Manoel Aquino Gomes

Classificação de Produtos com Base em Descrições Textuais

Brasil

2021

Manoel Aquino Gomes

Classificação de Produtos com Base em Descrições Textuais

Dissertação apresentada ao Instituto de Computação da Universidade Federal do Amazonas como requisito para obtenção de título de mestrado em Informática.

Universidade Federal do Amazonas – UFAM

Instituto de Computação – ICOMP

Programa de Pós-Graduação em Informática – PPGI

Orientador: Prof. Dr. Edleno Silva de Moura

Brasil

2021

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Gomes, Manoel Aquino
G633c Classificação de produtos com base em descrições textuais / Manoel Aquino Gomes . 2021
50 f.: il. color; 31 cm.

Orientador: Edleno Silva de Moura
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Recuperação da Informação. 2. Machine Learning. 3. Language Models. 4. Ecommerce. 5. Classificação . I. Moura, Edleno Silva de. II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Classificação de Produtos com Base em Descrições Textuais"

MANOEL AQUINO GOMES

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Edleno Silva de Moura - PRESIDENTE

Prof. Altigran Soares da Silva - MEMBRO INTERNO

Prof. Thierson Couto Rosa - MEMBRO EXTERNO

Manaus, 13 de Maio de 2021

Resumo

Muitas aplicações de *e-commerce* lidam com grandes conjuntos de dados de produtos que precisam ser classificados em uma taxonomia predefinida de categorias. Além disso, em alguns cenários práticos, o conjunto de dados é volátil, com novos produtos sendo frequentemente lançados e introduzidos nas categorias existentes. A classificação de produtos tem se tornado uma tarefa essencial para o bom funcionamento de plataformas de vendas em ambientes de *e-commerce*, facilitando a organização e o acesso à informação nos sites das empresas. Nesta dissertação, estudamos e discutimos métodos eficientes e eficazes para a classificação de produtos. Apesentamos uma solução rápida e competitiva para classificação baseada em *Language Models* para classificar produtos e discutimos o uso de um método de classificação proposto na literatura que tem sido usado com sucesso em outras aplicações, o FastText, adaptando-o e estudando-o no cenário de classificação de produtos. Estudamos formas de combinar os métodos propostos à segmentação da descrição de produtos, uma ideia utilizada anteriormente na literatura, e apresentamos experimentos com 3 bases de dados de produtos onde comparamos o desempenho das alternativas estudadas. Os resultados apresentados indicam que tanto o método baseado em *language models* quanto o FastText apresentam resultados qualitativos bem competitivos quando comparados a um modelo de classificação baseado em redes neurais que é considerado estado-da-arte. Os resultados foram obtidos com uma redução significativa nos custos e no tempo de processamento necessários para realizar os experimentos nas 3 bases de dados estudadas.

Palavras-chaves: Classificação de Produtos. Language Models. Comércio Eletrônico.

Abstract

Many e-commerce applications have to deal with a large set of product data that needs to be classified into a predefined product category taxonomy. In addition, in some practical scenarios, the data set is volatile, with new products being frequently launched and introduced in these product categories. Product classification has become an essential task for the good functioning of sales platforms in e-commerce environments, facilitating the organization and access to information on the companies' websites. In this dissertation, we study and discuss efficient and effective methods for product classification. We present a fast and competitive solution for classification based on Language Models to classify products and discuss the use of a classification method proposed in the literature that has been used successfully in other applications, FastText, adapting and studying it in the product classification scenario. We studied ways of combining the proposed methods with product description segmentation, an idea previously used in the literature, and we presented experiments with 3 product databases where we compared the performance of the alternatives studied. The results presented indicate that both the method based on Language Models and FastText present very competitive qualitative results when compared to a classification model based on neural networks that is considered state-of-the-art. The results were obtained with a significant reduction in costs and in the processing time necessary to carry out the experiments in the 3 databases studied.

Key-words: Product Classification. Language Models. E-Commerce.

Lista de ilustrações

Lista de abreviaturas e siglas

RI	Recuperação de Informação
LM	Language Models
AM	Aprendizagem de Máquina

Sumário

1 Introdução

Com a popularização da internet e a facilidade ao acesso da mesma para as pessoas de todas as classes sociais, muito se tem investido no comércio eletrônico para unir empresas a seus clientes. A compra de produtos e serviços sem sair de casa tem se tornado atrativa e tem crescido vertiginosamente nas últimas duas décadas. Em 2019, cerca de 1.92 bilhão de pessoas adquiriram bens ou serviços através de meios digitais, utrapassando 3,5 trilhões de dólares americanos em vendas no varejo eletrônico em todo o mundo [1]. Este fenômeno foi impulsionado pela atual situação de pandemia de COVID-19, onde o isolamento social causou um aumento significativo nas compras online em todo o mundo. Estima-se que 22% das compras em 2023 serão feitas através de comércio eletrônico.

Esse crescimento trouxe consigo alguns problemas ou desafios para o meio científico e empresarial, tais como a geração de uma taxonomia robusta e consistente para os produtos, a eficiência em fazer buscas por produtos de acordo com as consultas dos clientes, e a necessidade de classificação de produtos para facilitar a navegação e a busca por informação nas lojas digitais.

A classificação de produtos exerce papel importante na experiência de usuários em uma plataforma de compras *on-line* porque facilita o acesso a produtos, bem como sua comparação com outros itens da mesma classe. Além disso, a organização da informação em classes também pode ser usada como insumo para melhorar sistemas de busca e recomendação. Se for realizada manualmente, a tarefa de classificação de produtos é uma tarefa que pode demandar das empresas demasiada quantidade de tempo e mão-de-obra. Especialmente em lojas de e-shopping como eBay, Mercado Livre e Alibaba, onde existe um grande fluxo de produtos que precisam ser avaliados e classificados por editores para serem dispostos à venda. Esses custos podem ser minimizados por meio da aplicação de métodos automáticos ou semiautomáticos de classificação de produtos.

Para estruturar e organizar o universo de produtos as empresas de *ecommerce* utilizam taxonomias de produtos. Pode-se definir uma taxonomia como uma árvore de relacionamento hierárquico com classes interligadas verticalmente, onde um nó filho é uma especificação, ou ainda subclasse, do nó pai. A Figura 1 mostra exemplos de itens na base de dados de uma loja de comércio eletrônica (Americanas), bem como as classes às quais eles pertencem, mostrados em forma de árvore com 3 níveis. Além do relacionamento vertical (pai-filho), a árvore de taxonomia também possibilita uma visão de vizinhança (ou relacionamento de irmãos). Por exemplo, na Figura 1, as classes *Suportes Notebook*, *Teclado para Notebook* e *Hubs USB* podem ser consideradas classes irmãs.

[1] https://www.statista.com/topics/871/online-shopping/

A tarefa de classificação de um produto consiste em atribuir uma classe a um novo item a ser vendido na loja. Em nosso trabalho, consideramos que tal item será representado por uma descrição textual curta, o que é um cenário bastante comum em problemas de classificação enfrentados em lojas de comércio eletrônico. Para cada nível da árvore de taxonomia realizamos treinamento de modelos a partir de todas as categorias desse nível, e não somente das categorias filhas da categoria atribuída ao produto no nível anterior.

A tarefa de classificação pode ser bastante difícil em alguns casos até mesmo para um ser humano por diversos fatores. Dentre os fatores que percebemos em nosso trabalho, encontramos pelo menos 3 que dificultam o desenvolvimento de classificadores: i) Produtos que podem se encaixar em mais de uma classe e sobre os quais muitas vezes é difícil até mesmo para um ser humano ter certeza da classe correta. Por exemplo, na Figura 1 encontramos o produto *Blu-ray A Aventura Pelos Recifes de Corais* pertencendo à classe *Loja Pet Love*, mas esse produto poderia também ser enquadrado em *Papelaria*. ii) Por erros de escrita na descrição do item que podem levar o classificador a decisões errôneas. Por exemplo, o produto *Holy Week In Tome* está atrelado à classe *Rock Internacional* em nosso exemplo e parece estar com grafia trocada (*Tome* no lugar de *Rome* ou *Home*), o que pode afetar o processo de classificação automática. iii) Produtos com descrição muito vaga ou muito curta. Por exemplo, um DVD de música sertaneja que vem descrito simplesmente como DVD, o que torna impossível a tarefa de classificá-lo corretamente apenas com base em sua descrição. Como a descrição em muitos sites como E-bay ou Mercado Livre é feita pelas próprias pessoas que vão vender o produto, o sistema fica suscetível a esse e outros tipos de erro na descrição.

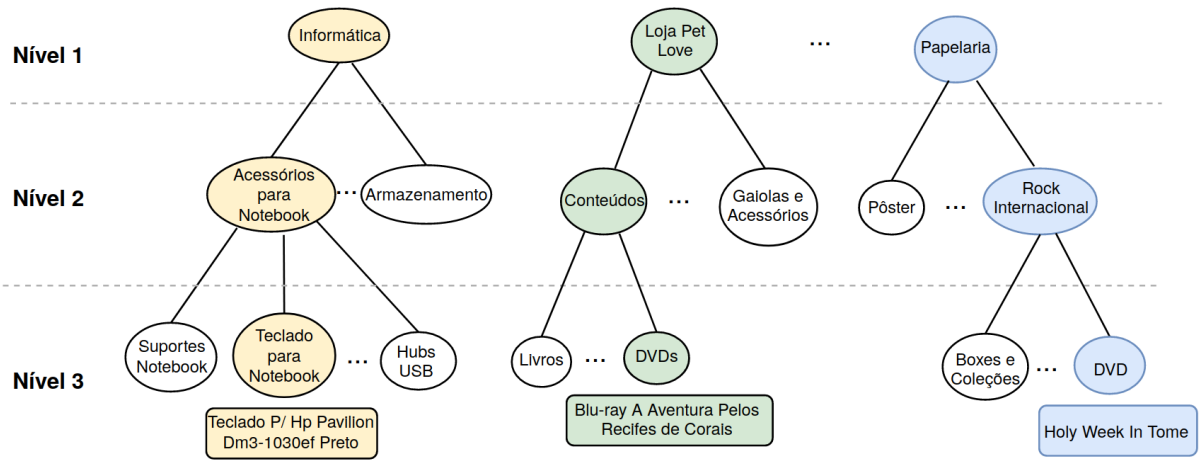


Figura 1 – Exemplo árvore de taxonomia com 3 níveis

Os algoritmos encontrados na literatura para classificação de itens descritos textualmente são baseados em técnicas de Aprendizagem de Máquina, onde aprende-se um modelo a partir de uma base de treino e posteriormente utiliza-se tal modelo para classificar novos itens inseridos na base. O aprendizado se faz através de reconhecimento de

padrões e características descritivas gerais contidos nos dados de treino. Pode-se utilizar regras de isolamento de dados para separar dados de classes distintas (como SVM e Árvore de Decisão), cálculo de similaridade entre elementos para verificar a qual classe pertence uma instância nova (KNN), ajustes de hiper-parâmetros em estruturas complexas a partir de análise de assertividade nos dados de treino (Redes Neurais), entre outras abordagens. A classificação automática de itens baseada em descrições textuais dos mesmos é um problema clássico de estudo que vem sendo abordado por diversos autores na literatura.

A classificação de produtos é um caso particular do problema, e várias soluções seguindo a abordagem de aprendizagem de máquina têm sido propostos na literatura, incluindo-se métodos que utilizam estratégias baseadas em vizinhança, como o KNN (Hu et al., 2018), técnicas probabilísticas, como o Naive Bayes probabilístico (Cortez et al., 2011) e SVM (Support Vector Machines) (Baeza-Yates and Ribeiro-Neto, 2011) ou mais recentemente modelos baseados em redes neurais (Yu et al., 2018; Chen et al., 2019; Joulin et al., 2016; Skinner, 2018). A classificação de produtos apresenta normalmente alguns desafios. Dentre eles, pode-se citar: o fato da descrição disponível sobre cada produto ser em geral muito sucinta, o que muitas vezes dificulta a identificação de sua classe; a enorme quantidade de classes existentes normalmente nesta aplicação; e a ocorrência frequente de discrepâncias significativas na quantidade de elementos das classes disponíveis para treino dos classificadores (Yu et al., 2018).

Apesar desses problemas, há peculiaridades que podem ser exploradas para a construção de um bom classificador de produtos. Por exemplo, normalmente há na taxonomia usada pelas lojas uma disposição hierárquica com que as classes de produtos estão estruturadas, o que abre a possibilidade das relações entre elas como informação extra na tarefa de classificação (Baeza-Yates and Ribeiro-Neto, 2011; Shen et al., 2011). As descrições de produtos também têm propriedades especiais como, por exemplo, a relação entre a posição de determinados termos e a classe à qual o produto pertence, que pode ser usada para melhorar o desempenho de classificadores (Cortez et al., 2011).

Nesta dissertação estudamos especificamente a adaptação e aplicação de dois modelos para o problema específico de classificação de produtos. O primeiro modelo estudado é o uso de Language Models para a tarefa de classificação de produtos. O segundo é a aplicação de uma técnica que modela o problema de classificação de textos utilizando representação de fragmentos de palavras (do inglês *subwords*) (Bojanowski et al., 2016) combinada à utilização de redes neurais para resolver problemas de classificação, a qual foi proposta em um método conhecido como FastText (Joulin et al., 2016).

1.1 Hipóteses de Pesquisa e Objetivos

O propósito dessa dissertação consiste em verificar se os dois modelos estudados são competitivos quando aplicados à tarefa de classificação de produtos. Para isso, implementamos e adaptamos tanto Language Models quanto o método FastText para serem aplicados à tarefa de classificação de produtos. Experimentamos os dois modelos, juntamente com a aplicação do método de segmentação das descrições textuais de produtos proposto por (Cortez et al., 2011) com o intuito de analisar se há impacto positivo no resultado final. Ao modelo que aplica *Language Models* com segmentação demos o nome de LangCat e ao modelo que aplica FastText com segmentação demos o nome de FastText-SEG. Fizemos experimentos para ajustar os modelos às características específicas do problema de classificação de produtos. Ao final, comparamos seu desempenho com métodos propostos na literatura e desenvolvidos especificamente para classificação de produtos que representam o estado da arte na solução do problema.

Ao decidir experimentar os dois modelos, nossa principal hipótese era de que a utilização dos mesmos em tarefas de classificação poderia gerar soluções competitivas quando comparadas a trabalhos anteriores na área. A hipótese surgiu porque ambos os modelos têm sido usados com sucesso em outras aplicações. As duas técnicas obtiveram bons resultados, em termos de qualidade e tempo de processamento, em outras aplicações de busca e classificação.

O método FastText foi escolhido para o estudo por ter se mostrado eficiente em classificação de textos de modo geral, obtendo bons resultados tanto em qualidade como em velocidade. A abordagem de *Subword Information* contorna eficientemente a ausência de termos na base de treino, o que pode ser útil principalmente quando temos muitas classes que possuem baixa quantidade de exemplos no treino. A aplicação de segmentação de texto junto com o modelo FastText é perfeitamente factível e tende a melhorar os resultados finais. Não encontramos na literatura trabalhos que buscassem aplicar Language Models ou FastText, juntamente com segmentação de texto, em processo de classificação de produtos. Por isso investigamos a aplicabilidade desses dois modelos. Assim, temos aqui o objetivo específico de combinar o uso de tais classificadores com as ideias de segmentação de textos propostas em (Cortez et al., 2011), onde concluiu-se que o uso de segmentação pode melhorar a qualidade de classificadores de produto.

Para validar nossa hipótese, comparamos os modelos com dois baselines encontrados na literatura que aplicam técnicas distintas de aprendizagem de máquina: TopCat (Cortez et al., 2011) que utiliza uma abordagem probabilística (rede bayesiana) e RakModel (Skinner, 2018) que utiliza Redes Neurais com camadas LSTM e técnicas de *ensembling*. Utilizamos métricas de MicF1 e MacF1 para na comparação dos modelos.

1.2 Contribuições

Como principais contribuições desse trabalho, podemos destacar:

- A análise do comportamento de Language Models com suavização e segmentação de texto no contexto de Classificação de Produtos, com o estudo da aplicação de languagem models ao problema de classificação de produtos, incluindo uma análise da influência de parâmetros de suavização do modelo ao resultado obtido pelo classificador;
- Análise da aplicabilidade do modelo FastText, juntamente com segmentação de texto como método para classificação de produtos.

Além das contribuições principais, procuramos responder também a questionamentos sobre quais as melhores alternativas existentes hoje em dia para a classificação automática baseada em descrição textual para bases de dados de produtos. Vimos que, dependendo do tamanho da base e da variabilidade das classes na taxonomia de produtos, podemos escolher modelos de maneira mais assertiva. Por exemplo, modelos mais complexos atendem melhor a bases de dados maiores e com grande diversidade de classes, enquanto modelos mais simples são mais apropriados para bases pequenas e com menor quantidade de classes.

2 Conceitos Básicos

Este capítulo tem objetivo o de apresentar o referencial teórico necessário para a compreensão das técnicas e métodos abordados neste trabalho.

2.1 Classificação de Produtos

A classificação de produtos consiste na tarefa de relacionar um produto às classes que ele pertence. Para isso utiliza-se um catálogo onde se encontram todas as classes às quais o produto pode pertencer, de acordo com a taxonomia utilizada no processo de classificação. Para representar um produto em um sistema de classificação pode-se utilizar dados textuais referentes ao produto (como descrição e comentários de compradores) (Vandic et al., 2018; Yu et al., 2018), dados visuais (imagem do produto) (Ristoski et al., 2018), metadados inerentes ao produto (como fabricante e preço) (Wolin, 2002), dentre outros.

A definição da tarefa de Classificação de Produtos pode ser sumarizada da seguinte forma: Dado um conjunto de atributos de produtos e uma taxonomia, o objetivo é construir um modelo de *Machine Learning* que busque encontrar padrões em dados rotulados com o intuito de inferir, com boa assertividade, as classes pertencentes a novos produtos não rotulados (Kozareva, 2015). Um Sistema de Classificação de Produtos pode seguir duas abordagens diferentes em relação ao número de classes atribuídas a um produto: *unioutput* (cada produto é atrelado a somente uma classe por nível da árvore de taxonomia (Beneventano and Magnani, 2004)) ou *multioutput* (a um mesmo produto podem ser atribuídas mais de uma classe (Shankar and Lin, 2011; Yu et al., 2018)).

A figura 2 mostra um fluxo representativo da dinâmica de obtenção, classificação e inferência de produtos em ambiente de comércio eletrônico. Os dados de produtos utilizados em treino e validação de modelo podem ser obtidos através de lojas de comércio eletrônico, por meio de API para transferência de dados, ou por meio de *Crawler* dedicado a captar informações de produtos em urls de compra online (Cortez et al., 2011). Após a coleta deve-se extrair as características (ou *features*) necessárias para o treinamento e validação do modelo. Com um modelo de *Machine Learning* já treinado, pode-se fazer inferências online (em tempo real) ou em *batch* (de tempos em tempos) para os novos produtos que são incluídos na base de dados da loja. Esses novos produtos, com suas classes já inferidas e validadas, podem ser adicionados à base rotulada para utilização em retreinamento do modelo, formando assim um processo cíclico e incremental de aprendizagem.

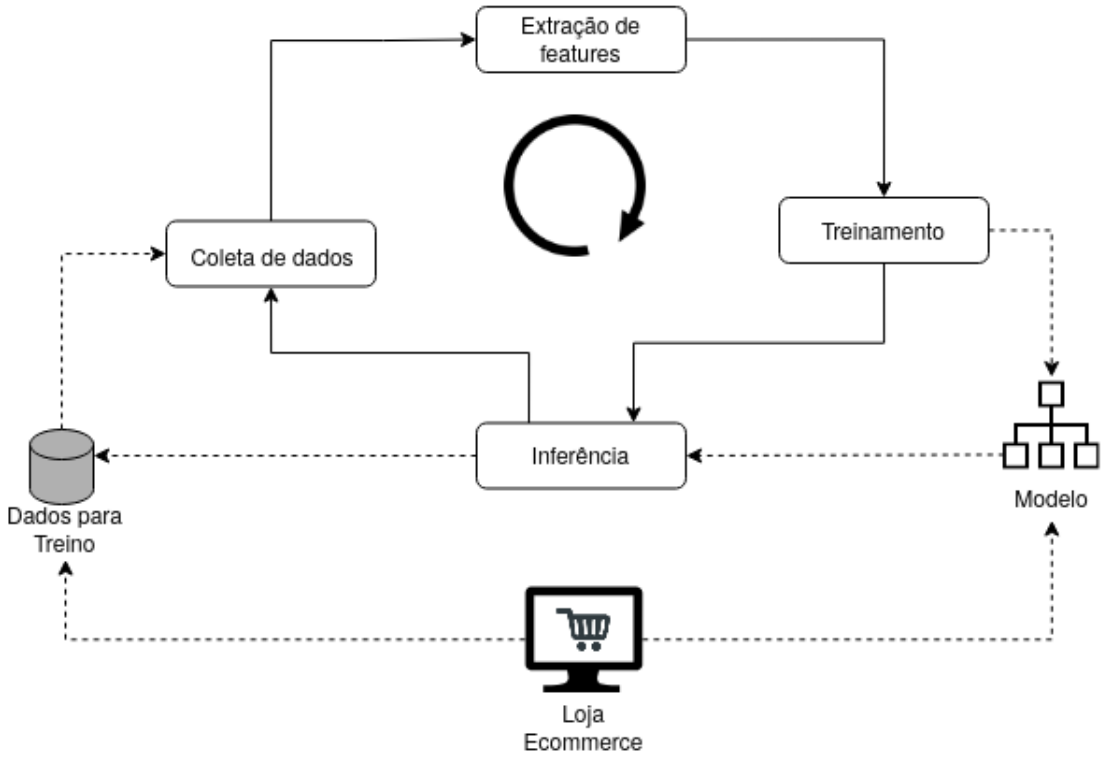


Figura 2 – Arquitetura de classificação em ambiente de Ecommerce

2.2 Language Models

Language Models são usados para modelar a probabilidade de ocorrência de um ou mais símbolos em fontes textuais. Utilizam-se meios probabilísticos para verificar a distribuição de ocorrências de palavras numa base de dados. O modelo calcula a probabilidade de um ou mais símbolos serem gerados por uma fonte textual. Tal probabilidade é calculada a partir das ocorrências das palavras em exemplos de textos produzidos a partir de tal fonte.

Os *Language Models* são inspirados em trabalhos relacionados à teoria da informação (Shannon, 1948) e têm sido aplicados ao longo dos anos na solução de diversos problemas, tais como reconhecimento de fala, compressão de dados, tradução de texto, correção de erros de digitalização em sistemas de OCR (Reconhecimento Ótico de Caracteres), entre outros.

Nesta dissertação, vamos usar *Language Models* para modelar classes de produtos descritas de forma textual. A forma mais simples de modelar uma fonte de dados textual utilizando *Language Models* é considerando cada palavra como sendo independente das demais. Ou seja, para um item $W = W_1W_2...W_n$, onde cada W_i é uma palavra que ocorre em W, a probabilidade de W ser gerado por uma fonte textual é dada por $P(W_1W_2...W_n) = P(W_1)P(W_2)...P(W_n)$.

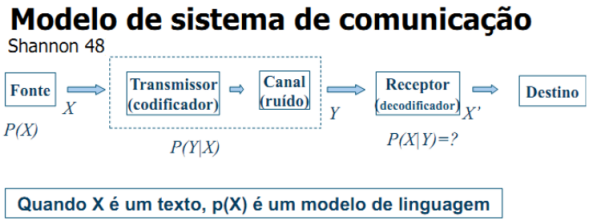


Figura 3 – LM aplicado em Sistema de Comunicação

Language Models têm sido bastante estudados e empregados no contexto de sistemas de busca. Vamos partir de seu uso nesse contexto como referência para resolver tarefas de classificação de produtos. Por essa razão, vamos descrever a seguir o seu uso em sistemas de busca.

Quando usado em sistemas de busca, onde deseja-se saber a probabilidade de uma dada consulta ser gerada por um documento, tal probabilidade é portanto calculada como:

P(Q=$\{w_1,...,w_m\}$|D) = p(w_1|D) x ... x p(w_m|D), Onde w_i é um termo de Q e m é o número de termos em Q.

A função P(w|D) pode ser obtida pela contagem do número de ocorrências do termo w no documento D. Ou seja, Se w aparece 10 vezes no documento D, que tem 100 termos, a função p(w|D) = 0,1. A equação 2.1 sumariza a probabilidade de um texto Q pertencer a um documento D, considerando $f(w_i|D)$ a frequência do termo $w_i \in Q$ em relação ao documento D.

$$p(Q|D) = \prod_{i=1}^{m} p(w_i, D) \tag{2.1}$$

$$p(w_i|D) = \frac{f(w_i|D)}{|D|} \tag{2.2}$$

Um grande problema com o uso de frequência de termos em documentos para calcular a probabilidade é que a probabilidade de um termo que não aparece no documento é dada por 0, o que não é razoável para sistemas de busca que procura englobar todo um vocabulário, mesmo com palavras que só aparecem em alguns documentos.

Para resolver tal problema, usa-se técnicas de suavização (em inglês *smoothing*). Uma técnica de suavização bastante simples é a suavização por adição. Ela consiste em somar uma constante c a cada palavra do vocabulário. Esse método é demonstrado na equação 2.3, onde V representa o número de termos distintos presentes no documento D. Por exemplo, pode-se adicionar a constante 1 à frequência de cada palavra em cada documento modelado. Com isso, mesmo palavras que não ocorrem no documento terão

frequência de ocorrência não nula, igual a 1, e portanto probabilidade não nula.

$$p(w_i|D) = \frac{f(w_i|D) + c}{|D| + c \times V} \tag{2.3}$$

2.2.1 Suavizações estudadas por Zhai e Lafferty

(Zhai and Lafferty, 2004) apresentam um estudo sobre suavização que trata do aperfeiçoamento da acurácia de *Language Models* para modelar fontes de dados textuais, ajustando o estimador de máxima verossimilhança. O princípio de máxima verossimilhança trata o problema de estimação baseado nos resultados obtidos por uma amostra, onde se quer determinar qual o modelo (ou estimador), dentre todos aqueles definidos pelos possíveis valores de parâmetros, com maior possibilidade de ter gerado tal amostra. Ou seja, Zhai e Lafferty buscaram encontrar o modelo que melhor estima a probabilidade de relevância de documentos em coleções de referência estudadas por eles.

No contexto de Sistemas de Busca, a suavização é realizada adicionando uma probabilidade diferente de zero a palavras que não são vistas no documento D, que é tipicamente tomada como sendo proporcional à frequência geral da palavra em todos os documentos. A fórmula geral de um modelo suavizado é demonstrada na equação 2.4.

$$p(w|D) = \begin{cases} p_s(w|D), & \text{se palavra w é vista no documento D} \\ \alpha_d \; p(w|T), & \text{caso contrário} \end{cases} \tag{2.4}$$

Onde $p_s(w|D)$ é a probabilidade suavizada de uma palavra vista no documento D, $p(w|T)$ é o modelo de linguagem gerado a partir das ocorrências de w em T (todos os termos da coleção) e α_d é um coeficiente dependente de D que controla a massa de probabilidade atribuída a palavras não vistas.

De acordo com (Zhai and Lafferty, 2004), temos:

$$\log p(t|D) = \sum_{i:f(w_i;D)>0} \log \frac{p_s(w_i|D)}{\alpha_d \; p(w_i|T)} + n. \log \; \alpha_d + \sum_{i=1}^{n} \log p(w_i|T). \tag{2.5}$$

Métodos de suavização individuais diferem no que tange à estimação de $p_s(w|D)$. Como mostrado em (Zhai and Lafferty, 2004), no contexto da *web* o método *Dirichlet prior* tem boa performance quando se trata de consultas pequenas formadas por poucas palavras-chaves. Nessa abordagem os valores de $p_s(w|D)$ e α_d são apresentadas como (2.6).

$$p_s(w|D) = \frac{f(w, D) + \mu p(w|T)}{|D| + \mu}, \; \alpha_d = \frac{\mu}{\mu + |D|} \tag{2.6}$$

Finalmente, define-se o modelo de linguagem de um termo sobre a coleção de documentos $p(w|T)$ como diretamente proporcional à frequência geral de w em todos os documentos:

$$p(w|T) = \frac{f(w,T)}{|T|} \tag{2.7}$$

2.3 Redes Neurais

Redes neurais são modelos de aprendizado baseados na estrutura do cérebro humano, onde cada elemento de processamento (ou neurônio artificial) possui uma importância na decisão final do modelo. Geralmente sua organização envolve centenas de neurônios separados por camadas, onde neurônios de camadas adjacentes são interligados, passando informações de contexto entre si. Em cada ligação de neurônio existe um peso atrelado que definirá o grau de importância da mensagem passada entre os neurônios (Agatonovic-Kustrin and Beresford, 2000). A figura 4 mostra a estrutura básica de um neurônio artificial (comumente chamado de Perceptron). Para cada valor x_i transmitido por um neurônio da camada anterior é aplicado um peso w_i, o qual atribuirá maior ou menor importância ao valor x_i. Todos os valores de entrada são somados observando as ponderações (pesos) e então aplica-se uma função de ativação, gerando um valor y que será propagado para os neurônios da próxima camada ligados ao neurônio atual.

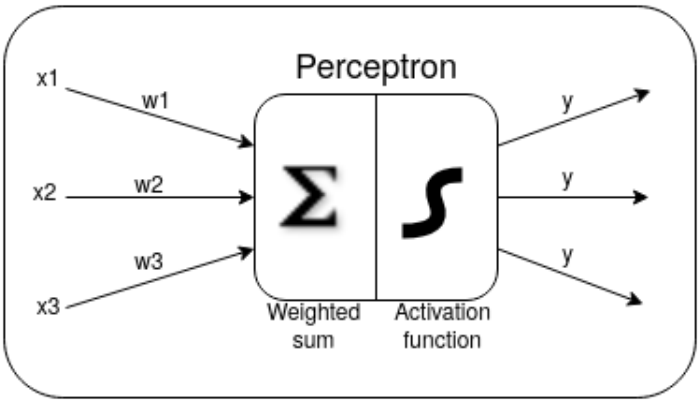


Figura 4 – Representação de um neurônio artificial - Perceptron

A figura 5 mostra um modelo simples de Rede Neural, composta basicamente por 3 partes: o vetor de entrada com valores numéricos que representam a instância ou objeto a ser analisado; as camadas intermediárias, onde os neurônios são dispostos e interligados; e a camada de saída que conterá os resultados do processamento da rede sobre os valores de entrada.

O processo de treinamento consiste em ajustar o conjunto de pesos (ou o vetor de pesos) de cada camada de maneira a otimizar a detecção de padrões dos dados de

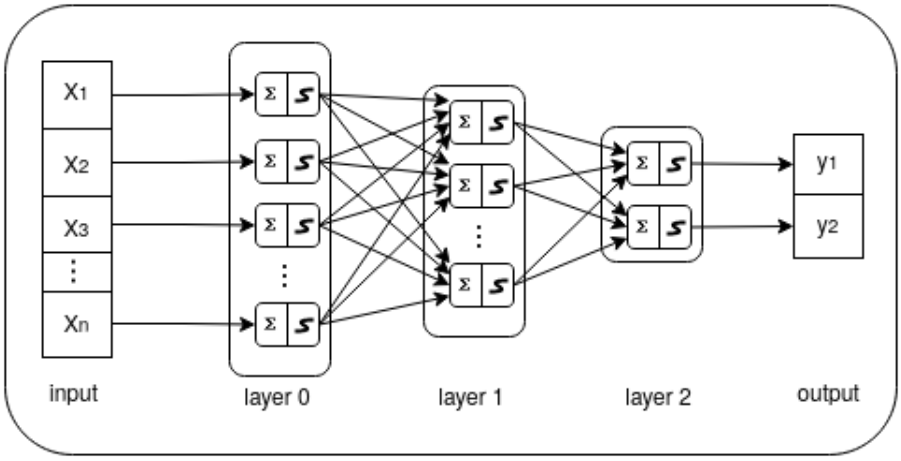


Figura 5 – Exemplo de Rede Neural Simples

treino, e assim melhor generalizar o modelo em relação aos dados. Quanto mais camadas e mais neurônios uma rede tiver, mais profunda será a rede, daí a sentido da expressão Aprendizagem Profunda (do inglês *Deep Learning*)

3 Trabalhos Relacionados

Na literatura encontramos diversos trabalhos que propõem modelos ou *frameworks* de classificação de produtos. Na sua grande maioria, utilizam abordagens genéricas de classificação comprovadamente eficazes em problemas de classificação em geral, como *K-nearest neighbors (KNN)* (Ziegler et al., 2004) e SVM (Chen and Warren, 2013). Temos presenciado nos últimos anos um forte crescimento na adoção de modelos complexos baseados em *Deep Learning* (Skinner, 2018; Tan et al., 2020) e Técnicas de *Ensembling* (combinação de modelos) (Yu et al., 2018; Goumy and Mejri, 2018). Isso se deve, entre outros fatores: a) aos bons resultados que esses modelos apresentam nas mais diversas áreas; b) à grande massa de dados de produtos, com milhares de classes, o que pode desencorajar o uso de alguns modelos clássicos e c) ao alto poder computacional mais acessível capaz de encurtar tempo de treino em bases grandes. Estudos mais intensivos sobre a importância de se automatizar o processo de classificação de produtos e as abordagens utilizadas podem ser encontrados em (Abels and Hahn, 2005; Mattos et al., 2012).

(Wolin, 2002) propôs um método de classificação chamado AutoCat que, diferentemente do nosso cenário, além da descrição recebe outras *features* de entrada como fabricante, modelo e preço do produto. O AutoCat utiliza uma variação do modelo de Espaço Vetorial (Salton and McGill, 1986) para representar cada produto através de suas *features*. A cada *feature* do produto é atribuído um peso junto à respectiva representação vetorial. A combinação de todas as *features* e seus pesos geram um *rank*, e a classe no topo do *rank* é atribuída ao produto. Experimentos revelaram que o uso de preço como *feature* de classificação não causou impacto nos resultados. (Cortez et al., 2011) utilizam AutoCat como *baseline* para comparar com o modelo proposto, TopCat, onde verificaram que seu modelo obteve melhores resultados a partir de métricas de Micro-F1 e Macro-F1. TopCat é um dos trabalhos que utilizamos como *baselines*.

(Bojanowski et al., 2016) apresentam o Modelo Subword, procurando resolver o problema de muitos modelos de representação de palavras ignorarem a análise morfológica das palavras, por representarem cada palavra completa num espaço n-dimensional. Na abordagem dos autores, cada parte (*n-gram*) de palavra é mapeada a um vetor, e assim uma palavra é representada pela soma dos vetores das partes que a formam. Com essa abordagem, até mesmo palavras que não estão contidas na base de treino podem ser recuperadas na fase de teste ou utilização de modelos. Essa abordagem aplica um modelo de vetorização, chamado modelo Skip-gram (Mikolov et al., 2013b), que foi originalmente idealizado para palavras inteiras, mas os autores aplicaram em trechos de palavra (ou *subwords*). Essa abordagem é uma das principais características do modelo FasText (Joulin et al., 2016), que é estudado por nós e por outros autores. Nesta disserta-

ção, estudamos a aplicação do modelo FastText ao problema de classificação de produtos, comparando sua eficácia à de outros métodos desenvolvidos especificamente para executar a tarefa de classificação.

Uma análise semelhante à nossa foi realizada em (Yu et al., 2018), onde os autores verificaram a efetividade do modelo FastText (Joulin et al., 2016) no ambiente de classificação de produtos. (Yu et al., 2018) combinam *machine learning, deep learning*, e processamento de linguagem natural. Utiliza-se o algoritmo *FastText* como *baseline*, buscando aproveitar aspectos contidos nele. O método utilizado consiste em construir vários modelos de classificação, combinando-os para gerar um único modelo. Ou seja, analisa-se vários algoritmos *(FastText, TextCNN, TextRNN, árvore de diretório e modelos AbLSTM)*, e baseado em resultados, escolhe os melhores modelos e os combina para formar um classificador. No trabalho, procura-se resolver, através da utilização de *word vectors*, o problema de, na base de teste, haver muitas palavras que não estão na base de treino. Ou seja, procura-se por similaridade semântica de novas palavras com palavra que já estejam na base.

Apesar de (Yu et al., 2018) também utilizar o modelo FastText na classificação de produtos, o nosso trabalho aplica este modelo de maneira diferente. (Yu et al., 2018) utiliza o modelo para gerar vetores de palavras (*embeddings*) utilizadas como *inputs*, e no processo de classificação combina o modelo FastText com vários outros modelos. Nossa abordagem com o FastText consiste em utilizá-lo isoladamente na classificação de produtos e analisar o impacto que a segmentação de texto aplicado no modelo causa no resultado final.

(Chen et al., 2019) propuseram um método de classificação baseado em aprendizagem profunda, denominado por eles como Modelo Neural de Classificação de Produtos (NPC em inglês). O NPC adota uma camada de *embedding* convolucional em nível de caractere para aprender as representações de composição de palavras, e uma camada residual em espiral para extrair as anotações de contexto de palavras em uma tentativa de modelar dependências complexas de longo alcance e informações estruturais. Seu método classifica um produto, reconhecendo classes através do conteúdo do produto e através de vocabulários predefinidos de classe. Experimentos extensivos realizados em conjuntos de dados reais oriundos de plataforma de *e-commerce* ilustram a eficácia dos modelos propostos.

Além de inferir a classe de um produto a partir de uma taxonomia pré-definida, o método proposto em (Chen et al., 2019) também é capaz de gerar novas classes de produtos, modificando a árvore de taxonomia. Isso nos impossibilita de tomá-lo como *baseline* para comparação com nossos modelos pois trabalhamos com estruturas de classes fixas. Outra abordagem que difere de nosso trabalho é que eles usam dados de seção do usuário como *feature* para classificação com o intuito de ajudar nos casos de classes com

poucas ocorrências, assumindo que o comportamento do usuário provavelmente conecta produtos a classes.

(Tan et al., 2020) propõem um novo paradigma de classificação usando tradução de texto (do inglês *Neural Machine Translation - NMT*) para classificar produtos. Isso é feito traduzindo a descrição, em linguagem natural de um produto, em uma sequência de *tokens* que representam um caminho da raiz à folha em uma taxonomia de produto. A classificação é feita de ponta a ponta na taxonomia, ou seja, para cada nível uma classe desse nível é atribuída ao produto, de forma que a saída do modelo seja um caminho raiz-folha. O trabalho proposto é capaz de inferir novos caminhos a partir da raiz da taxonomia até a folha, modificando a árvore. O uso de *Machine Translation* (MT) traz benefícios como a reutilização de algoritmos de tradução que as lojas de comércio eletrônico (como o Alibaba, por exemplo) já possuem, o desenvolvimento expressivo que MT sofreu nos últimos tempos e o fato de que essa abordagem é capaz de lidar com o ruído da informação na comunicação em linguagem natural. Não usamos este trabalho como nosso *baseline* por conta do método de classificação ponta a ponta, gerando todas as classes de todos os níveis, enquanto nosso modelo infere uma classe de um nível por vez. Outro fator de exclusão foi a possibilidade de modificação da árvore de taxonomia, ou seja, assim como no modelo anterior, esse modelo pode modificar a estrutura das classes (criando ligações entre duas classes de níveis adjacentes, por exemplo).

Em nossos experimentos adotamos dois *baselines* que são soluções específicas para classificação de produtos em ambiente de e-commerce. O primeiro, proposto em (Cortez et al., 2011), chamado de TopCat, usa uma abordagem probabilística para executar a classificação. Esse modelo usa uma rede *bayesiana* onde cada produto a ser classificado é representado por atributos, sejam eles descrição, marca ou preço. Para cada um desses atributos, uma probabilidade é calculada para cada classe; no final cada classe é representada pela combinação de todas as probabilidades atribuídas a ela em todos os atributos.

O trabalho mostra que o particionamento da descrição de um produto em segmentos e a análise de cada segmento como uma *feature* distinta causa grande impacto nos resultados de classificação. Dessa forma, para comparação de *baseline*, aplicamos na implementação do modelo proposto pelos autores (TopCat) a mesma estratégia de segmentação com as mesmas configurações: A descrição de cada produto foi dividida em três segmentos e uma probabilidade foi gerada para cada segmento, além da descrição. Finalmente, as probabilidades geradas por cada segmento e pela descrição completa foram agregadas através de uma combinação linear simples. A classe com maior probabilidade agregada é atribuída ao produto.

A figura 6 ilustra a rede *bayesiana* utilizada pelos autores. Cada nó C_j representa uma classe; o nó PO representa um produto a ser classificado; os nós de K_1 a K_t re-

presentam as palavras que aparecem nas descrições de todos os produtos encontrados na coleção; os nós S_1 a S_q representam as marcas e os nós P_1 a P_l representam os preços modelados em valores discretos. Cada nó CD_j representa a probabilidade de o produto representado por PO pertencer à classe j a partir das palavras contidas na descrição; similarmente cada nó CS_j e P_j representa a probabilidade do produto pertencer à classe j tomando em consideração a marca e o preço respectivamente. Finalmente, cada nó C_j representa a probabilidade condicional da classe j, dada pela equação 3.1.

Para efeito de comparação equitativa entre os modelos utilizamos somente as descrições de produtos como atributo de entrada do modelo. Conforme (Cortez et al., 2011) a função de probabilidade para a classe j baseada na descrição do produto é dada pela equação 3.2. A função $f(k, C_j)$ retorna a frequência global to termo k em todas as descrições de produtos pertencentes à classe j; a função $terms(\mathbf{k})$ retorna o conjunto de termos na descrição do produto; η é uma constante de normalização, dada por $1/|PO|$, onde $|PO|$ é o número de palavras na descrição do produto (Pearl, 1988); C_j é a classe representada pelo nó CD_j.

$$\begin{aligned} P(c_j|PO) = 1 &- (1 - P(cd_j|PO)) \\ &\times (1 - P(cp_j|PO)) \\ &\times (1 - P(cs_j|PO)) \end{aligned} \tag{3.1}$$

$$P(cd_j|\mathbf{k}) = \eta \sum_{k \in terms(\mathbf{k})} \frac{f(k, C_j)}{\sum_{c' \in C} f(k, c')} \tag{3.2}$$

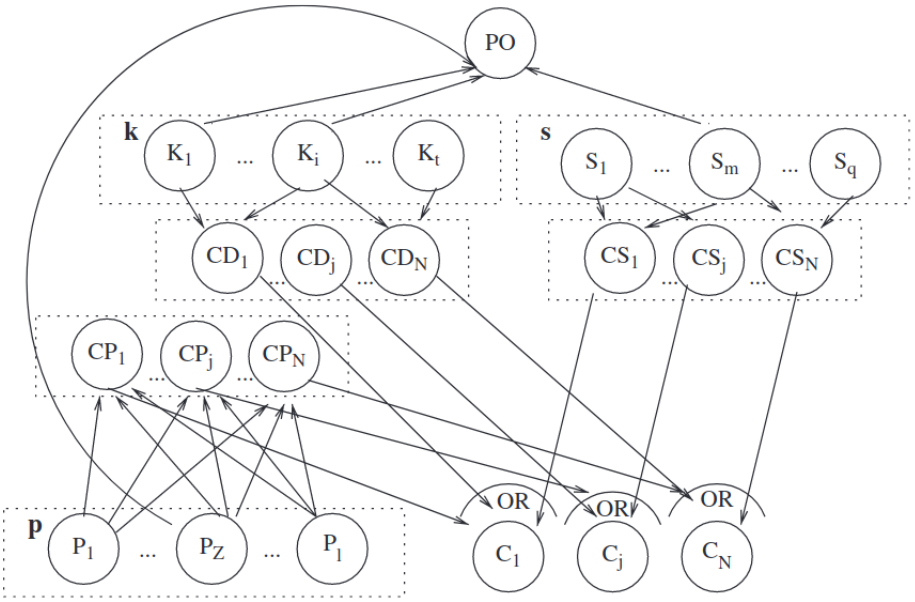


Figura 6 – Rede bayesiana para combinação de probabilidades no modelo TopCat

Incluímos o TopCat em nosso estudo de métodos de classificação por apresentar um modelo probabilístico de baixo custo, assim como o nosso. Além disso, utilizamos e experimentamos ideias de segmentação de descrições propostas pelos autores, combinando-as ao uso tanto de Language Models quanto do modelo FastText.

O segundo modelo usado como *baseline* (Skinner, 2018) foi escolhido por ter vencido o Rakuten Data Challenge (SIGIR 2018 ECOM), uma competição internacional proposta pela empresa Rakuten para o desenvolvimento de soluções de classificação automática de produtos. Denominaremos o modelo aqui como RakModel, dado que os autores não sugeriram nenhum nome para o método proposto. Trata-se de um modelo baseado em aprendizagem profunda que utiliza redes neurais com camadas LSTM (Long Short-term Memory), *Balanced Pooling View* e técnicas de *ensembling* (combinação de modelos) na tarefa de classificação de produtos. Nos últimos anos Redes LSTM têm se mostrado bastante promissoras quando se trata de problemas que envolvem dados sequenciais, como tradução de texto, extração de tópicos, entre outros problemas na área de Processamento em Linguagem Natural (PLN). Balanced Pooling View consiste em estreitamento entre conexões de camadas recorrentes e camadas de saída através da utilização de camadas de *pooling*. Com técnicas de *ensembling* é possível agregar diferentes modelos em um único modelo na tentativa de reter as melhores características de cada modelo agregado.

Para obter os benefícios de técnicas de *Ensembling* o algoritmo reverte a sequência da entrada, simulando um aumento de dados, e treina um segundo modelo independente com os dados reversos. Nesse trabalho foram usadas três redes do tipo *forward* (com os caracteres de entrada em suas posições reais) e três redes do tipo *backward* (com os caracteres de entrada na posição oposta ao original). Fazendo isso o F-score sofreu um aumento de quase 0.01 no conjunto de dados de validação provido na competição. Os autores usaram *tokenização* de caracteres por conta de grande incidência de símbolos e códigos alfanuméricos. Por exemplo, em muitas descrições há "palavras"com números e letras que representam séries e modelos de produtos, o que de fato não são palavras no contexto gramatical. Por isso a *tokenização* a nível de caractere se torna mais eficiente do que a nível de palavra nesse contexto. Além disso, para resolver problemas de esparsividade, caracteres que ocorriam menos de 10 vezes eram substituídos por *"unknown"token*. No final o vocabulário continha 128 *tokens*. 25% dos dados de treino foram separados para validação com *Stratified Sampling* para manter uma boa representação de dados. *Dropout* (desabilitação de algumas células em uma camada) foi usado depois de camadas de *embeddings*, entre camadas LSTM e depois da saída para regularizar o modelo. A figura 7 mostra a arquitetura utilizada por (Skinner, 2018) para modelar e tratar o problema.

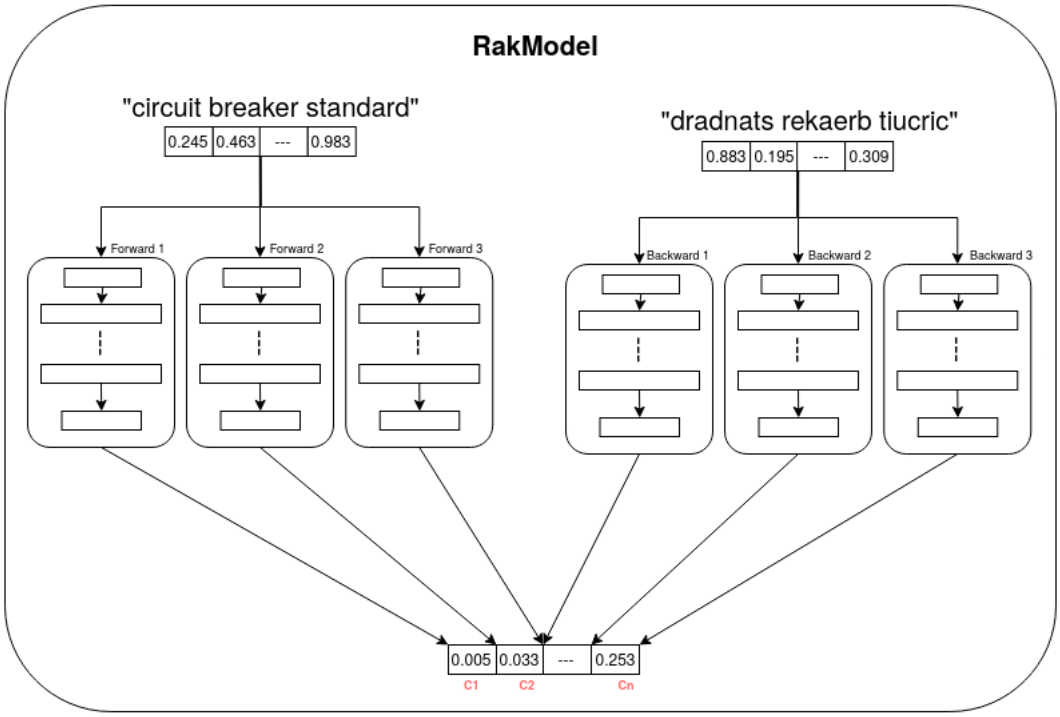


Figura 7 – Arquitetura do Modelo RakModel

4 Modelos de Classificação Estudados

Neste capítulo apresentamos as soluções propostas e estudadas por nós para a classificação de produtos.

4.1 Classificação de Produtos Usando Language Models

Em uma das abordagens estudadas nesta dissertação, estudamos o uso de Language Models (LM) para classificação de produtos e produzimos um método que denominamos de LangCat. Sabemos que um produto pode ser representado por um texto, por meio da descrição ou de comentários referentes ao produto; com isso podemos aproveitar ideias de modelagem usadas em sistemas de busca textual usando LM para aplicá-las ao problema de classificação de produtos. A aplicação de LM tem por objetivo descobrir qual a probabilidade da descrição textual de um dado produto ser gerada por uma fonte de dados que modela uma dada classe.

Nesse contexto, a equação 4.1 representa a fórmula geral do modelo de linguagem, gerado com a probabilidade condicional da descrição d de um produto ser gerada a partir de uma classe c. Essa probabilidade é alcançada a partir da conjunção das probabilidades de cada termo $w_i \in d$ pertencer a c. Aqui, c é representado pela concatenação das descrições de todos os produtos que pertencem a c.

$$p(d|c) = \prod_{i=1}^{n} p(w_i, c) \tag{4.1}$$

Para que se tenha uma probabilidade não nula, mesmo para palavras que não ocorrem na fonte de dados que representa a classe, utilizamos a suavização proposta em (Zhai and Lafferty, 2004), que visa ajustar o estimador de máxima verossimilhança de um ML para que ele seja bem acurado. O princípio da máxima verossimilhança é um procedimento que visa determinar qual modelo, dentre todos aqueles definidos pelos possíveis valores de seus parâmetros, possui maior possibilidade de ter gerado o texto de entrada. Ou seja, quanto maior a verossimilhança um ML mais representativo estará em relação ao texto de entrada. A equação 4.2 representa a fórmula otimizada de ML aplicando a suavização de Zhai e Lafferty no ambiente de Classificação de Produtos.

$$\log p(d|c) = \sum_{i:f(w_i;c)>0} \log \frac{p_s(w_i|c)}{\alpha_c \; p(w_i|C)} + n. \log \; \alpha_c + \sum_{i=1}^{n} \log p(w_i|C). \tag{4.2}$$

$$\text{Sendo } p_s(w|c) = \frac{f(w,c) + \mu p(w|C)}{|c| + \mu},\ \alpha_c = \frac{\mu}{\mu + |c|}$$

Como visto em (Zhai and Lafferty, 2004), a fórmula de suavização aplicada às classes, $\alpha_d = \frac{\mu}{\mu+|c|}$, faz uso da variável μ como parâmetro para ajustar a densidade das palavras com base em sua ocorrência em todo o banco de dados. Durante os experimentos do modelo Langcat, variamos o parâmetro μ entre os valores de 100 e 5000 e obtivemos o valor de $\mu = 500$ como aquele com o melhor resultado.

A figura 8 mostra a dinâmica do modelo proposto. Nesse modelo, parte dos dados serão usados para popular as estruturas das classes (podemos chamar essa fase de treino), e a outra parte para gerar os modelos de cada classe, partindo das informações previamente adquiridas dos dados de treino. Na primeira fase, cada classe possui uma estrutura para guardar as ocorrências de termos contidos em produtos que pertencem à classe. Para isso utilizamos listas invertidas pois, como a base de dados não sofre atualização após sua criação, ela será utilizada apenas para consultas de ocorrências de termos em classes. Listas separadas são atribuídas aos segmentos, cada segmento tornando-se uma *feature* de entrada de dados independente da descrição completa.

Após a entrada de uma consulta, para cada classe o algoritmo gera um modelo de linguagem, buscando verificar qual é a probabilidade dos termos da descrição do produto ser gerada a partir dos termos de cada classe. Aplica-se nessa fase o tratamento de suavização de Zhai e Lafferty para amenizar as diferenças entre termos muito abundantes e termos bem escassos na base. Após essa fase, para cada classe é gerado um modelo com a sua respectiva probabilidade. A classe que contiver a maior probabilidade será a classe atribuída ao produto. Os resultados que alcançamos são impressionantes, com o método alcançando uma qualidade de classificação pareado com os modelos mais sofisticados, sendo ao mesmo tempo simples de implementar e rápido.

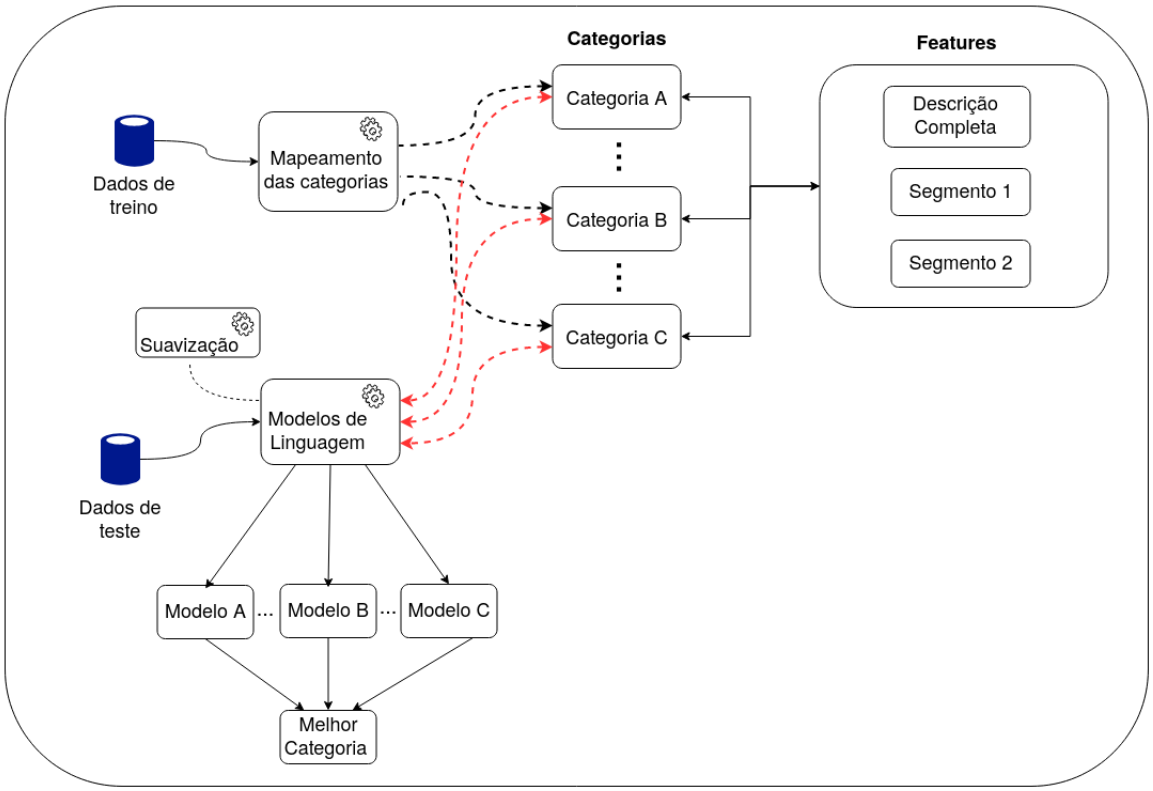


Figura 8 – Arquitetura do Classificador de Produtos utilizando Language Models - LangCat

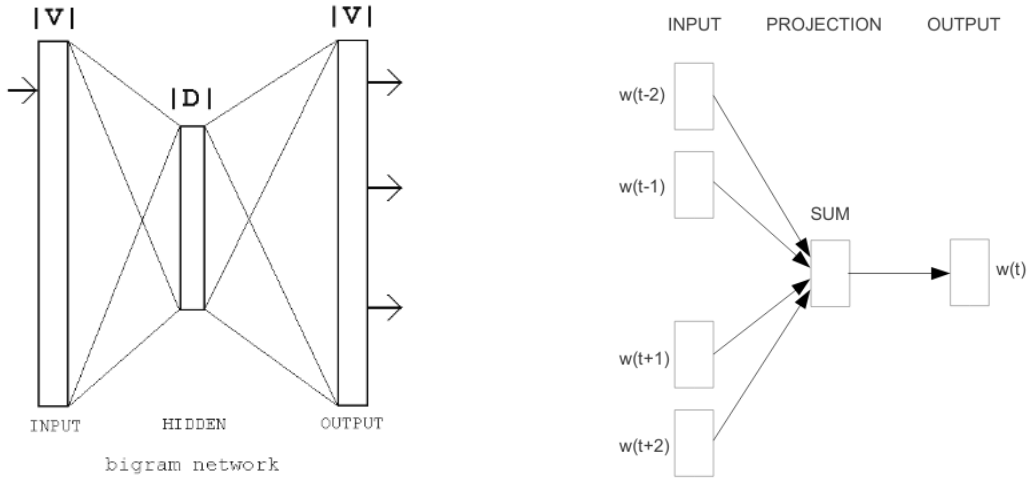
4.2 FastText

Proposto por (Joulin et al., 2016), trata-se de um algoritmo de classificação de textos em geral. Os autores aplicam o modelo ***CBOW - Continuous Bag-of-Words Model*** (Mikolov et al., 2013a) juntamente com a abordagem de representação de palavras através de trechos de caracteres *(n-gramas)*, chamada ***Subword Information*** (Bojanowski et al., 2016).

O modelo CBOW utiliza uma arquitetura de treinamento em duas etapas: Primeiramente uma rede neural bigrama é treinada para gerar representação vetorial de palavras em espaço n-dimensional (vide imagem 9a). Essa rede é composta por três camadas: a primeira camada sendo de *input* de palavra codificada em *one-hot vector* (um vetor de tamanho |V|, sendo V o total de palavras no vocabulário, onde todas as posições possuem o valor 0, com exceção da posição que representa a palavra, que possui valor 1), a segunda sendo uma camada escondida de tamanho variado (entre 20 e 60 neurônios) e a camada de *output* que gera a distribuição de probabilidade. O treino consiste em, dada uma palavra w do vocabulário V, estima-se a distribuição de probabilidade da próxima palavra num texto de entrada. A intenção final desse treinamento é otimizar a representação de palavras (também conhecida como *embedding*) pelos vetores gerados na camada intermediária, de modo que palavras com probabilidades semelhantes estejam 'perto' no espaço

multi-dimensional, demonstrando que são palavras com algum significado próximo.

Depois do treinamento de *embeddings*, outra rede neural é configurada e treinada com a tarefa de inferir uma palavra t dando como entrada palavras que estão no contexto (n palavras antes e n palavras após a ocorrência de t). Essa rede também possui 3 camadas: *input*, projeção e *output*. Na cama de *input*, os vetores das palavras do contexto são inseridas (esses vetores são extraídos a partir da camada intermediária da rede representada pela figura 9a). A camada intermediária da segunda rede consiste numa camada de projeção compartilhada por todas as palavras, que contém a média dos vetores. A terceira camada representa a palavra inferida pelo modelo. A figura 9b mostra a arquitetura da segunda rede neural.



(a) Rede bigrama utilizada para gerar representação vetorial de palavras

(b) Rede neural utilizada para inferência

Figura 9 – Redes neurais utilizadas no modelo CBOW

A abordagem de **Subwork Information** consite em representar cada palavra da coleção por uma ***bag of n-gramas*** extraída da palavra original. Cada *n-grama* é representado por um vetor e cada palavra é representada pela soma de todos os vetores dos *n-gramas* que são encontrados nela. Algumas vantagens incluem a celeridade no treinamento de modelos a partir de bases muito grandes (contornando um ponto negativo em abordagens com *deep-learning*) e a possibilidade de encontrar a representação vetorial de palavras que não constam nos dados de treino, apenas usando seus *n-gramas*, desde que tais *n-gramas* sejam encontrados em outras palavras na base de treino.

O modelo leva em consideração características morfológicas para verificar a similaridade entre duas palavras diferentes. Essa abordagem resolve algumas limitações de tratar somente palavras completas e ignorar palavras com diferenças tênues em relação às palavras contidas na base. Por exemplo: considerando que temos a palavra *refrigerador* em nossa base; se quisermos analisar a palavra *refrigeração* oriunda de uma consulta. Usando *Subword Information*, torna-se possível verificar que *refrigerador* e *refrigeração*

possuem alguma similaridade a partir de seus n-gramas. Com isso a palavra *refrigeração* não é sumariamente desprezada, e sim vista como tendo certa similaridade com a palavra *refrigerador*.

FastText tem se mostrado promissor na tarefa de classificação em várias áreas, inclusive em classificação de Produtos (Yu et al., 2018). Utilizamos as descrições como *input* no processo de treino e inferência na execução do FastText. Avaliamos o impacto que a técnica de segmentação utilizada junto com o FastText causa no resultado final. Fizemos experimentos, alternando parâmetros e analisando os resultados obtidos e assim ajustamos a configuração dos modelos na execução de treino e teste.

4.3 Segmentação de texto

A segmentação de texto consiste numa busca por aumentar a quantidade de *features* relevantes para o processo de classificação. O texto é particionado (não necessariamente em partes iguais) com o intuito de analisar cada segmento (ou partição) de maneira independente, buscando obter ganhos com segmentos que contenham palavras que caracterizam fortemente a classe à qual pertencem. Isso se dá porque, ainda que os termos do segmento estejam na descrição completa, quando transformado em uma *feature* distinta e analisado isoladamente, esse segmento adquire uma probabilidade maior quando seus termos forem mais relevantes para a categoria, pois outros termos menos relevantes não exercem influência para a probabilidade do segmento. (Cortez et al., 2011) apresenta um estudo sobre o impacto da segmentação da descrição de produtos nos resultados de classificação. Eles mostram que segmentos pequenos, usualmente no começo ou no final da descrição podem ser usados para melhor descrever as classes de produtos. Verificou-se que a posição de ocorrências de termos na descrição pode caracterizar um importante papel no processo de classificação. Por exemplo, muitos produtos da classe CD contêm a palavra *CD* no início da descrição. A tabela 1 evidencia a importância das primeiras palavras das descrições dos produtos como caracterizadoras de classes. Nela podemos ver o número distinto de palavras para cada posição da descrição (até a terceira posição) e a média de produtos onde essas palavras ocorrem. Por conta do aspecto *longtail* dos dados, sabemos que as médias não representam a realidade, pois há palavras com grande incidência e outras com pouca incidência nas descrições dos produtos.

Baseando-se nessa concepção, os autores sugerem uma estratégia de dividir a descrição em N segmentos, sendo os primeiros $N-1$ segmentos compostos por sequências com número de palavras fixas e o último com o remanescente da descrição. Na abordagem dos autores, aplicada no modelo TopCat, a probabilidade de cada segmento é computada da mesma forma como é feita com a descrição, e finalmente combina as probabilidades de todos os segmentos, juntamente com as probabilidades de todos os demais atribu-

tos. Seus resultados experimentais indicam que o uso dessa abordagem pode aumentar consideravelmente a qualidade da classificação.

Usamos a abordagem de segmentação de texto para otimizar nosso modelo baseado em *Language Models* (LangCat). Realizamos vários testes variando o número de segmentos (entre 1 e 6), assim como o número de palavras (entre 1 e 3) nos primeiros segmentos, para verificar a melhor configuração. Vimos que usar 2 segmentos, com apenas uma palavra cada segmento, foi a melhor configuração alcançada. Isso se deve à grande carga de significados que geralmente as primeiras palavras possuem na descrição dos produtos, como "fogão", "geladeira", etc. Esse fato ratifica a tese proposta pelos autores que propuseram a segmentação, de que segmentos pequenos, geralmente no início ou no final, descrevem melhor as classes de produtos. Vale ressaltar que a base de dados do Rakuten possui textos em Inglês. Enquanto na língua portuguesa a descrição de itens, em geral, começa com algum termo que dá forte indício da classe à qual o produto pertence, na língua inglesa isso é menos percebido. Todavia, esse aspecto não influenciou na performance do uso de segmentação na base do Rakuten.

Além da execução do modelo padrão do FastText, fizemos experimentos com a conjunção do FastText e o método de segmentação das descrições, afim de verificar se há ganhos no resultado final. A estratégia utilizada foi treinar modelos separadamente (um treinamento para cada segmento e um para a descrição) e depois, para cada produto na base de teste, inferir as probabilidades para cada classe presente na taxonomia. Utilizamos 3 segmentos, sendo os dois primeiros contendo apenas uma palavra cada e o último contendo todo o restante da descrição. No final somamos as probabilidades atribuídas por classe, e a classe com maior probabilidade se torna a classe inferida.

	Palavras distintas por posição					
	Walmart		**Americanas**		**Rakuten**	
	Total de produtos	34524	**Total de produtos**	660902	**Total de produtos**	789036
Posição	**Palavras distintas**	**Média**	**Palavras distintas**	**Média**	**Palavras distintas**	**Média**
1ª	2767	12.48	30323	21.80	65398	12.07
2ª	4052	8.52	39836	16.59	128898	6.12
3ª	5947	5.81	61737	10.71	116014	6.80

Tabela 1 – Detalhamento das variações de palavras de acordo com a posição na descrição.

5 Experimentos

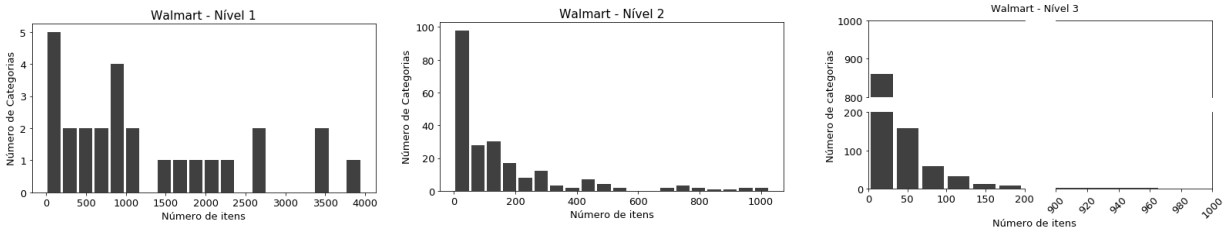
5.1 Base de dados

Para realizar as avaliações fazemos uso de duas bases de dados extraídas de *websites* das lojas *Walmart* e *Americanas*, ambas são lojas de comércio eletrônico brasileiras, e uma base de dados usada no SIGIR Data Challeng 2018 (Rakuten). A coleção de *Walmart* possui 34.524 produtos, a coleção de *Americanas* possui 660.902 produtos e a base do Rakuten possui 789.036 produtos, ambas são distribuídas em três níveis de classes. A tabela 2 mostra a quantidade de classes por base de dados distribuídas por níveis da árvore de taxonomia, a média de itens por classe e a mediana das quantidades de itens nas classes. Quanto mais perto a mediana estiver da média, mais balanceada será a distribuição de produtos entre as classes. Se a mediana estiver muito abaixo da média, significa que há muitas classes com escassez de itens enquanto algumas poucas classes possuem muitos itens.

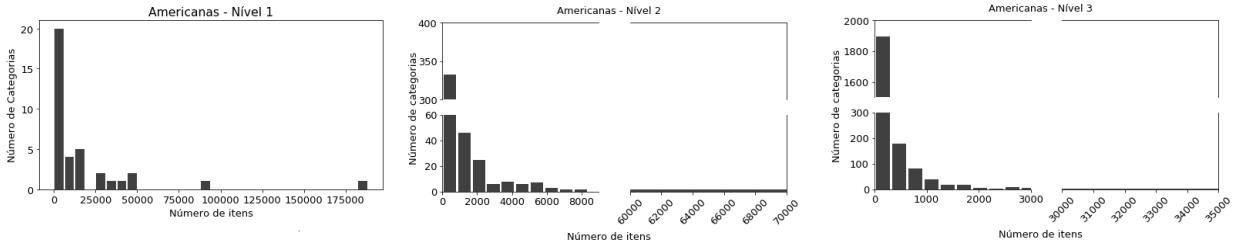
A figura 10 apresenta histogramas que representam a distribuição das bases. Podemos notar que em todas as bases o primeiro nível contém uma distribuição relativamente balanceada. Isso se deve às poucas quantidades de classes nesse nível. Para os níveis 2 e 3 de todas as bases vemos histogramas no formato de *longtail* com queda acentuada no início do gráfico. Nesses níveis há um desbalanceamento na distribuição de produtos por classe, ou seja, poucas classes possuem muitos itens e muitas classes possuem poucos itens.

	Distribuição das Categorias								
	Walmart			**Americanas**			**Rakuten**		
Nível	**Qtd de categorias**	**Média Itens**	**Mediana**	**Qtd de categorias**	**Média Itens**	**Mediana**	**Qtd de categorias**	**Média Itens**	**Mediana**
1	27	1278.67	917	37	17862.21	4160	13	60695.07	23529
2	224	154.12	79	453	1458.94	332	101	7812.23	1653
3	1155	29.89	11	2286	289.1	58	865	912.18	60
Total	1406			2776			979		

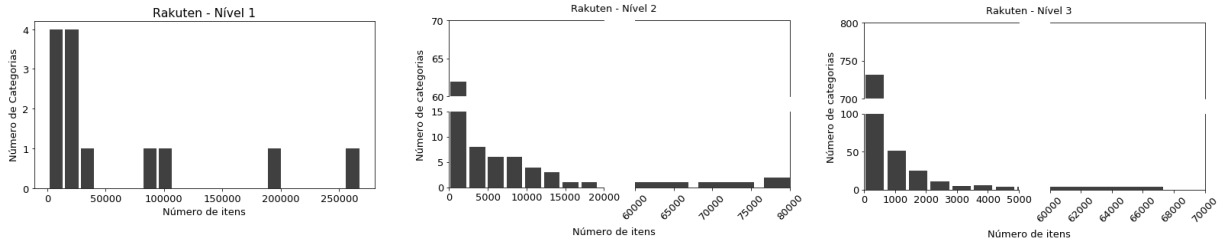
Tabela 2 – Detalhamento das distribuições de categorias entre os níveis.



(a) Distribuição da Base Walmart



(b) Distribuição da Base Americanas



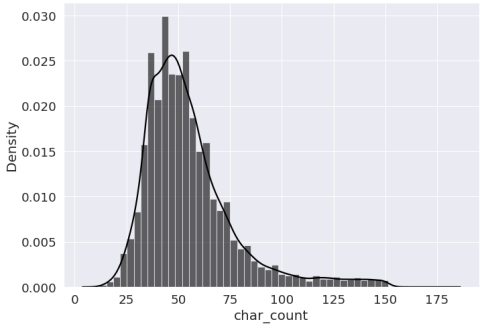
(c) Distribuição da Base Rakuten

Figura 10 – Distribuição das Bases de Dados.

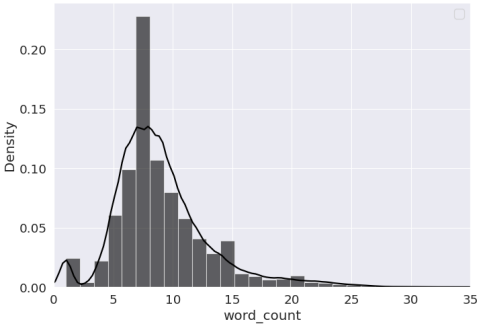
A figura 11 mostra os histogramas da densidade de número de caracteres e palavras contidas nas descrições dos produtos. As imagens da coluna à esquerda representam a distribuição do número de caracteres, enquanto que as imagens da coluna à direita representam a distribuição do número de palavras. Podemos notar uma certa semelhança na distribuição de caracteres entre as bases de Walmart e Americanas, com grande massa de densidade entre 20 e 150 caracteres e pico perto de 50 caracteres. Comportamento similar acontece também com a distribuição de número de palavras nessas duas bases: grande massa entre 3 e 25 palavras e pico em 7 palavras. A base de dados Rakuten se distingue levemente das outras duas, apresentando mais quantidades de caracteres e de palavras. Muito provavelmente isso acontece pelo fato de ser uma base com títulos de produtos em inglês, enquanto que as outras duas possuem títulos de produtos em português. Esse fator pode favorecer o modelo Rakuten, pois ele utiliza Redes Neurais LSTM (*Long short-term memory*) que usa células com memória capazes de lidar com relações semânticas entre palavras que não estejam "perto" uma da outra na sequência de palavras que compõem a descrição do produto.

Outro ponto interessante a se observar é que, nas plotagens de distribuições de número de palavras, as linhas de KDE (*Kernel Density Estimation*) sofrem bastante oscilações. Com isso podemos perceber que a distribuição não é contínua, ou seja, o padrão de

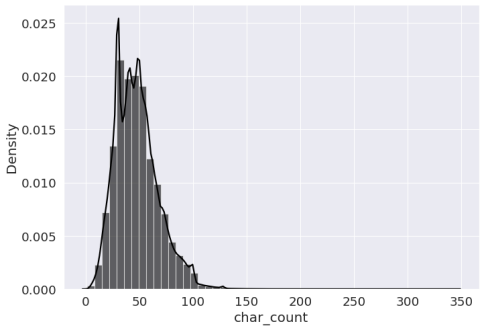
comportamento é mais desorganizado. Esse fenômeno é mais nítido na base de Americanas e Rakuten, muito provavelmente pelo fato de essas bases conterem centenas de milhares de produtos, enquanto a base de Walmart contém dezenas de milhares de produtos.



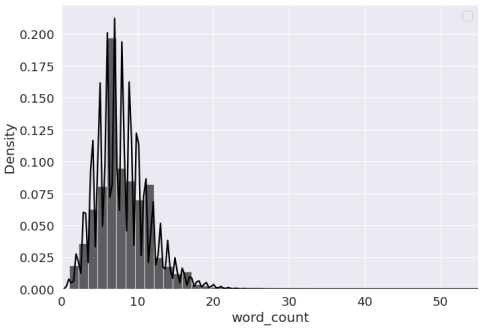
(a) Dist. Caracteres Walmart



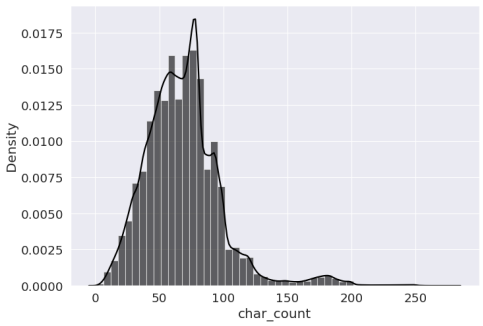
(b) Dist. Palavras Walmart



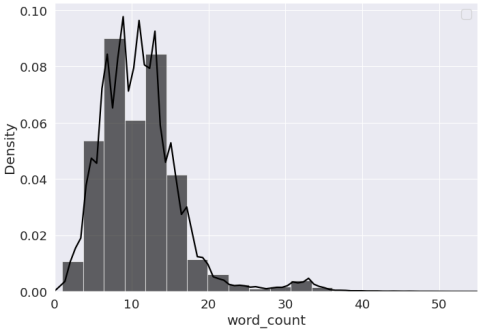
(c) Dist. Caracteres Americanas



(d) Dist. Palavras Americanas



(e) Dist. Caracteres Rakuten



(f) Dist. Palavras Rakuten

Figura 11 – Distribuição de densidade de número de caracteres e palavras nos títulos dos produtos.

5.2 Métricas para Avaliação

Para avaliar cada resultado, usamos as métricas de classificação tradicionais ***micro-averaged F1 (micF1)*** e ***macro-averaged F1 (macF1)*** (Baeza-Yates and Ribeiro-Neto, 1999). Estas são diferentes funções de médias baseadas na media *F1-score*, que combina valores de *precisão* e *revocação* de uma categoria para avaliar a performance de classificação.

A precisão *P(c)* de uma classe c é a porcentagem de produtos corretamente classificados como sendo da classe c entre todos os produtos assimilados a essa classe. A revocação *R(c)* de cada classe c é definida como a porcentagem de produtos corretamente classificados como pertencentes à classe c entre todos os produtos que realmente pertencem a c. Sendo assim, a *F1-score* de uma classe c é calculada como segue:

$$F_1(c) = \frac{2P(c)R(c)}{P(c) + R(c)} \tag{5.1}$$

Dado um conjunto de classes $/C/$ usado para classificação, a *macro-média F1* calcula a média de *F1-scores* de todas as classes:

$$macF_1 = \frac{\sum_{c=1}^{|C|} F_1(c)}{|C|} \tag{5.2}$$

A *micro-média F1* considera a soma de precisões P e a soma de revocações R de todas as classes e calcula a média como segue:

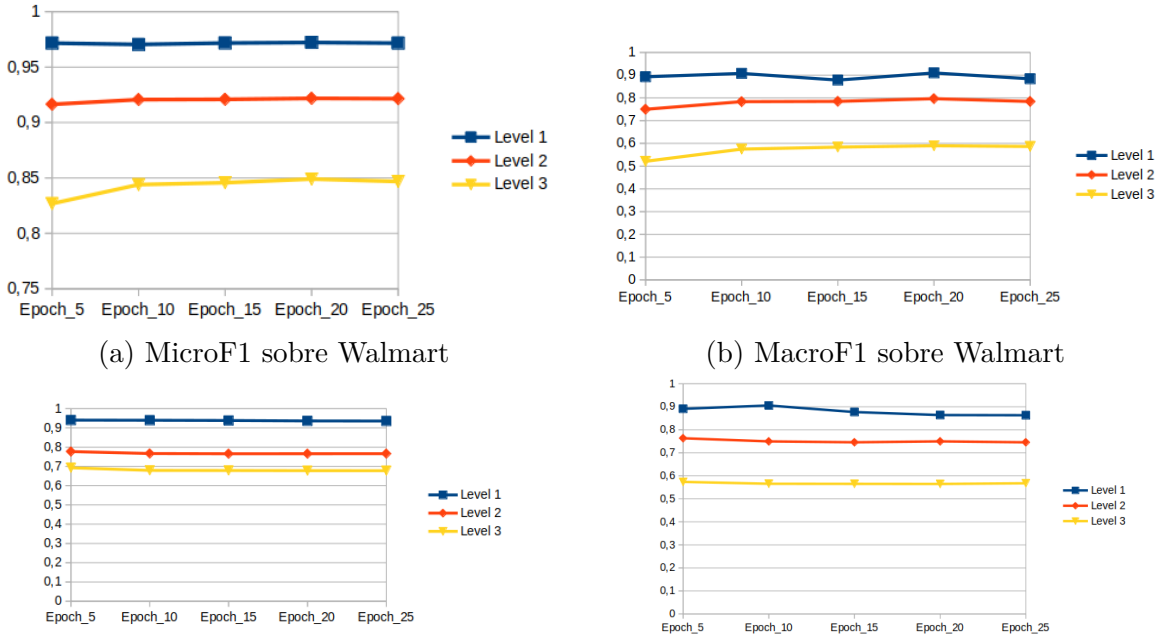
$$micF_1 = \frac{2PR}{P + R} \tag{5.3}$$

5.3 Parametrização do FastText

A tabela 3 mostra uma análise comparativa entre o modelo FastText com e sem a segmentação, aplicando ambas as abordagens nas bases de dados de Rakuten, Walmart e Americanas. Utilizamos 3 segmentos, sendo os dois primeiros com apenas uma palavra e o terceiro com o restante da descrição. Podemos constatar que a implementação de segmentação de texto junto com o FastText proporciona um ganho considerável em todos os casos analisados. Por conta disso, utilizamos o modelo FastText com segmentação (FasText-SEG) para comparar com os *baselines*. Também fizemos experimentos a fim de verificar a melhor configuração de épocas. A figura 12 descreve os experimentos feitos no FastText comparando MicroF1 e MacroF1. Esses experimentos foram feitos sobre as bases das lojas de *ecommerce* Walmart and Americanas.

Base	Base de Dados: Rakuten				Base de Dados: Walmart				Base de Dados: Americanas			
	FastText		FastText-SEG		FastText		FastText-SEG		FastText		FastText-SEG	
Nível	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1
1	0.9008	0.7879	**0.9107**	**0.8122**	0.9637	0.9146	**0.9759**	**0.9347**	0.9342	0.8864	**0.9427**	**0.8970**
2	0.8410	0.6292	**0.8511**	**0.6465**	0.9272	0.8450	**0.9295**	**0.8558**	0.7661	0.7464	**0.7720**	**0.7543**
3	0.7944	0.3774	**0.8032**	**0.3836**	0.8530	0.6564	**0.8592**	**0.6863**	0.6777	0.5707	**0.6821**	**0.5764**

Tabela 3 – Comparação entre modelos FastText Padrão e FastText com Segmentação.



(a) MicroF1 sobre Walmart

(b) MacroF1 sobre Walmart

(c) MicroF1 sobre Americanas

(d) MacroF1 sobre Americanas

Figura 12 – MicroF1 e MacroF1

Baseado nos resultados vistos na figura 12 usamos a configuração *epoch = 20* no modelo FastText para comparar com os *baselines*, pois essa configuração obteve resultados bem interessantes de acurácia nos dados analisados.

5.4 Parametrização do RakModel

Na implementação do modelo RakModel, seguindo as configurações do modelo cedido pelos autores, 3 pares de modelos (*forward* e *backward*) foram treinadas para cada nível da taxonomia de produtos. A diferença entre eles está na taxa de aprendizagem (*learning rate* - lr) e a quantidade de épocas. O primeiro par de modelos foi treinado com 40 épocas e lr= 0.8, o segundo com 20 épocas e lr=1.2, e o terceiro com 30 épocas e lr=1.0.

5.5 Tabelas comparativas

As tabelas 4, 5 e 6 mostram os resultados dos experimentos realizados nas bases de dados Walmart, Americanas e Rakuten. Para cada modelo, pares de valores microF1 (MicF1) e macroF1 (MacF1) são organizados em cada nível da taxonomia. Assim, podemos fazer uma análise comparativa por nível e por tipo de métrica (MicF1 e MacF1), onde o melhor resultado é mostrado em negrito. Utilizamos validação cruzada com 5 folds em cada experimento e tiramos a média dos valores de MicF1 e MacF1.

Os modelos LangCat, TopCat e FastText-SEG foram treinados em um computador com Sistema Operacional Linux/Ubuntu, processador Intel Core i7 com 8 cores, 16G de memória RAM. Para o modelo RakModel, por necessitar de processamento em GPU, utilizamos o ambiente do Google Colaboratory com GPU integrada, Sistema Operacional Linux/Ubuntu, processador Intel Xeon com 2 cores, 16G de memória Placa Gráfica NVidia 64 bits, 33MHz. A tabela 7 mostra o tempo de processamento da fase de treino dos modelos. Utilizamos a notação com prefixos de horas(h), minutos(m) e segundos(s) por conta da alta diferença existente entre tempo de treinamento de modelos distintos.

	Base de Dados: Walmart							
	Langcat		TopCat		FastText-SEG		RakModel	
Nível	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1
1	0.9609	0.8813	0.9298	0.8873	**0.9759**	**0.9347**	0.9658	0.9072
2	0.9091	0.8444	0.8533	0.7866	**0.9295**	**0.8558**	0.9207	0.8380
3	0.8383	**0.6901**	0.7573	0.6071	**0.8592**	0.6863	0.8550	0.6892

Tabela 4 – Comparação dos modelos sobre a base de dados do Wamart.

	Base de Dados: Americanas							
	Langcat		TopCat		FastText-SEG		RakModel	
Nível	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1
1	0.8992	0.8090	0.9071	0.8255	0.9427	**0.8970**	**0.9475**	0.8950
2	0.7565	0.6669	0.7209	0.6190	0.7720	**0.7543**	**0.7831**	0.7503
3	0.6641	0.5314	0.6110	0.4214	0.6821	**0.5764**	**0.6949**	0.5617

Tabela 5 – Comparação dos modelos sobre a Base de Dados de Americanas.

	Base de Dados: Rakuten							
	Langcat		TopCat		FastText-SEG		RakModel	
Nível	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1	MicF1	MacF1
1	0.8578	0.7217	0.8420	0.6567	0.9107	0.8122	**0.9231**	**0.8369**
2	0.7754	0.5482	0.7619	0.4996	0.8511	0.6465	**0.8713**	**0.6609**
3	0.7166	0.3065	0.6928	0.2543	0.8032	0.3836	**0.8260**	**0.4198**

Tabela 6 – Comparação dos modelos sobre a base de Dados do Rakuten.

	Tempo de Processamento			
	Langcat	TopCat	FastText-SEG	RakModel
Walmart	2s	8s	2m 24s	2h 8m 56s
Americanas	28s	20s	1h 47m 8s	25h 21m 44s
Rakuten	40s	26s	49m 50s	48h 8m 4s

Tabela 7 – Tempo de treinamento dos modelos sobre as bases de dados.

Podemos notar que o modelo Langcat, apesar de ser uma abordagem mais simples, compete entre os melhores aplicados à base de dados Walmart, inclusive saindo na frente em um dos casos. Para a base de dados Americanas, o modelo FastText-SEG se sobressai na métrica MacF1 enquanto o modelo RakModel obtém vantagem na métrica MicF1. Já na base Rakuten, o modelo RakModel vence em todos os casos. Vale ressaltar que o RakModel foi feito especificamente para a base de dados da competição em que os autores estavam participando, que era exatamente a base do Rakuten. A calibração do modelo à base pode justificar o absolutismo na comparação. É interessante notar também que a base de dados Americanas é quase vinte vezes maior do que a base de dados do Walmart e possui quase o dobro das classes do Walmart em sua taxonomia. Isso nos dá uma ideia de que Langcat é mais eficiente em bases de dados menos complexos ou diversificados. Através da tabela 7 vemos que os modelos propostos (LangCat e FastText-SEG) exigem tempo de processamento razoável, enquanto o modelo RakModel exige dias de processamento em alguns casos.

6 Conclusões

Podemos ver que o modelo Langcat está entre os melhores quando aplicado a bases de dados menos volumosos, embora não apareça em primeiro lugar em experimentos com bases de dados massivos, apesar de obter bons resultados. Isso concorda com (Cevahir and Murakami, 2016) em que dados com escalas e dimensões grandes são melhor modelados com abordagens mais complexas, como redes neurais profundas (*Deep Learning*), enquanto para bases menores é mais viável usar modelos de aprendizado superficial (como o LangCat). Isso porque esses tipos de modelos não precisam de grandes estruturas computacionais como GPUs e geralmente convergem para uma solução plausível mais rapidamente com bases de dados pequenas. Também constatamos que a aplicação de Segmentação no modelo FastText melhora significativamente o resultado, fazendo-o despontar como melhor entre os modelos analisados para alguns casos. Uma grande vantagem em utilizar os modelos Langcat e FastText-SEG em classificação de produtos, além de obter resultados competitivos, é a celeridade e simplicidade no uso, pois ambos não precisam de infra-estrutura dispendiosa para sua aplicação.

Referências

Abels, S. and A. Hahn

2005. Automatic classification and re-classification of product data in e-business. volume 2005, Pp. 350–353. cited By 1. Citado na página 25.

Agatonovic-Kustrin, S. and R. Beresford

2000. Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 22(5):717–727. Citado na página 23.

Baeza-Yates, R. and B. Ribeiro-Neto

2011. *Modern Information Retrieval*. Bookman. Citado na página 15.

Baeza-Yates, R. A. and B. A. Ribeiro-Neto

1999. *Modern Information Retrieval*. ACM Press / Addison-Wesley. Citado na página 40.

Beneventano, D. and S. Magnani

2004. A framework for the classification and the reclassification of electronic catalogs. volume 1, Pp. 784–788. cited By 12. Citado na página 19.

Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov

2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*. Citado 3 vezes nas páginas 15, 25 e 33.

Cevahir, A. and K. Murakami

2016. Large-scale multi-class and hierarchical product categorization for an E-commerce giant. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Pp. 525–535, Osaka, Japan. The COLING 2016 Organizing Committee. Citado na página 45.

Chen, H., J. Zhao, and D. Yin

2019. Fine-grained product categorization in e-commerce. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, Pp. 2349–2352. Citado 2 vezes nas páginas 15 e 26.

Chen, J. and D. Warren

2013. Cost-sensitive learning for large-scale hierarchical classification. Pp. 1351–1360. cited By 5. Citado na página 25.

Cortez, E., M. R. Herrera, A. S. da Silva, E. S. de Moura, and M. Neubert
2011. Lightweight methods for large-scale product categorization. *Journal of the American Society for Information Science and Technology*, Pp. 1839–1848. Citado 7 vezes nas páginas 15, 16, 19, 25, 27, 28 e 35.

Goumy, S. and M. Mejri
2018. Ecommerce product title classification. In *The SIGIR 2018 Workshop On eCommerce co-located with the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2018), Ann Arbor, Michigan, USA, July 12, 2018*. Citado na página 25.

Hu, H., R. Zhu, Y. Wang, W. Feng, X. Tan, and J. X. Huang
2018. A best match knn-based approach for large-scale product categorization. In *eCOM@SIGIR*. Citado na página 15.

Joulin, A., E. Grave, P. Bojanowski, and T. Mikolov
2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*. Citado 4 vezes nas páginas 15, 25, 26 e 33.

Kozareva, Z.
2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Pp. 1329–1333, Denver, Colorado. Association for Computational Linguistics. Citado na página 19.

Mattos, A., M. Van Kampen, C. Carriço, A. Dias, and A. Crivellaro
2012. E-commerce market analysis from a graph-based product classifier. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7243 LNAI:291–297. cited By 0. Citado na página 25.

Mikolov, T., K. Chen, G. Corrado, and J. Dean
2013a. Efficient estimation of word representations in vector space. Citado na página 33.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean
2013b. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, Pp. 3111–3119. Citado na página 25.

Pearl, J.
1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Citado na página 28.

Ristoski, P., P. Petrovski, P. Mika, and H. Paulheim
2018. A machine learning approach for product matching and categorization. *Semantic Web*, 9(5):707–728. Citado na página 19.

Salton, G. and M. J. McGill
1986. *Introduction to Modern Information Retrieval*, 1 edition. McGraw-Hill, Inc. Citado na página 25.

Shankar, S. and I. Lin
2011. Applying machine learning to product categorization. Citado na página 19.

Shannon, C. E.
1948. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423. Citado na página 20.

Shen, D., J.-D. Ruvini, and B. Sarwar
2011. Large-scale item categorization for e-commerce. *eBay Research Labs Technical Report*. Citado na página 15.

Skinner, M.
2018. Product categorization with lstms and balanced pooling views. In *eCOM@SIGIR*. Citado 4 vezes nas páginas 15, 16, 25 e 29.

Tan, L., M. Y. Li, and S. Kok
2020. E-commerce product categorization via machine translation. *ACM Trans. Manage. Inf. Syst.*, 11(3). Citado 2 vezes nas páginas 25 e 27.

Vandic, D., F. Frasincar, and U. Kaymak
2018. A framework for product description classification in e-commerce. *Journal of Web Engineering*, 17(1-2):1–27. cited By 0. Citado na página 19.

Wolin, B.
2002. Automatic classification in product catalogs. *SIGIR '02*. Citado 2 vezes nas páginas 19 e 25.

Yu, W., Z. Sun, H. Liu, Z. Li, and Z. Zheng
2018. Multi-level deep learning based e-commerce product categorization. In *The SIGIR 2018 Workshop On eCommerce co-located with the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2018), Ann Arbor, Michigan, USA, July 12, 2018*. Citado 5 vezes nas páginas 15, 19, 25, 26 e 35.

Zhai, C. and J. Lafferty
2004. A study of smoothing methods for language models applied to information retrieval. Citado 3 vezes nas páginas 22, 31 e 32.

Ziegler, C.-N., G. Lausen, and L. Schmidt-Thieme

2004. Taxonomy-driven computation of product recommendations. Pp. 406–415. cited By 125. Citado na página 25.