diversos fatores tal como a praticidade de realizar as compras sem sair de casa.

O número de consumidores adeptos ao modelo de mercado *on-line* tem crescido, tal como o número de vendedores e, consequentemente, a quantidade de ofertas estão cada vez maiores, oferecendo uma variedade de produtos (Krishnan and Amarthaluri, 2019). Isto gera uma grande quantidade de dados que em geral encontram-se em linguagem natural não estruturada, havendo a necessidade de realizar a transformação para a linguagem estruturada, afim de possibilitar a gestão de catálogo e outras funcionalidades. Devido ao grande volume de dados, a transformação em dados estruturados se tornou viável graças à aplicação de ferramentas automatizadas de aprendizado de máquina aplicadas à resolução de tarefas de processamento de

linguagem natural (e Silva et al., 2020) (Kannan et al., 2011).

A classificação automática de produtos vem sendo bem solucionada com redes neurais com aprendizado profundo, uma vez que possuem bom desempenho justamente com grande quantidade de dados, que é o caso dos *marketplaces* atuais (Fadlullah et al., 2017). Mas um problema para estes classificadores é a chegada de classes novas de produtos que não passaram pelo treinamento do classificador, sendo necessário constantemente fazer o retreino com as classes novas. Grande parte destas classes possuem poucas amostras, sendo isso um fator limitante para classificadores fundamentados em redes neurais (Chaves et al., 2021a).

A obtenção de classificadores de produtos que levem em consideração novas classes com poucas amostras é uma tarefa complexa, seja pelo custo computacional de retreinar os modelos existentes de forma a contemplar as novas classes, ou seja pela baixa quantidade de amostras dessas novas classes. Vale ressaltar que o número de classes em comércio eletrônico é da ordem de dezenas de milhares e, portanto, o projeto e treinamento do classificador para tal tarefa é bastante complexo. Dessa forma, alternativas que evitem o retreino dos classificadores ao incorporar o conhecimento de novas classes são muito bem vindas (Chaves et al., 2021b).

Nesse sentido, técnicas de *Few Shot Learning* (FSL) são opções promissoras por serem projetadas especificamente para lidar com treinamento de modelos com pequenas quantidades de amostras por classe (Jadon and Garg, 2020) (Wang et al., 2020). As técnicas *few-shot learning* são inspiradas no aprendizado humano que generaliza bem após ter visto alguns poucos exemplos e reconhece variações desses conceitos em percepções futuras (Lake et al., 2011).

O contraste das redes neurais de aprendizado profundo (em relação à quantidade de dados) com o aprendizado humano traz grande atenção para a pesquisa *few-shot learning* (Yang et al., 2020). Em virtude disso, vários algoritmos de aprendizado FSL tem surgido, como: Redes Siamesas (Koch et al., 2015), *Memory-Augmented Neural Networks* (MANN) (Santoro et al., 2016), *Matching Networks* (MN) (Vinyals et al., 2016), *Model Agnostic Meta-Learning* (MAML) (Finn et al., 2017), *Graph Neural Networks* (GNN) (Garcia and Bruna, 2018) e *Distribution Propagation Graph Network* (DPGN) (Yang et al., 2020).

Desta forma, neste trabalho são propostos classificadores de produtos utilizando as redes *Matching* (MN), redes DPGN e o algoritmo *K-Nearest Neighbors* (KNN) que foram testados utilizando dados previamente extraídos a partir do processo de *transfer learning*. As *Matching* (MN), e DPGN por serem redes neurais do tipo *few-shot learning*, são capazes de realizar o treinamento de bases pequenas contendo somente classes novas com poucas amostras. Deste modo, as novas classes poderão ser classificadas sem a necessidade de realizar o treinamento do classificador em operação nos *marketplaces* com todas as ofertas da base de dados antiga.

Esta abordagem tem como benefício manter uma boa acurácia dos classificadores em operação nos *marketplaces*, mesmo com a chegada de novas classes. Também pode reduzir o uso do servidor em nuvem, disponibilizando o mesmo para uso em outras aplicações ou reduzindo o valor do contrato. Manter uma boa acurácia dos classificadores significa ter a plataforma de *marketplace* bem categorizada, com as especificações de produto organizadas e anúncios padronizados em uma estrutura de texto que atenda às necessidades do consumidor. Isto facilita o processo de busca e compra de produtos, fazendo com que o consumidor encontre todos os produtos que deseja e tenha uma boa experiência de compra.

2. REFERENCIAL TEÓRICO

2.1 Few-Shot Learning

Uma pessoa pode identificar a presença de uma outra em uma imagem após ter visto ela uma única vez anteriormente. No entanto, o processo de aprendizagem com a utilização de poucos dados é uma tarefa complexa para as redes neurais tradicionais (Jadon and Garg, 2020). Para o caso de uma pequena base de dados, um algoritmo simples como o KNN (k vizinhos mais próximos) pode ter desempenho melhor do que uma rede neural *Multi Layer Perceptron* (MLP), caso se tenha um bom pré-processamento que leve a uma boa separação entre as classes e a escolha de uma boa métrica para o cálculo da distância (Jadon and Garg, 2020).

As redes de aprendizado profundo conseguem um bom desempenho em uma variedade de tarefas, porém, necessitam de uma grande quantidade de dados para aprender (Zheng et al., 2019). No entanto, existem bases de dados com apenas algumas poucas amostras por classes e isso dificulta o processo de treinamento das redes neurais tradicionais, obtendo desempenho reduzido. Neste caso, é necessário implementar métodos de aprendizagem de máquinas específicos para redes neurais que sejam capazes de realizar o treinamento quando se tem poucas amostras por classe, estes métodos são conhecidos como *Few-Shot Learning* ou *One-Shot Learning* (Wang et al., 2020), (Chaves et al., 2021a).

2.2 K-Vizinhos Próximos

O k-vizinhos mais próximos (KNN) é um dos mais simples e tradicionais algoritmos, que quando utilizado como classificador, possui desempenho competitivo em relação aos mais complexos classificadores da literatura. Trata-se de um dos principais algoritmos utilizados em reconhecimento de padrões, categorização de textos, reconhecimento de objetos, dentre outras aplicações. A grande vantagem do KNN é que o mesmo não depende de treinamento, a sua tarefa se resume em identificar, em um conjunto rotulado, os k objetos mais semelhantes ou próximos (com menores distâncias) da amostra não rotulada (Trstenjak et al., 2014). Portanto, o KNN é uma opção interessante para conjunto de dados com pequena quantidade de amostras. De fato, o KNN pode ser considerado uma técnica de *few-shot learning* uma vez que, a partir do momento que uma ou mais amostras de uma certa classe nova estiver disponível, tal amostra pode ser prontamente adicionada no conjunto de dados, quando passa a estar apta a ajudar na classificação de outra amostra de sua mesma classe, caso isso seja necessário (Chaves et al., 2021b).

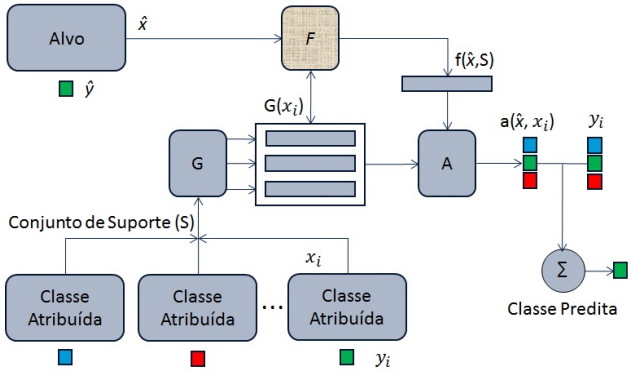
2.3 Redes Matching

As redes *matching* aprendem a mapear uma pequena base de dados de treino e teste em um mesmo espaço de *embeddings* (tarefa *few-shot*). Elas são treinadas para aprender uma representação com os *embeddings* da pequena base de dados de treino de forma adequada, em busca da minimização dos erros do cálculo de similaridade entre o alvo e os representantes de classe. As redes *matching* funcionam com a ideia de um KNN diferenciável com medida de similaridade do cosseno. Além dos *encoders* que formam os *embeddings* estas redes utilizam um mecanismo de atenção para auxiliar na predição, este que consiste na aplicação da função *softmax* na saída da distância do cosseno (Vinyals et al., 2016).

As redes *matching* consistem em cinco principais etapas: (i) Extrator de *embeddings* G, que forma o *encoder* com os dados das amostras; (ii) *Embeddings* de contexto completo F, cujo o objetivo é obter o contexto entre as amostras, quando houver (este é formado por uma rede LSTM bidirecional e o seu uso é opcional); (iii) cálculo de similaridade com a função de distância cosseno c; (iv) mecanismo de atenção A que é dado pela aplicação da função *softmax* na saída da função de cálculo de similaridade c e, por último, (v) a função de perda entropia cruzada (Jadon and Garg, 2020).

A arquitetura com as etapas das redes *matching* descritas anteriormente pode ser vista na Figura 1. O conjunto de suporte (S) (à esquerda da figura) é o conjunto de dados de entrada, como se fossem os representantes (vizinhos) do KNN, sendo que x_i é o vetor de dados, y_i são os *labels* e $\hat{x}$ é o alvo (Chaves et al., 2021b).

Figura 1. Arquitetura das redes *matching*



Fonte: Do Autor (2021)

Os dados do conjunto de suporte formados são encaminhados para o extrator de *embeddings* G, que aprende uma nova representação para as amostras. Depois os dados passam pelo *embedding* de contexto completo F, ou seja, a saída de G e o alvo $\hat{x}$ passam pela rede *BILSTM* representada por F (Sproat and Jaitly, 2016) (Vinyals et al., 2016). Conforme a Figura 1, abaixo de F pode ser observada a equação $f(\hat{x}, S)$ que é a nova representação dos dados do conjunto de suporte (S) e do alvo $\hat{x}$ obtida por F.

O próximo passo antes da aplicação da função do mecanismo de atenção é o cálculo de similaridade utilizando a função do cosseno. A similaridade é calculada entre a nova representação do conjunto de suporte (x_i) e o alvo $\hat{x}$. O mecanismo de atenção $\boldsymbol{A}$ é obtido pela aplicação da função *softmax* na saída da função de similaridade do cosseno (1):

$$A(\hat{x}, x_i) = \frac{e^{c(F(\hat{x}),G(x_i))}}{\sum_{j=1}^{k} e^{c(F(\hat{x}),G(x_j))}} \tag{1}$$

em que c é a função que executa o cálculo de similaridade com a distância do cosseno e e representa a função *softmax*.

A predição é dada por:

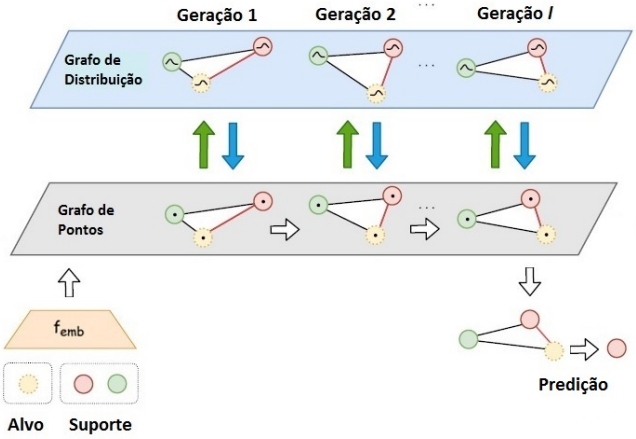
$$P(\hat{y}|\hat{x}, S) = \sum_{i=1}^{k} A(\hat{x}, x_i) y_i, \tag{2}$$

que implementa a combinação linear da função *softmax* da camada de atenção com o vetor *one hot encoded* dos *labels* y_i. Esta equação é uma combinação linear de probabilidades que determina a qual classe o alvo pertence. A função de perda normalmente utilizada é a entropia cruzada.

2.4 Redes DPGN

As redes DPGN (*distribution propagation graph network*) é um tipo de rede de grafo aplicada para a tarefa *few-shot learning*. Esta contém uma dupla arquitetura chamada: "grafo de ponto" (PG) e "grafo de distribuição" (DG). O "grafo de ponto" consiste na inferência no nível de instância e o "grafo de distribuição" é a inferência no nível de distribuição. Desta forma, é processada a representação dos dados em diferentes níveis de forma independente, aproveitando a representação no nível de instâncias e no nível de distribuição. A atualização das redes acontece da seguinte forma: PG gera DG reunindo a relação $1{\times}n$ em todos os exemplos, ou seja, a relação de um nó com toda a vizinhança. Enquanto DG refina PG através da entrega das relações de distribuição. Desta forma, as relações de PG e DG são fundidas ao longo das gerações (Yang et al., 2020).

Figura 2. Arquitetura das redes DPGN



Fonte: Yang et al. (2020)

A arquitetura das redes DPGN pode ser vista na Figura 2, em uma tarefa com duas classes (2-*way*) e 1 uma amostra por classe (1-*shot*). A formação do *encoder* f_{emb} é similar

às redes *matching*, onde são extraídas as características no nível de instância do conjunto de suporte e dos alvos (Chaves et al., 2021a). Estes dados são entregues ao "grafo de ponto" da arquitetura dupla das redes DPGN que realizam ciclos de transformação "ponto para distribuição" (P2D) e "distribuição para ponto" (D2P) ao longo das gerações. A seta verde indica a transformação (P2D) cujo papel é agregar a similaridade de instância para construir a representação no nível de distribuição. A seta azul indica a transformação (D2P) que por sua vez, agrega a similaridade de distribuição a fim de obter características de instância mais discriminativas. No final das gerações as redes DPGN faz a predição do alvo (Yang et al., 2020).

3. MATERIAIS E MÉTODOS

3.1 Base de Dados

A bases de dados utilizada foi cedida por uma empresa parceira e é composta por amostras com textos não estruturados provenientes de plataforma de comércio eletrônico. A base de dados é constituída por 3120 ofertas gerais divididas em 312 classes. A descrição dessa base de dados pode ser encontrada na Tabela 1.

Tabela 1. Base de dados de classes novas

Descrição da Base	Valores
Número de classes	312
Número de amostras por classe	10
Número de amostras	3120
Número de características	1000

Fonte: Do Autor (2021)

Cada oferta em linguagem natural é transformada em um vetor de dados numéricos com 1000 valores, que são as características de cada oferta que serão usadas nos classificadores *few-shot learning*. Esses valores são extraídos por um modelo que encontra-se em operação e que foi previamente treinado com amostras de classes diferentes daquelas utilizadas neste trabalho. Ou seja, a partir de um classificador (Rede Neural Artificial) treinado previamente com um número grande de classes e amostras, são extraídas características (saídas dos neurônios da última camada intermediária da rede neural), em um processo que pode ser interpretado como *transfer learning*.

3.2 Ferramentas: Modelos e Softwares

No desenvolvimento deste projeto foi utilizada a IDE de *python google colab PRO*. A vantagem desta IDE é poder executar os algoritmos na nuvem, não sendo necessária a instalação de bibliotecas e também pode-se utilizar GPU (*Graphics Processing Units*) ou TPU (*Tensor Processing Units*). Ela permite também ter aplicações remotas e com mais de um desenvolvedor ao mesmo tempo, sendo bom para trabalhos em equipe.

Para a implementação dos algoritmos, foram utilizadas diversas bibliotecas *python*. As principais bibliotecas são as seguintes: *pandas* que foi utilizada para trabalhar na manipulação da base de dados; *math* e *numpy* para a realização de cálculos numéricos, tal como cálculos de distâncias no algoritmo KNN; *string* para manipulação de strings, *random* para geração de números aleatórios e *tqdm* para criar barras de progresso afim de possibilitar maior interação com o código.

As redes neurais FSL foram implementadas utilizando as bibliotecas *open source tensorflow* e *pytorch*. As redes FSL foram originalmente aplicadas em bases de dados de imagens da literatura tais como, *omniglot* (Lake et al., 2011) e *mini*ImageNet (Vinyals et al., 2016). Neste trabalho, as redes FSL foram aplicadas na base com dados proveniente de plataforma de *marketplace* com ofertas de produtos descritos em linguagem natural, sendo que estas ofertas tiveram as características previamente extraídas conforme descrito na Seção 3.1.

3.3 Ferramentas de Análises Estatísticas

Para a avaliação de desempenho dos algoritmos FSL implementados neste trabalho, foram utilizadas as medidas de acurácia média e desvio padrão, obtidas no método de validação cruzada k-fold (SCHREIBER et al., 2017). Os resultados dos algoritmos foram comparados utilizando-se o teste-t, adotando-se o nível de 95% de confiança.

O método *k-fold* utiliza um subconjunto da base de dados para compor o conjunto de teste e no conjunto de treino são utilizados k subconjuntos restantes de mesma dimensão que o conjunto de teste. Considerando que as classes da base de dados possuem 10 amostras, foi possível fazer a divisão dos testes em dez *folds*. Em cada *fold*, uma amostra de cada classe é separada para formar o subconjunto de teste e as 9 restantes são utilizadas para compor a base de representantes.

A classificação de produtos empregando as técnicas *few-shot learning* foi realizada utilizando características previamente extraídas das redes da empresa parceira (processo de *transfer learning*). Os dados das amostras descritas em linguagem natural foram recebidos pela redes neurais da empresa parceira que os transformaram em dados numéricos ou características que representam os textos das amostras a serem classificadas. Neste sentido, este trabalho é uma aplicação das redes FSL na tarefa de processamento de linguagem natural, visto que a base de dados antes da extração de características era formada por dados das ofertas de produtos descritas em linguagem natural.

As abordagens estudadas foram o algoritmo KNN, as redes *matching* e as redes DPGN, conforme apresentado nas subseções a seguir. Sendo que o KNN e as redes *matching* foram testadas com 145 e 312 classes da base de dados e as redes DPGN foram testadas com 145 classes. O objetivo inicial era testar todas as redes com 312 classes, mas com os recursos computacionais disponíveis foi possível testar as redes DPGN somente com 145 classes, então o algoritmo KNN e as redes *matching* também foram testados com as mesmas 145 classes para possibilitar comparação entre os três algoritmos.

3.4 Aplicação do Algoritmo k-NN

O algoritmo KNN foi aplicado como classificador *few-shot* para 312 classes da base de dados descrita na Tabela 1. As distâncias utilizadas neste projeto foram: *Euclidiana (ED), cosseno (CD), Manhattan (MD), Hassanat (Has-D),* e *Lorentzian (LD)*. Também foram testados diferentes valores do parâmetro k.

3.5 Aplicação das Redes Matching

Na parametrização das redes *matching*, além dos parâmetros tradicionais tais como os existentes na MLP, existem três parâmetros. O parâmetro "classes por conjunto de suporte" é o número de classes utilizadas no teste de similaridade realizado em cada posição do lote. O parâmetro "amostras por classe" é bem intuitivo, pois é a quantidade de amostras de cada uma das classes que vão compor o lote, é algo similar ao número de representantes do KNN. O lote representa o número execuções de cálculos de similaridades para atualizar a rede.

Para avaliar os parâmetros, foram realizados testes utilizando o primeiro *fold* da base (o primeiro *fold* que é constituído pela primeira amostra de todas as classes da base de dados). Após obtido bons resultados para o primeiro *fold* foram utilizados todos os dez *folds* na obtenção dos parâmetros.

Na parametrização o primeiro passo foi a escolha do número de camadas da MLP do *encoder* G das redes *matching*. Uma vez obtido o número de camadas, o próximo passo foi escolher o restante dos parâmetros das redes *matching*. Os parâmetros que apresentavam bons resultados foram sendo fixados.

As redes *matching*, utilizando 312 classes, foram implementadas empregando os seguintes parâmetros: função de perda de entropia cruzada; otimizador *Adam* e taxa de aprendizagem de 0,0001. Foram utilizadas 312 "classes por conjunto de suporte" e 2 "amostras por classe". O número de épocas de treinamento foi 90 e foi utilizado 12 lotes. Apenas uma camada intermediária com 450 neurônios foi utilizada na arquitetura da MLP do *encoder* G. A função não linear utilizada foi tangente hiperbólica (*tanh*). O *embedding* de contexto completo F não foi utilizado. (Os parâmetros foram ajustados experimentalmente usando o banco de dados de treino, com a seguinte faixa de variação: "amostras por classe" (2 a 8), lotes (4 a 20), taxa de aprendizagem (0,01 a 0,00001). No caso da MLP as faixas de variações dos parâmetros da camada intermediária foram as seguintes: camadas (1 a 5), neurônios (60 a 1000), função não linear (*tanh*, *relu* e *leaky-relu*)).

3.6 Aplicação das Redes DPGN

As redes DPGN apresentam como problema uma alta demanda por memória RAM, devido a diversos cálculos de similaridades realizados e também devido à complexidade da sua dupla arquitetura. Desta forma, não foi possível realizar testes utilizando as 312 classes da base de dados e portanto, o teste das DPGN foi realizado somente com 145 classes. Foram escolhidas para teste 145 classes que apresentou ao menos 1 erro de predição utilizando o KNN com distância do cosseno, k=1, aplicado na base de dados com 312 classes. Esta escolha foi realizada apenas para tornar o problema mais complexo.

Para que a comparação seja realizada na mesma condição das redes DPGN, foram realizados testes com o algoritmo KNN com distância do cosseno (k=1) e redes *matching* com as mesmas 145 classes utilizadas pelas DPGN. A configuração das redes *matching* implementada para comparação com as redes DPGN foi a mesma da seção 3.5, exceto o parâmetro "classes por conjunto de suporte" que foi utilizado 145 classes ao invés de 312.

Na parametrização das redes DPGN, além dos parâmetros tradicionais tais como os existentes na MLP do *encoder* f_{emb}, existem outros parâmetros que estão listados na Tabela 2. As redes DPGN também contém os parâmetros "classes por conjunto de suporte" e "amostras por classe" tal como as redes *matching*. Os parâmetros número de gerações e lote das DPGN foram mantidos fixos devido a necessidade de reduzir demanda por memória RAM. O critério de seleção dos parâmetros das redes DPGN foi similar às redes *matching*, os parâmetros que apresentavam bons resultados foram sendo fixados. Porém, para avaliar os parâmetros foram realizados testes utilizando os dez *folds*.

Tabela 2. Descrição dos parâmetros das redes DPGN

Parâmetros	**Descrição**
Gerações	Número de gerações utilizadas pelas DPGN
Distância	Métrica de distância escolhida
Lote	Número execuções para atualizar a rede
Interações	Número de atualizações
Ajuste de erro	Número de interações para o início do ajuste
lr-adj-base	Taxa que vai decaindo no ajuste de erro

Fonte: Do Autor (2021)

As redes DPGN foram implementadas empregando os seguintes parâmetros: função de perda de entropia cruzada; otimizador *Adam* e taxa de aprendizagem de 0,001. Foram utilizadas 145 "classes por conjunto de suporte" e 2 "amostras por classe". Número de interações foi 800, o início do ajuste de erro foi com 500 interações, foi utilizado 12 lotes e 1 geração. A configuração da MLP do *encoder* f_{emb} utilizada foi com uma camada intermediária com 450 neurônios e função *tanh*, e a saída foi com 128 neurônios. (Os parâmetros foram ajustados experimentalmente usando o banco de dados de treino, com a seguinte faixa de variação: "amostras por classe" (2 a 8), lotes (4 a 20), taxa de aprendizagem (0,01 a 0,00001), interações de (200 a 1400). No caso da MLP do *encoder* f_{emb} as faixas de variações dos parâmetros da camada intermediária foram as seguintes: camadas (1 a 5), neurônios (60 a 1000), função não linear (*tanh*, *relu* e *leaky-relu*)).

4. ANÁLISE E DISCUSSÃO DOS RESULTADOS

4.1 O Algoritmo KNN

Os experimentos com o KNN foram iniciados com 1 vizinho, $k = 1$, com as 312 classes e 3120 amostras da base de dados. A Tabela 3 apresenta os resultados obtidos para diferentes medidas de similaridade. A distância do cosseno apresentou o melhor resultado com 92,56% de acurácia. Além disso a distância do cosseno também é interessante pela possibilidade de implementação de um limiar de decisão de classificação. Com a distância do cosseno pode ser utilizado um limiar único para todas as classes, no entanto se fosse utilizado as outras distâncias estas dependeriam mais da distribuição dos dados.

Com o intuito de encontrar o melhor valor de k vizinhos, foram escolhidas as distâncias do cosseno (CD) e *Manhattan* (MD). A escolha foi devido a distância do cosseno

Tabela 3. Valores da acurácia (k=1) e desvio padrão do KNN para várias distâncias

Distâncias	CD	Has-D	LD	MD	ED
Acurácia	**92,56**	91,31	90,74	89,97	86,73
Desvio P.	1,49	1,70	1,83	1,95	1,96

*Abreviações: Desvio P. = Desvio Padrão
Fonte: Do Autor (2021)

apresentar o melhor resultado e a distância de *Manhattan* que apesar de apresentar resultado inferior em relação as distâncias *Hassanat* (Has-D), *Lorentzian* (LD) foi escolhida por ser uma função mais simples e consequentemente a execução dos testes é rápido. Os resultados encontrados são apresentados na Tabela 4. Pode ser inferido que o valor de $k = 1$ obteve os melhores resultados, pois para $k > 1$ os valores de acurácia diminuíram. Em vista disso, será escolhido o resultado do KNN com $k = 1$ e distância do cosseno para ser utilizado para comparação com os outros algoritmos.

Tabela 4. Valores da acurácia e desvio padrão do KNN (k >1)

Valores (k)	k=1	k=3	k=4	k=5	k=6	k=7
Acurácia CD	**92,56**	91,06	91,44	90,58	90,61	90,48
DP. CD	1,49	2,00	1,89	1,93	1,77	1,53
Acurácia MD	**89,97**	89,55	89,04	87,88	87,79	86,83
DP. MD	1,95	2,28	2,08	1,92	1,48	2,06

*Abreviações: DP. = Desvio Padrão; CD = distância do cosseno; MD = distância de *Manhattan*
Fonte: Do Autor (2021)

Por meio de análise de matriz de confusão do KNN (distância do cosseno e k=1), das 312 classes testadas 57% não apresentam erros de predição e 28% das classes testadas apresentam apenas um erro de predição. O resultado do KNN de mais de 92% é considerando bom também devido ao número de classes testadas ao mesmo tempo, ou seja, 312 classes, visto que o KNN não apresenta treinamento. O índice de acerto do KNN indica que a maioria das amostras estão bem localizadas em seus respectivos agrupamentos, validando o uso do extrator de características utilizado na obtenção da base pelo método de *transfer learning*.

4.2 Redes Matching

Os resultados de acurácia, desvio padrão e o tempo aproximado para realizar o treinamento e teste *k-fold* das redes *matching* (MN) utilizando 312 classes são mostrados na Tabela 5. Também pode ser encontrado o resultado do KNN com $k = 1$ e distância do cosseno para comparação com as redes *matching*. O tempo de processamento (treino e teste) foi obtido por simulação no *google* colab PRO sem a utilização de GPU. Este tempo foi medido considerando apenas 1 dos 10 treinamentos executados com *k-fold*.

Tabela 5. Resultados de acurácia em (%), desvio padrão e tempo de treinamento em horas dos algoritmos FSL utilizando 312 classes

Algoritmo	Acurácia	Tempo
KNN	92,56 +-1,49	NA*
MN	93,78 +-1,33	1,17

*Abreviações: NA = não aplicável
Fonte: Do Autor (2022)

O desempenho das redes *matching* e do KNN foram comparados utilizando o teste-t. Como a hipótese nula não foi verificada, ou seja, existe diferença entre as médias dos resultados, então pode-se concluir que as redes *matching* conseguiram resultado estatisticamente superior ao KNN. Porém, não foi uma tarefa fácil superar o KNN, devido a maioria das amostras da base de dados possuírem características de boa qualidade, o que faz com que o KNN apresente resultado satisfatório. Por outro lado, foi observado nas amostras que apresentaram erros de predição, que muitos destes erros são devidos à erros de rotulação, sendo este montante equivalente a 1,6% da base de dados. E isso faz com as características dessas amostras não sejam de boa qualidade, sendo projetadas em outros *cluters*, causando assim erros de predição.

Dentre os erros de predição, também existem amostras em que o texto de descrição do produto é muito diferente das demais do mesmo *cluster*, e às vezes com texto incompleto. À vista disso, se houver erros de rotulação ou as características sendo de qualidade inferior, mesmo a rede neural MLP, tal como o *encoder* das redes *matching*, não seria capaz de contornar estes problemas do banco de dados. Para trabalhos futuros podem ser utilizados algoritmos para realizar o tratamento dos erros de rótulo, tal como *confidence learning* (Northcutt et al., 2021).

4.3 Redes DPGN

Conforme foi apresentado na Seção 3.6 os testes com as redes DPGN foram feitos utilizando 145 classes. Na Tabela 6, pode ser visto o resultado de acurácia, desvio padrão e o tempo aproximado para realizar o treinamento e teste *k-fold* das redes DPGN. O tempo de treinamento das redes DPGN foi obtido com a utilização de GPU do *google* colab PRO. Este tempo foi medido considerando apenas 1 dos 10 treinamentos executados com *k-fold*. As redes DPGN são bem preparadas para serem treinadas com GPU, pois o treinamento de um *fold* com a utilização de GPU tem duração de 20 minutos, enquanto que com a utilização de CPU este mesmo treinamento apresenta uma duração de 6 horas.

Tabela 6. Resultados de acurácia em (%), desvio padrão e tempo de treinamento em horas dos algoritmos FSL utilizando 145 classes

Algoritmo	Acurácia	Tempo
KNN	85,79 +-3,04	NA*
MN	89,66 +-2,31	1,2
DPGN	90,00 +-1,95	0,3

*Abreviações: NA = não aplicável
Fonte: Do Autor (2022)

O resultado do teste realizado com o algoritmo KNN com distância do cosseno (k=1) e o realizado com as redes *matching* com as mesmas 145 classes utilizadas pelas DPGN também são mostrados na Tabela 6.

O desempenho das redes DPGN e *matching* foram comparados utilizando o teste-t, que mostrou que os resultados foram estatisticamente iguais. Assim sendo as redes *matching* se mostraram a melhor opção para aplicação em comércio eletrônico devido ao fato de não haver tanta demanda por memória RAM, como é o caso das redes

DPGN, permitindo facilmente realizar testes com maior número de classes no conjunto de suporte.

As redes DPGN possuem uma arquitetura sofisticada e que apresentou resultados bem competitivos (no início de 2020) em tarefas *few-shot learning* conforme pode ser visto no trabalho desenvolvido por Yang et al. (2020). Entende-se, portanto, que existe o que melhorar nos resultados destas redes, como a escolha da melhor parametrização. É interessante testar outras configurações dessas redes, mas para isso é necessário tratar os problemas de estouro de memória RAM, buscando uma arquitetura mais eficiente. Isto traz como benefício a possibilidade de realizar treinamento com configurações melhores ou com maior número de classes no conjunto de suporte. Portanto fica como sugestão de trabalhos futuros melhorar o pré-processamento e a alocação de memória RAM das redes DPGN afim de reduzir o consumo. Uma sugestão que também pode ser utilizada é uso de *ensembles* e definição de representantes por classes. Uma boa opção também é testar outras redes de grafos tal como as (EGNN) (*Edge-Labeling Graph Neural Network*) (Kim et al., 2019) e GNN (*Graph Neural Network*) (Garcia and Bruna, 2018).

5. CONCLUSÃO

Visto a necessidade de classificadores de entidades de produtos em tarefas que se dispõem de poucos dados de algumas classes, a abordagem utilizando algoritmos *few-shot learning* (FSL) se mostrou uma boa opção. Estas redes supriram uma lacuna deixada pelas redes tradicionais, estas que são muito utilizadas nos sistemas de comércio eletrônico, porém possuem bom desempenho somente para bases que possuem muitas amostras por classe. Também outro diferencial deste trabalho foi a aplicação das redes FSL para tarefa de processamento de linguagem natural, visto que na literatura as bases de dados normalmente utilizadas são de imagens.

Foram implementadas as técnicas de *few-shot learning* redes *matching*, redes DPGN, e o algoritmo KNN. As redes DPGN e as redes *matching* obtiveram resultados igualmente bons utilizando 145 classes, conseguindo superar o KNN. Apesar das redes DPGN terem obtido um bom resultado utilizando 145 classes, não foi possível realizar o treinamento delas para 312 classes devido ao estouro de memória RAM do *google* colab PRO. No entanto, as redes *matching* foram treinadas também com todas as 312 classes da base de dados obtendo melhor resultado que o KNN, e sem ter nenhum problema relacionado a consumo de memória RAM.

Em virtude disso as redes *matching* se mostraram mais adequadas para resolver o problema *few-shot* abordado que é relacionado a *comércio eletrônico*. Uma vez que, as redes DPGN seja colocado em operação, mesmo que as bases de dados não sejam tão grandes tal como as utilizadas pelos classificadores principais que se encontram em operação atualmente (formado por redes tradicionais), entretanto o número de classes utilizadas que possuem poucas amostras pode ser mais de mil classes. Então para que as redes DPGN possam trabalhar nesta condição elas devem ser otimizadas, ou senão estas redes vão depender de um sistema com memória RAM robusta para serem treinadas.

AGRADECIMENTOS

Os autores agradecem à empresa *Omnilogic Inteligência S/A*, ao CNPq e à FAPEMIG por apoiarem este trabalho.

REFERÊNCIAS

Chaves, A.G.S., Barbosa, B.H.G., Ferreira, D.D., and Nascimento, S.T. (2021a). Extração de entidades de produtos utilizando técnicas de few-shot learning. In *Anais do XXX Congresso de Pós-Graduação da UFLA*. Universidade Federal de Lavras, Minas Gerais, MG, Brasil.

Chaves, A.G.S., Silva, F.C.e., Barbosa, B.H.G., Ferreira, D.D., and Nascimento, S.T. (2021b). Extração de entidades de produtos utilizando técnicas de few-shot learning. In *SBAI 2021 - XV Simpósio Brasileiro de Automação Inteligente*. Sociedade Brasileira de Automática, Rio Grande, RS, Brasil.

e Silva, F.C., de Sousa, R.H., Chaves, A.G.S., Barbosa, B.H.G., and Ferreira, D.D. (2020). Classificador fuzzy-genético aplicado ao processamento de linguagem natural. *Anais do XXIII Congresso Brasileiro de Automática*, 2.

Fadlullah, Z.M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., and Mizutani, K. (2017). State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. *IEEE Communications Surveys and Tutorials*, 19(4), 2432–2455. doi:10.1109/COMST.2017.2707140.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 3, 1856–1868.

Garcia, V. and Bruna, J. (2018). Few-Shot Learning With Graph Neural Networks. In *arXiv*, 1–13.

Jadon, S. and Garg, A. (2020). *Hands-On One-shot Learning with Python Learn*, volume 1.

Kannan, A., Givoni, I.E., Agrawal, R., and Fuxman, A. (2011). Matching unstructured product offers to structured product specifications. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 404–412. doi:10.1145/2020408.2020474.

Kim, J., Kim, T., Kim, S., and Yoo, C.D. (2019). Edge-labeling graph neural network for few-shot learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June, 11–20. doi:10.1109/CVPR.2019.00010.

Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. *Proceedings of the 32nd International Conference on Machine Learning*, 37.

Krishnan, A. and Amarthaluri, A. (2019). Large Scale Product Categorization using Structured and Unstructured Attributes. In *Conference on Knowledge Discovery and Data Mining, August 04–08, 2019, Anchorage, Alaska. ACM, New York, NY, USA*, 9.

Lake, B.M., Salakhutdinov, R., Gross, J., and Tenenbaum, J.B. (2011). One shot learning of simple visual concepts. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, 6.

Northcutt, C.G., Jiang, L., and Chuang, I.L. (2021). Confident learning: Estimating uncertainty in dataset labels.

Journal of Artificial Intelligence Research, 70, 1373–1411. doi:10.1613/JAIR.1.12125.

Ristoski, P., Petrovski, P., Mika, P., and Paulheim, H. (2018). A machine learning approach for product matching and categorization. *Semantic Web*, 9(5), 707–728.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-Learning with Memory-Augmented Neural Networks. *33rd International Conference on Machine Learning, ICML 2016*, 4, 2740–2751.

SCHREIBER, J.N.C., BESKOW, A.L., MÜLLER, J.C.T., NARA, E.O.B., SILVA, J.I.D., and REUTER, J.W. (2017). Técnicas de validação de dados para sistemas inteligentes: Uma abordagem do Software SDbayes. In *XVII Colóquio Internacional de Gestão Universitária*, 1–18. doi:10.1017/CBO9781107415324.004.

Sproat, R. and Jaitly, N. (2016). RNN Approaches to Text Normalization: A Challenge.

Trstenjak, B., Mikac, S., and Donko, D. (2014). Knn with tf-idf based framework for text categorization. *Procedia Engineering*, 69, 1356–1364. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 3637–3645.

Wang, Y., Yao, Q., Kwok, J., and Ni, L.M. (2020). Generalizing from a Few Examples: A Survey on Few-Shot Learning. *ACM Comput. Surv.*, 1(1), 1–34.

Yang, L., Li, L., Zhang, Z., Zhou, X., Zhou, E., and Liu, Y. (2020). DPGN: Distribution propagation graph network for few-shot learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 13387–13396. doi:10.1109/CVPR42600.2020.01340.

Zheng, Y., Wang, R., Yang, J., Xue, L., and Hu, M. (2019). Principal characteristic networks for few-shot learning. *Journal of Visual Communication and Image Representation*, 59, 563–573.