UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

RODOLPHO ROSA DA SILVA

Aplicação de Abordagens Word Embedding ao Problema de Categorização de Ofertas

Prof. Leandro Guimarães Marques Alvim,
D.Sc.
Orientador

Rio de Janeiro, Fevereiro de 2017

Aplicação de Abordagens Word Embedding ao Problema de Categorização de Ofertas

Rodolpho Rosa da Silva

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

_____

Rodolpho Rosa da Silva

Aprovado por:

_____

Prof. Leandro Guimarães Marques Alvim, D.Sc.

_____

Prof. Fellipe Ribeiro Duarte, M.Sc.

_____

Prof. Filipe Braida do Carmo, M.Sc.

RIO DE JANEIRO, RJ - BRASIL

Fevereiro de 2017

Agradecimentos

Primeiramente, agradeço a Deus por ter me dado forças para trilhar o caminho necessário para que eu conseguisse chegar até aqui. Agradeço aos meus pais pela dedicação e apoio ao longo de toda a minha vida, e que me deram educação e respeito para que eu me tornasse um ser humano melhor.

Agradeço aos meus amigos, que sempre me ajudaram de bom grado durante toda a graduação e que fizeram desses anos os mais memoráveis e divertidos da minha vida.

Agradeço também ao meu orientador Leandro Guimarães Marques Alvim e a Eraldo Rezende Fernandes pela paciência e empenho para que pudéssemos realizar este trabalho. E agradeço a todo o corpo docente do curso de Ciência da Computação da Universidade Federal Rural do Rio de Janeiro pelo conhecimento comigo compartilhado ao longo dos anos.

RESUMO

Aplicação de Abordagens Word Embedding ao Problema de Categorização de Ofertas

Rodolpho Rosa da Silva

Fevereiro/2017

Orientador: Leandro Guimarães Marques Alvim, D.Sc.
DCC/IM/UFRRJ

O problema de categorização de ofertas objetiva inferir de forma eficiente a categoria à qual uma nova oferta pertence. Esse problema é enfrentado por sites de comparação de preços, que coletam milhões de ofertas diariamente e necessitam organizá-las entre as categorias existentes em seu sistema. Essas ofertas são organizadas de acordo com uma taxonomia de produtos, que consiste em categorias-pai e categorias-filhas, com diferentes níveis de generalização. Para a categorização são utilizados diferentes atributos das ofertas, como descrição do produto, preço e nome de loja.

Neste trabalho, propomos a utilização de duas técnicas de aprendizado de máquina em quatro modelos distintos. O primeiro deles utiliza *word embedding* não-supervisionado. Esses vetores são concatenados e enviados diretamente para um classificador. O segundo modelo consiste em uma rede neural convolucional alimentada com *word embedding* não-supervionado. O terceiro modelo é uma rede convolucional que aprende os *embeddings* durante o treinamento. E o quatro extende o terceiro modelo, acrescentando informações do nome da loja e da categoria na loja.

Para avaliar as técnicas utilizadas, foi realizada uma série de experimentos, comparando-as com um classificador que utiliza *Bag of Words* como modelo de representação de dados. Os resultados demonstraram uma superioridade das Redes Neurais Convolucionais, com acurácia de 96,82%. O uso de *Word Embedding*, porém, não trouxe vantagens, monstrando-se inferior ao *Bag of Words*.

ABSTRACT

Aplicação de Abordagens Word Embedding ao Problema de Categorização de Ofertas

Rodolpho Rosa da Silva

Fevereiro/2017

Advisor: Leandro Guimarães Marques Alvim, D.Sc.
DCC/IM/UFRRJ

Offer categorization problem aims to efficiently infer the category to which a new offer belongs. This problem is faced by price comparison sites, which daily collects millions of offers and need to organize them in the categories that exist in their system. These offers are organized according to a product taxonomy, which consists of parent-categories and child-categories, with different levels of generalization. To categorization, different attributes, such as product description, price and name of the store, are used.

In this work, we propose the utilization of two machine learning techniques in four different models. The first uses unsupervised word embedding. These vectors are concatenated and sent directly to a classifier. The second model consists of a convolutional neural network fed with unsupervised word embedding. The third one is a convolutional network that learns the embedding throughout the training. And the fourth model extends the third one by adding information of the name of the store and the category in the store.

To evaluate these techniques, a series of experiments were performed, comparing them to a classifier that utilizes Bag of Words as data representation model. The results showed the superiority of Convolutional Neural Networks, that had accuracy of 96,82%. The use of Word Embedding, however, was not advantageous, and was inferior to Bag of Words.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

BOW	Bag of Words
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network com *embedding* supervisionado
CNN+ftrs	Convolutional Neural Network com *embedding* supervionado e *features*
CNNUWE	Convolutional Neural Networks com *embedding* não-supervisionado
NNLM	Neural Network Language Model
RNNLM	Recurrent Neural Network Language Model
SPC	Site de Comparação de Preços
SVM	Support Vector Machine
UWE	Unsupervised Word Embedding

Sumário

Capítulo 1

Introdução

Na rede mundial de computadores há uma quantidade crescente de lojas online oferecendo milhões de produtos de diferentes tipos. Mas, apesar do comércio eletrônico reduzir os problemas da barreira física, surgem, porém, outros problemas, como busca, análise e comparação de produtos (Wong et al., 2008). Diante disso, nasce uma nova categoria de *site*: os *comparadores de preços*.

Um site de comparação de preços (SCP) é um mecanismo de pesquisa usado para filtrar e comparar ofertas de diferentes varejistas considerando alguns critérios, tais como: preço, características do produto e reputação do varejista. Companhias de SCP obtêm seus lucros através de anunciantes, que pagam por cliques em anúncios de suas ofertas. No estágio inicial de seu desenvolvimento, entre 1995 e 2000, esses *websites* eram capazes de somente comparar ofertas. Hoje em dia, SCP provêm serviços de recomendação, avaliação e *review* de produtos, os quais se tornaram fundamentais para usuários.

Esse modelo de *site* tem como principal missão coletar, classificar e recomendar aos usuários ofertas de *sites* de comércio eletrônico de acordo com sua pesquisa. Diariamente, milhões de ofertas passam por esse processo, que se inicia com um *crawler*, que coleta e extrai características, como descrição, preço, nome da loja e outros, e termina em um buscador (Wong et al., 2008). Por fim, o sistema classifica ofertas em categorias para facilitar a usabilidade dos usuários. Geralmente, essas

categorias estão organizadas em uma estrutura hierárquica. De acordo com Abels et al. (2006); Grabowski et al. (2002); Wolin (2002), um humano demora de 5 a 10 minutos para categorizar uma única oferta, e isso deve custar em torno de US$ 0,25. Dado o grande volume de ofertas produzidas por lojas de vendas *online* nos dias de hoje, é impraticável realizar essa tarefa manualmente. Assim, criar classificadores de ofertas acurados é altamente valioso.

Tendo em vista essa necessidade, viramos nossa atenção para o algoritmo de classificação. Nessa etapa, é necessário definir um modelo de representação textual que seja mais compreensível à maquina e de mais fácil manipulação. Um dos modelos mais antigos e amplamente utilizados para essa finalidade é o *Bag of Words* (Harris, 1954). Embora esse modelo demonstre alta qualidade em tarefas de classificação, ele possui algumas desvantagens, como perda da ordem das palavras e possibilidade de uma mesma representação ser compartilhada por sentenças distintas, desde que elas possuam as mesmas palavras. Além disso, a semântica das sentenças não é impressa em suas representações (Le and Mikolov, 2014).

1.1 Objetivos e Metodologia

Neste trabalho, objetivamos demonstrar a qualidade das técnicas de aprendizado de máquina aplicando-as ao problema de categorização de ofertas, de maneira que possamos construir um classificador que aprenda, de maneira eficiente, características através do texto e gere resultados mais acurados. Para isso, fazemos a aplicação de duas técnicas: *Unsupervised Word Embedding* e Redes Neurais Convolucionais.

Utilizamos quatro modelos distintos. O primeiro deles, utiliza *Word Embedding* não-supervisionado. Os vetores gerados são concatenados e enviados diretamente para um classificador Gradiente Descendente Estocástico. O segundo, consiste em uma Rede Neural Convolucional cuja entrada é formada pela concatenação de *word embedding* não-supervisionado. O terceiro, é uma rede convolucional que aprende os *embeddings* ao longo do treinamento. Os três utilizam apenas o texto da descrição de oferta para classificação. O quarto e último modelo consiste em uma rede

convolucional que aprende os *embeddings* de forma supervisionada e utiliza, além da descrição da oferta, as informações do nome da loja e da categoria na loja.

Primeiramente, realizamos uma análise do impacto dos atributos das ofertas na performance dos classificadores. SCP extraem vários atributos de ofertas, e eles podem ser úteis para categorização. Contudo, algumas ofertas não possuem todos os atributos, como *nome da loja* e *categoria na loja*, e o impacto real de cada um deles é incerto.

Realizamos uma análise dos diferentes métodos de representação de texto. A performance dos algoritmos de aprendizado de máquina depende fortemente da representação da entrada. Recentemente, abordagens de *word embedding* têm sido aplicadas com êxito para representar dados textuais. Alguns métodos superam problemas da abordagem *bag of words* tradicional, como a maldição da dimensionalidade. Não temos conhecimento de trabalhos na literatura que apliquem *word embedding* para categorização de ofertas. Assim, analisamos diferentes aspectos da aplicação dessa técnica sobre esse problema.

Comparamos uma simples abordagem *word embedding* com um modelo *bag of words*, tomando como parâmetro as k primeiras palavras da descrição da oferta. Essa decisão foi necessária devido à dimensionalidade dos *embeddings* e o tamanho do conjunto de dados. Também fazemos o uso de uma abordagem que aplica convolução ao longo da descrição. O método de convolução é incorporado a uma rede de aprendizado profundo cujos parâmetros são aprendidos através de *back-propagation*. Comparamos também uma abordagem que utiliza *word embedding* não-supervisionado com outra que os aprende durante o treino supervisionado da rede.

1.2 Resumo dos Resultados

Na Tabela 1.1, apresentamos a performance dos nossos melhores modelos UWE e CNN, e da nossa melhor linha de base – o modelo BOW+ftrs. Os modelos CNN+ftrs e BOW+ftrs usam a descrição da oferta e *features* adicionais (e.g., o nome da loja). O

modelo CNN+ftrs superou substancialmente a performance do modelo BOW+ftrs, com quase 40% de redução de erro na acurácia e quase 50% de redução de erro no macro F1.

Métrica	UWE	BOW+ftrs	CNN+ftrs
Acurácia	89,09	94,71	96,82
Macro F1	79,26	88,13	93,77

Tabela 1.1: Performance dos melhores modelos comparados ao *BOW*.

O modelo *UWE* teve desempenho inferior ao modelo *BOW* mais simples, i.e., que utiliza apenas a descrição da oferta.

1.3 Principais Contribuições

- *Estudo sobre Classificação de Ofertas*

  Este trabalho contribuiu como material de estudo em língua portuguesa para o problema de Classificação de Ofertas. Também pudemos promover uma análise desse problema utilizando dados do mundo real.

- *Análise do impacto das informações de ofertas para os classificadores*

  Outra importante contribuição obtida através desse trabalho encontra-se na análise dos atributos das ofertas e sua influência nos resultados dos experimentos. Além disso, pudemos demonstrar a engenharia de *features* utilizada e validá-la.

- *Comparação entre abordagens estudadas*

  Por fim, os experimentos por nós executados contribuíram para comparar as diferentes metodologias utilizadas em problemas que envolvem processamento de linguagem natural.

1.4 Organização da Dissertação

Este trabalho está organizado da seguinte forma: no capítulo 1 fazemos uma breve introdução do problema de Categorização de Ofertas, objetivos e metodologias, resumo dos resultados e principais contribuições. No capítulo 2 fazemos uma descrição mais detalhada do problema de classificação de ofertas aqui abordado e definimos os conceitos de oferta e categoria, cujo entendimento é imprescindível para o andamento desse trabalho. Em 3, descrevemos os modelos utilizados neste trabalho e a engenharia de *features* aplicada no tratamento dos dados. No capítulo 4, fazemos a apresentação das técnicas de aprendizado de máquina abordadas neste trabalho. Na primeira parte, faremos a introdução da metodologia *Unsupervised Word Embedding*; descrevemos os conhecimentos necessários para o seu desenvolvimento e fazemos uma breve revisão da literatura, frisando o surgimento e evolução da técnica, e o estado da arte. Em seguida, apresentamos as Redes Neurais Convolucionais, descrevendo a metodologia e fazendo uma revisão dos principais trabalhos presentes na literatura. No capítulo 5, apresentamos a base de dados da Buscapé, o *framework* Gensim e a descrição da metodologia utilizada durante os experimentos, e fazemos uma análise dos resultados obtidos. E por fim, no capítulo 6, apresentamos nossas conclusões e trabalhos futuros.

Capítulo 2

Categorização de Ofertas

Os comparadores de preço nasceram diante do crescimento do comércio eletrônico (*e-commerce*). Embora essa nova modalidade de comércio tenha transposto as barreiras físicas da prática, outros problemas surgiram. Diante da quantidade crescente de lojas de vendas *online*, pesquisar e comparar preços de produtos oferecidos têm se tornado uma tarefa laboriosa. Assim, os comparadores de preço se propõem a solucionar essa impasse coletando, classificando e retornando aos usuários as melhores ofertas existentes na rede mundial de computadores (Wong et al., 2008).

Esse modelo de *site* seleciona o maior número de lojas de vendas *online*, buscando os melhores preços e condições de venda. Dezenas de milhões de ofertas são coletadas diariamente. E, a fim de recomendá-las aos usuários como resultados de suas buscas, é necessário, antes de tudo, categorizar, de maneira eficiente e acurada, o maior número de ofertas por hora. Em alguns SCP, esse trabalho era feito com intervenção humana; contudo, com o crescimento do comércio eletrônico, isso tornou-se impraticável. Dessa forma, o desenvolvimento de um algoritmo rápido e eficiente é imprescindível.

Surge, agora, uma nova questão a ser resolvida, referente à medição da qualidade da classificação automática. À primeira vista, métricas clássicas de aprendizado de máquina, como f1, *precision* e *recall*, são suficientes. Todavia, no domínio do

problema, erros na centésima página, quase não visualizada, devem ter peso menor quando comparados a erros que aparecem na primeira página de pesquisa. Outra questão importante é a *taxonomia de produtos*. As ofertas são representadas por uma árvore n-ária. Dessa forma, erros que acontecem entre *categorias-pai* são mais críticos que erros entre *categorias-filhas*. Por exemplo, classificar uma TV como *calçado* é pior que classificar um *smartphone* como um aparelho telefônico regular.

2.1 Taxonomia de Produtos

Dentro do escopo do problema aqui abordado, os produtos das ofertas são organizados segundo uma taxonomia. De forma geral, a taxonomia de produtos é representada por uma árvore que classifica os produtos de forma que cada nível, a partir das folhas, aumenta o grau de generalização (Cho and Kim, 2004). Em outras palavras, cada nó-folha equivale a um produto do conjunto de dados, ao passo que cada nó não-folha, uma categoria, obtida pela combinação de vários nós de um nível mais baixo em um. Cada produto é categorizado por sua natureza e finalidade; dessa forma, os nós não-folhas dos níveis mais baixos agrupam produtos de forma mais granulada, considerando características mais específicas. E a categorização torna-se mais generalizada à medida que se aproxima da raiz.

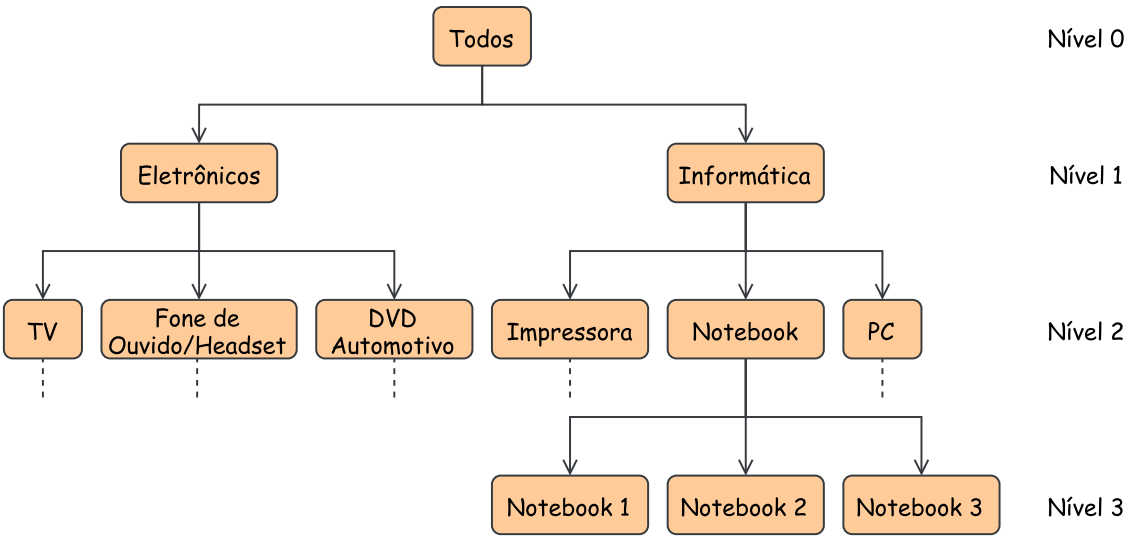

Figura 2.1: Exemplo da taxonomia de produtos.

A Figura 2.1 mostra um exemplo da taxonomia. Ela possui quatro níveis, tendo, portanto, dois tipos de categorias. No primeiro nível (nível 0) está a raiz, que engloba todas as categorias. No nível 1 encontram-se as *categorias-pai*. E no terceiro nível, estão as *categorias-filha*.

2.2 Oferta: o que é?

Uma oferta pode ser definida como um conjunto de informações que descrevem um produto. A Tabela 2.1 apresenta um exemplo de oferta do mundo real presente no conjunto de dados por nós utilizado. O primeiro atributo identifica a categoria da oferta. Em particular, essa oferta pertence à categoria *Tênis*. O atributo seguinte é a descrição da oferta. Geralmente, esse atributo contém a marca, o modelo e a cor do produto relacionado à oferta. A seguir, temos o preço da oferta. Os dois últimos atributos correspondem ao nome da loja e uma categoria interna opcional que é dada pelo vendedor.

ID da Categoria	2858
Descrição	Tênis Nike Air Visi Branco
Preço	199.99
Nome da Loja	Dafiti
Categoria na Loja	Calçados

Tabela 2.1: Exemplo de oferta do mundo real.

2.3 Trabalhos Relacionados

Wolin (2002) apresenta o sistema AutoCat para categorização de produtos. Esse modelo utiliza uma variação do modelo espacial vetorial de modo que atributos como descrição, nome da loja e categoria possam ser representadas. Essas *features* são combinadas através de um método de otimização de pesos a fim de gerar um *ranking* onde a categoria de maior peso é atribuída ao produto. O sistema opera sobre uma matriz esparsa contendo referências de termos indexados a uma lista de categorias

nas quais eles foram observados. Dessa forma, produtos são agrupados em vetores de categorias. O melhor resultado obtido foi de 79,5% de acurácia utilizando todas as *features* estudadas. O preço foi o atributo que menos contribuiu para os resultados.

Em Pavlov et al. (2004) os autores solucionam o problema de classificação de produtos do Yahoo! Shopping utilizando o algoritmo *Naïve Bayes*. Devido à estrutura da representação dos produtos e da quantidade de categorias, esse problema se mostra um desafio para *SVMs*. E esse trabalho tem como objetivo superar as limitações dessa metodologia estudando a eficiência de transformações nos dados usando um classificador *Naïve Bayes*. Uma série de experimentos com diferentes transformações foi executada sobre conjuntos de dados reais contendo pouco menos de 100000 produtos e 58 categorias. Os resultados mostraram que algumas delas foram capazes de alavancar significativamente a acurácia do classificador.

Cortez et al. (2011) apresenta um estudo sobre categorização de ofertas utilizando um modelo probabilístico. A abordagem utilizada procura pela melhor combinação entre uma oferta não vista, representada pelos seus atributos ou características, e as categorias em uma dada taxonomia, representada pelos atributos/características das ofertas conhecidas que pertencem a essas categorias. As ofertas são compostas pelos campos descrição, preço e nome da loja. Um cálculo probabilístico é feito sobre cada um desses campos. Essas probabilidades são combinadas para estimar a probabilidade da oferta pertencer a cada categoria. Dessa forma, a oferta é atribuída à categoria com maior probabilidade. Experimentos foram realizados utilizando dois conjuntos de dados do mundo real. Os autores concluíram que o preço e o nome da loja são úteis no processo de classificação, mas provêm pouca melhoria quando comparados com a descrição da oferta.

Pode-se notar que o problema de classificação de ofertas, embora não seja recente, ainda é pouco explorado na literatura, sendo, então, uma importante base para novos estudos.

A seguir, faremos uma descrição teórica das técnicas utilizadas neste trabalho.

Capítulo 3

Modelos

3.1 Bag of Words

O modelo *Bag of Words* é provavelmente a abordagem mais popular utilizada para representar dados textuais para alimentar algoritmos de aprendizado de máquina (Harris, 1954). Nessa técnica, uma entrada de texto é representada como um vetor binário em um espaço N-dimensional, onde N é o número de palavras distintas que ocorrem nos dados (as quais são geralmente da ordem de centenas de milhares). Cada palavra do vocabulário é associada a um índice no vetor de entrada.

Assim, dada uma descrição de oferta, sua representação consiste em um vetor esparso onde os índices de cada palavra da descrição tem um valor igual a um, e os índices restantes valor igual a zero. A maior desvantagem dessa técnica é a esparsidade, uma vez que a maior parte das descrições das ofertas consiste em menos de 15 palavras, enquanto o tamanho do vetor é igual a centenas de milhares.

Como apresentado no capítulo 2, uma oferta é composta por um conjunto de atributos (como descrição, nome da loja, etc.) das quais derivamos as *features*. E a partir dos experimentos aqui administrados, buscamos realizar uma análise do impacto de cada uma dessas informações no desempenho do algoritmo de classificação. Para tanto, cada *feature* foi tratada de modo que ela forme um vocabulário que caracterize a informação por ela representada e que alimentará os algoritmos aqui

utilizados.

Dessa forma, consideramos aqui cinco *features* geradas a partir dos dados das ofertas. Suas respectivas descrições são apresentadas a seguir.

- *Primeira palavra*:
  A primeira palavra de uma descrição de oferta ajuda a evitar a ambiguidade. Por exemplo, as ofertas

  Exemplo 1. *regata triton reta renda vinho roupas blusas feminino, 234.0, Dafiti, Moda e Acessórios*

  Exemplo 2. *vinho frances mouton cadet reserve st emilion rothschild 19591 1 cod 1870, 135.0, Loja de Bebidas, Alimentos e Bebidas*

  têm ambas a palavra *vinho* na sua descrição. No Exemplo 1, a palavra *vinho* significa o nome de uma cor, ao passo que, no Exemplo 2, o nome de uma bebida. No Exemplo 2, a palavra *vinho* está mais relacionada à categoria da oferta; e ela é a primeira palavra da descrição. Contudo, no Exemplo 1, essa palavra aparece no meio da descrição, e a primeira palavra da descrição é *regata*, que é altamente relacionada à categoria dessa oferta. Portanto, a primeira palavra é adicionada duas vezes ao vocabulário utilizado: a primeira sem processamento, e a segunda, com o acréscimo de um caractere especial para indicar que esta é a primeira palavra do texto da descrição.

- *Bigramas*:
  Ainda considerando os Exemplos 1 e 2, é notável que o contexto de algumas palavras é importante. Por exemplo, os bigramas *renda vinho* e *vinho francês* incluem a palavra *vinho*. Contudo, ela tem significados diferentes devido ao seu contexto. Assim, a partir do texto da descrição da oferta, formamos um conjunto de bigramas consecutivos. Dessa forma, além de considerarmos as palavras individualmente, teremos também a informação da vizinhança de cada uma delas nas ofertas.

- *Categoria na loja*:

  Algumas ofertas incluem uma categoria interna dada pela loja. Embora não seja uma regra, essa categoria está frequentemente relacionada ao nosso conjunto de categorias. Por exemplo, as ofertas

  Exemplo 3. *bota casual cavalera andy 13010805 chocolate, Masculino Calcados Bota Casual, 219.99, Passarela Calçados*

  Exemplo 4. *nutri whey protein chocolate 907g integralmedica, Whey Protein, 48.4, OtimaNutri.com.br*

  têm ambas a palavra *chocolate* na sua descrição. Mas, como nos exemplos anteriores, essa palavra tem dois significados diferentes de acordo com o contexto em que aparece. Na primeira oferta, é o nome de uma cor, enquanto na segunda, o nome de um sabor. Contudo, a primeira oferta é da categoria *Masculino Calçados Bota Casual*, e a segunda, da categoria *Whey Protein*. As palavras que compõem essa *feature* são concatenadas e marcadas com caracteres especiais a fim de que possamos distingui-las das outras *features*.

- *Categoria por preço*:

  Cada oferta do conjunto de dados tem como informação o preço do produto. A partir desse dado, podemos calcular o preço médio para cada categoria, seja pai ou filha. E com base nessa informação, é possível inferir a qual categoria uma oferta pode pertencer. Por exemplo, o preço médio das ofertas da categoria *Papelaria e Escritório* é igual a R$ 4,26; esse valor é significativamente inferior ao preço médio da categoria *Fotografia*, que tem preço médio de R$ 113,59.

  Portanto, dado o preço de uma oferta, calculamos a distância absoluta entre esse e o preço médio de cada uma das categorias. A menor distância encontrada indica a qual categoria essa oferta deve, provavelmente, pertencer. A categoria retornada se torna uma palavra do vocabulário.

- *Nome da loja*:

  Finalmente, uma *feature* relevante é o nome da loja de onde a oferta é origi-

nária. Há lojas que vendem produtos específicos e às vezes seu nome nos dá uma dica da categoria da oferta. Por exemplo, as ofertas

Exemplo 5. *imaginext cars 2 cod 4627, 99.99, Panda Brinquedos*

Exemplo 6. *barbie bailarina cod 4817, 34.99, Panda Brinquedos*

são de uma mesma loja, que vende brinquedos e produtos relacionados.

Na Tabela 3.1, nós utilizamos um exemplo de oferta do mundo real para sumarizar o processo apresentado nesta seção.

Oferta	lacoste elegance edt masculino embalagem 50 ml, Perfumes lacoste, 149.9, Import Perfume
Primeira Palavra	^ lacoste$
Bigramas	lacoste+elegance, elegance+edt, edt+masculino, masculino+embalagem, embalagem+numero, numero+ml
Categoria na Loja	*perfumes+lacoste*
Categoria por Preço	fotografia_por_preco
Nome da Loja	**import+perfume**

Tabela 3.1: Engenharia de *features* aplicada em uma oferta do mundo real.

3.2 Word Embedding

Abordagens com *Word Embedding* têm sido aplicadas com sucesso em diversos problemas que envolvem dados textuais (Collobert et al., 2011; Bengio and Heigold, 2014). A ideia básica é "embutir" o espaço vetorial *BOW* original em um espaço vetorial com menor dimensão. Nesse espaço, cada palavra é representada por um vetor de valor real denso.

Para a maior parte das tarefas, o tamanho da representação vetorial das palavras varia entre algumas dúzias a algumas centenas; o que é muito menor que o tamanho do espaço vetorial original do *BOW*. Dessa forma o *word embedding* objetiva

solucionar a maldição da dimensionaliade, utilizando um espaço vetorial de menor dimensão (Bengio et al., 2003).

A ferramenta *word2vec* implementa um algoritmo bastante eficiente para aprendizado não-supervisionado de representações vetoriais a partir de um *corpus* grande. Basicamente, dado um *corpus* grande (milhões de palavras), *word2vec* aprende vetores de palavras que são bons em discriminar contextos reais (sequências de palavras presentes no *corpus*) de contextos com ruído (contextos obtidos a partir da substituição de alguma palavra dentro da sequência por uma palavra aleatória). Empregando uma técnica de *word embedding*, evita-se o problema da esparsidade presente nos modelos *BOW*. Contudo, o uso de tais representações não é direto, porque o tamanho da descrição varia de oferta em oferta, e os algoritmos de aprendizado de máquina requerem uma representação de tamanho fixo.

Neste trabalho, propomos a abordagem *Word Embedding* não-supervisionada que utiliza as k primeiras palavras da descrição da oferta. Dessa forma, a dimensionalidade da entrada é igual a $k \cdot d$, onde d é o tamanho dos vetores de palavras. Se uma descrição de oferta é menor que k palavras, incluímos cópias de uma palavra artificial para obter uma descrição de tamanho k. Assim, alimentamos o classificador com essa representação.

A Figura 3.1 descreve a arquitetura do *word2vec* usando o algoritmo *CBOW* aplicada sobre o texto *Tênis Nike Air Visi Branco*. Tomamos como alvo a palavra *Air* e como contexto as demais palavras. Os valores apresentados são aleatórios, utilizados apenas por motivos didáticos.

Esse modelo tem como objetivo calcular a probabilidade de uma palavra ocorrer dado o contexto em que essa palavra aparece. Dessa forma, tomamos um conjunto de k palavras vizinhas (anteriores e/ou posteriores), e almejamos, a partir da representação vetorial binária das palavras do contexto (*context*), chegar à representação da palavra alvo (*target*); em outras palavras, a probabilidade dessa palavra deve ser igual a um, e das demais palavras do vocabulário, igual a zero.

A camada de entrada ($\mathrm{L}^{(1)}$) recebe c vetores binários de dimensão $1 \times v$, onde

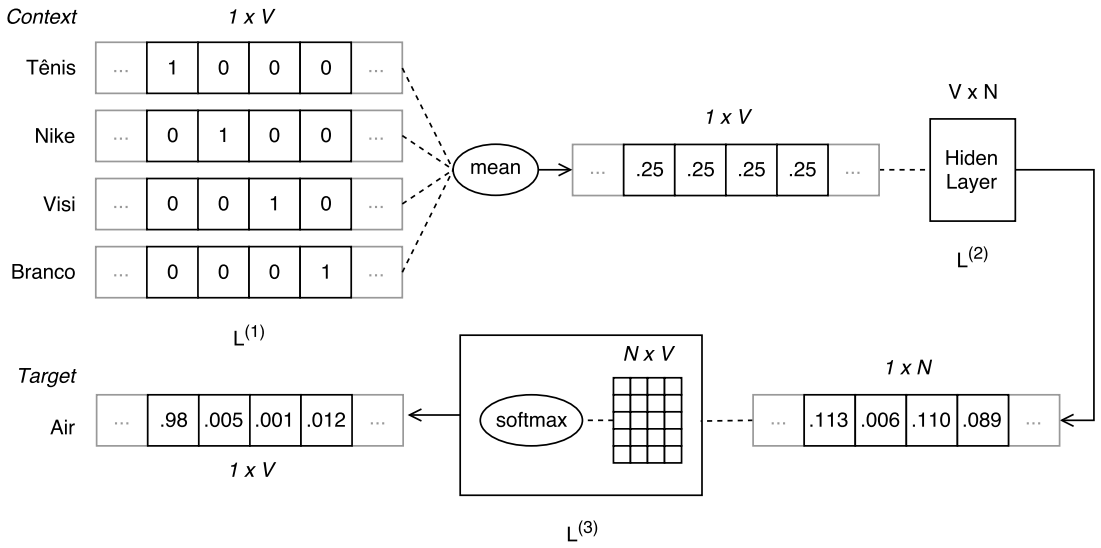

Figura 3.1: Descrição do modelo UWE aplicado ao texto *Tênis Nike Air Visi Branco*.

v é o tamanho do vocabulário. Cada um desses vetores corresponde a uma palavra do contexto. Esses vetores são somados e o resultado dividido por c. Temos assim, um único vetor $1 \times v$, que corresponde à media dos vetores de entrada. Esse vetor é dado como entrada para uma camada escondida. Essa camada pode ser representada por uma matriz de pesos *reais* aleatórios de dimensão $v \times n$, onde n corresponde à dimensão do espaço vetorial, i.e, o número de características aprendidas. A matriz é multiplicada pelo vetor. O resultado é um novo vetor $1 \times n$, que servirá de entrada para a camada de saída.

A camada de saída contém uma nova matriz $n \times v$ de pesos aleatórios. O vetor de entrada é multiplicado por essa matriz, gerando um novo vetor $1 \times v$. Uma vez que o objetivo dessa camada é calcular a probabilidade das palavras segundo o contexto, é necessário que a soma dos valores desse vetor seja igual a um. *Word2vec* consegue isso aplicando, sobre estes valores, uma função de *softmax*. A partir do vetor gerado, o erro pode ser calculado e as matrizes de pesos são atualizadas através de *backpropagation*.

3.3 Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNN) têm sido estudadas há algumas décadas e têm sido aplicadas em muitos problemas que envolvem imagem. Uma CNN também pode ser aplicada sobre dados sequenciais, como textos (Collobert and Weston, 2008), ao invés de dados bidimensionais, como imagens.

O aspecto principal de uma CNN é a invariância espacial. No caso de textos, isso quer dizer que ela pode aprender características que independem da posição da palavra dentro do texto de entrada. Por exemplo, uma CNN é capaz de aprender que o termo *Tênis* é altamente relacionado à categoria *Moda e Acessórios*, independentemente de sua posição dentro da descrição da oferta. Essa é uma vantagem crucial para uma CNN em comparação a UWE.

Neste trabalho, aplicamos um modelo baseado em CNN para categorização de ofertas. Demonstramos tal modelo através da Figura 3.2.

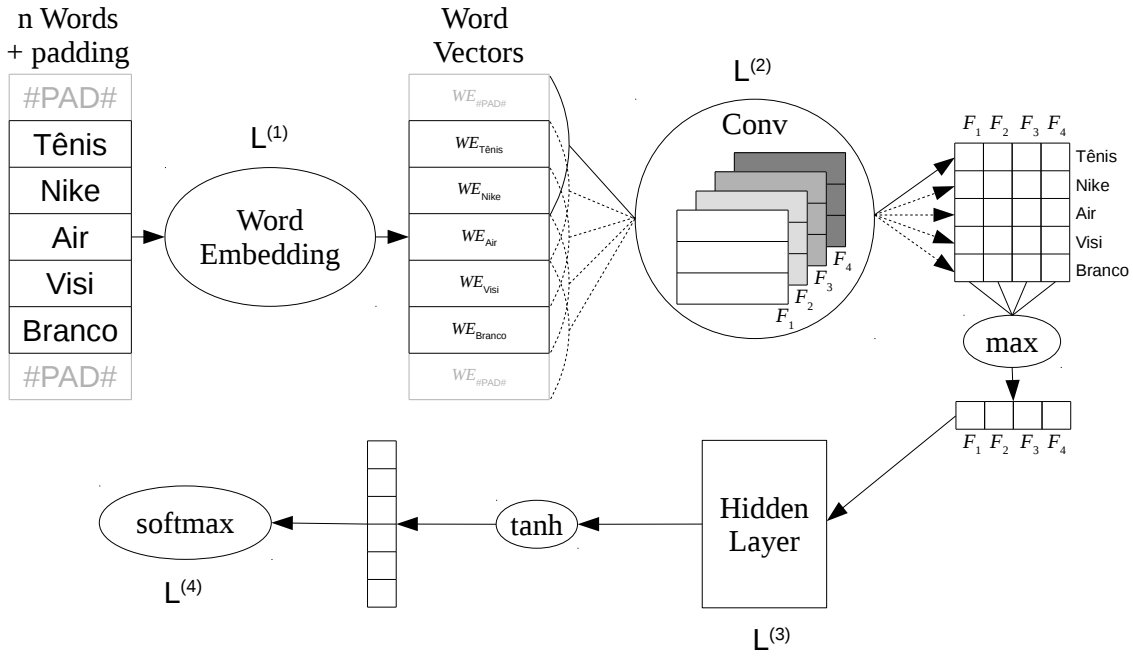

Figura 3.2: Descrição do modelo CNN aplicado ao texto *Tênis Nike Air Visi Branco*.

Esse é um modelo de aprendizagem profunda que inclui uma camada de convolução ($L^{(2)}$). A rede recebe como entrada a descrição de uma oferta, aumentada com palavras de preenchimento (*padding*) no início e no final do texto, resultando em

uma sequência de n palavras. Uma camada de *embedding* ($L^{(1)}$) toma a sequência como entrada e gera uma matriz $n \times d$ formada pela concatenação dos vetores que representam todas as n palavras da entrada. Cada vetor está no espaço R^d. Assim, a camada de convolução $L^{(2)}$ escaneia a sequência aplicando um conjunto de filtros $(F_1, F_2, \ldots, F_r)$ a cada palavra *real*. A entrada de cada filtro é, na realidade, uma janela de t palavras consecutivas (na figura, $t = 3$), compreendendo uma matriz $t \times d$. Cada filtro é também uma matriz $t \times d$, que é multiplicada, elemento-a-elemento, por cada janela de palavra, e os resultados são somados, resultando em um valor único para cada combinação de janela e filtro.

Dessa forma, aplicando os r filtros ao longo de janelas de n palavras, geramos uma matriz $n \times r$. Uma vez que n varia de oferta em oferta, aplicamos uma operação de *pooling* máximo sobre esta matriz. Essas operações selecionam, para cada filtro, o valor máximo ao longo de todas as palavras da sequência de entrada. Assim, a saída da operação de *pooling* máximo consiste em r valores. Uma camada escondida recebe esse vetor de tamanho r e aplica sobre ele tangente hiperbólico (tanh). Finalmente, a saída da camada escondida é dada como entrada para uma camada de *softmax* que tem como saída a predição da rede, que é dada por uma distribuição de probabilidade sobre as categorias.

Os parâmetros dessas camadas podem ser juntamente treinados através de *back-propagation* por meio de gradiente descendente estocástico ou outro algoritmo de aprendizagem. Os parâmetros aprendidos podem ainda consistir no *word embedding*. Essa é uma vantagem da CNN em relação ao UWE. Ao passo que na última abordagem o *embedding* é aprendido de forma não-supervisionada, na primeira, o *embedding* pode ser aprendido usando *feedback* supervisionado, o que pode ser bastante benéfico. Demonstramos nos experimentos que CNN é substancialmente superior ao UWE e também ao BOW.

Capítulo 4

Word Embedding

Este capítulo destina-se à apresentação dos principais conceitos que compõem as técnicas de aprendizado de máquina utilizadas no presente trabalho.

4.1 Introdução a Word Embedding

Nesta seção, descreveremos alguns dos principais algoritmos de aprendizado de *word embedding*, e apresentaremos uma breve revisão literária acerca do método.

4.1.1 Feedforward Neural Network Language Model

O modelo proposto por Bengio et al. (2003) é composto por uma rede neural de alimentação direta que tem como objetivo converter palavras de um contexto para representações no espaço vetorial contínuo e calcular a probabilidade condicional dessas palavras. Esse modelo é conhecido como *FeedForward Neural Network Language Model* (NNLM).

Além de uma camada de entrada e uma de saída, a rede possui duas camadas internas. A primeira delas, de projeção, corresponde à representação contínua de todas as palavras do contexto. A segunda camada, ou camada escondida, é necessária para atingir uma estimativa da probabilidade não-linear.

Seja $h_j = w_{j-n+1}^{j-1}$ um conjunto de $n-1$ palavras do contexto; P a dimensão dos atributos da camada de projeção; M e b as matrizes de pesos e viéses entre a camada de projeção e a camanha escondida, respectivamente; H, a dimensionalidade da camada escondida; e V e k as matrizes de pesos e viéses entre a camada escondida e a camada de saída, respectivamente. A Figura 4.1 apresenta a arquitetura desse modelo. Um *perceptron* de múltiplas camadas é utilizado.

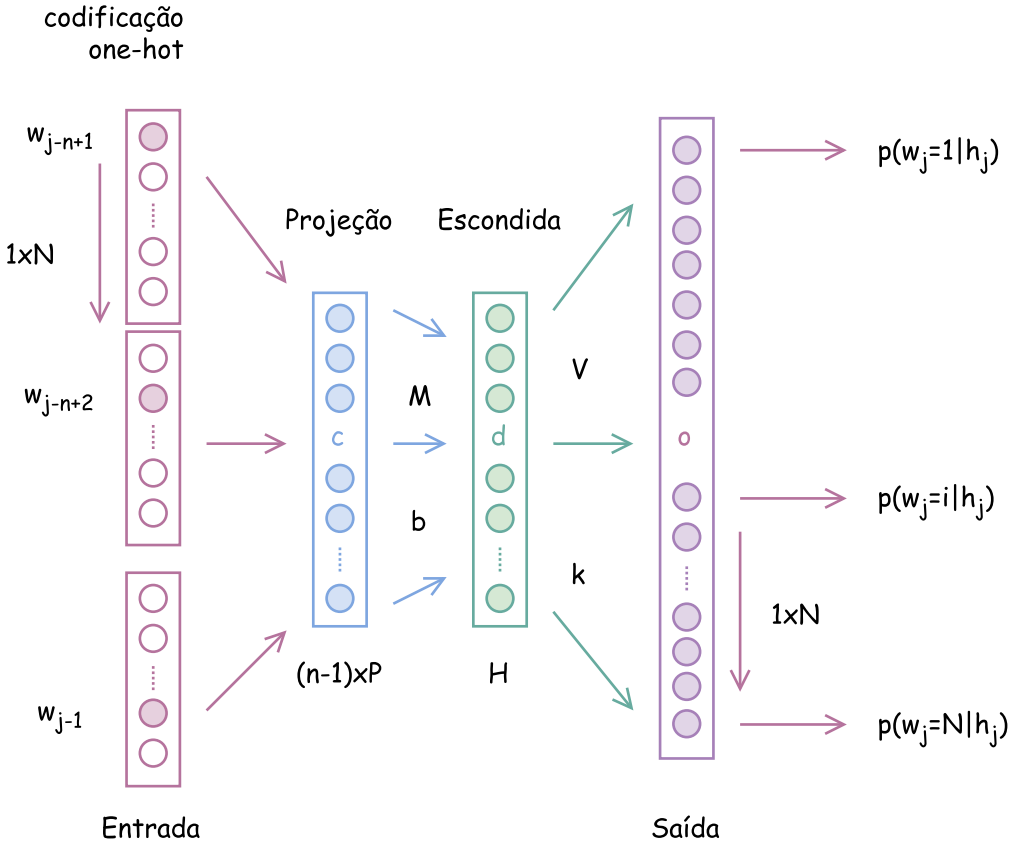

Figura 4.1: Feedforward Neural Network Language Model.

Segundo Arisoy et al. (2012), a entrada da rede é um conjunto de $n-1$ palavras predecessoras, isto é, do histórico, representadas por vetores esparsos onde apenas o índice da palavra no vocabulário é 1 e os demais valores são 0. A essa representação dá-se o nome de codificação *one-hot*. Cada vetor esparso é mapeado, por meio de projeções lineares, para um vetor de números reais. Essa projeção dá-se através de uma matriz de pesos de dimensões $N \times P$, sendo N o tamanho do vocabulário e P, a dimensão dos atributos. Essa matriz é compartilhada com todas as palavras do contexto, ou seja, a matriz de pesos é a mesma quando projetando w_{j-n+1},

w_{j-n+2}, etc. A concatenação desses vetores reais forma a camada de projeção; a i-ésima linha da matriz resultante dessa etapa corresponde, então, à representação da i-ésima palavra.

A camada escondida utiliza tangente hiperbólica como função de ativação. Seja c_l, com $l = 1, \ldots, (n-1) \cdot P$, as ativações lineares da camada de projeção. Schwenk (2007) define as ativações da segunda camada escondida por:

$$d_j = \tanh\left(\sum_l m_{jl} c_l + b_j\right), \forall j = 1, \ldots, H, \tag{4.1}$$

onde m_{jl} são os pesos entre a camada de projeção e a segunda camada interna e b_j são os vieses da segunda camada escondida. As operações da camada de saída e a probabilidade $P(w_j = i|h_j)$ de uma palavra dado seu contexto são calculadas como segue:

$$o_i = \sum_j v_{ij} d_j + k_i, \tag{4.2}$$

$$p_i = \frac{e^{o_i}}{\sum_{l=1}^{N} e^{o_l}}, \forall i = 1, \ldots, N, \tag{4.3}$$

onde v_{ij} são os pesos entre a segunda camada interna e a camada de saída, e k_i são os vieses da camada de saída. A equação 4.3 é conhecida como *softmax*. Dado um conjunto de valores, essa função tem como objetivo normalizá-los no espaço $[0, 1]$ de modo que a soma dos valores normalizados seja igual a 1.

Segundo Schwenk (2007), o treinamento da rede é realizado com algoritmo *back-propagation*, minimizando a seguinte função de erro:

$$E = \sum_{i=1}^{N} t_i \log p_i + \epsilon \left(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2\right), \tag{4.4}$$

onde t_i denota a saída desejada da rede neural, isto é, a probabilidade deve ser 1.0 para a palavra seguinte na sequência de treinamento e 0.0 para as demais.

Segundo Mikolov et al. (2011), essa arquitetura se torna computacionalmente complexa entre as camadas de projeção e a segunda camada interna, uma vez que os valores da camada de projeção são densos. Ademais, a segunda camada interna computa probabilidades condicionais para todas as palavras do vocabulário, resultando em uma camada de saída de dimensionalidade V. A complexidade computacional desse modelo é dominada por $H \times N$ multiplicações na camada de saída. Várias soluções práticas já foram propostas para contornar isso. Entre elas estão o uso de versões hierárquicas do *softmax*, e a implementação de modelos que não realizam normalização.

4.1.2 Recurrent Neural Network Language Model

Outro modelo proposto na literatura é o *Recurrent Neural Network Language Model* (RNNLM). Visando superar algumas limitações do NNLM, como a necessidade de especificar o tamanho do contexto, esse modelo pode, em teoria, representar padrões mais complexos que os demais modelos. Através de uma matriz recorrente que conecta a camada interna a si mesma, RNNLM é capaz de formar uma memória de curto prazo, de modo que a rede pode lidar com informações atuais e de passos anteriores (Mikolov et al., 2013a).

A arquitetura desse modelo é ilustrada na Figura 4.2. Essa rede não possui camada de projeção, sendo então composta por uma camada de entrada x, uma camada interna s, também chamada de camada de contexto ou estado, e uma camada de saída y. A entrada $x(t)$ da rede é formada pela concatenação da representação vetorial w da palavra atual utilizando codificação *one-hot* e a saída dos neurônios da camada de contexto no tempo $t-1$ (Mikolov et al., 2010). Os valores nas camadas são calculados como segue:

$$x(t) = w(t) + s(t-1), \tag{4.5}$$

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right), \tag{4.6}$$

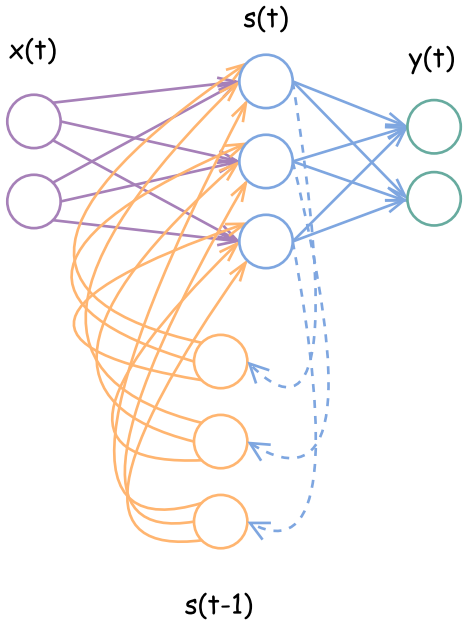

Figura 4.2: Recurrent Neural Network Language Model.

$$y_k(t) = g\left(\sum_j s_j(t) v_{kj}\right), \tag{4.7}$$

onde u_{ji} são os pesos entre a camada de entrada e a camada de contexto e v_{kj} são os pesos entre essa e a camada de saída. As funções $f(z)$ e $g(z)$ são uma função de ativação sigmoide e uma função *softmax*, respectivamente. A rede é treinada utilizando algoritmo de *backpropagation*. Mais recentemente, propôs-se o uso de *backpropagation throught time* (BPTT) (Mikolov et al., 2011).

A complexidade desse modelo é dada por $Q = H \times H \times H \times V$, onde H é a dimensão da camada interna. Analogamente ao modelo descrito acima, o termo $H \times V$ pode ser reduzido para $H \times \log_2 V$ através da utilização de uma versão hierárquica do *softmax* (Mikolov et al., 2013a).

4.1.3 Skip-gram

Um dos modelos utilizados para a computação de *word embeddings* é o *skip-gram* (Mikolov et al., 2013a). Esse modelo tem como objetivo calcular representações vetoriais que sejam capazes de predizer palavras em uma vizinhança. Essa variante

do *n-grama* permite que, dado um texto, extraia-se desse uma sequência de *tokens*, que podem ser consecutivos ou não.

Guthrie et al. (2006) demonstrou através de testes que esse modelo é capaz de captar o contexto de forma acurada. Embora o tamanho do conjunto de treinamento e o tempo de processamento tendam a ser mais extensos, o modelo *skip-gram* apresenta um bom percentual de cobertura de n-gramas em documentos de testes similares aos de treinamento e expande a informação contextual nos casos em que o conjunto de dados de treinamento são limitados.

Recentemente, Mikolov et al. (2013a) apresentou um modelo neural para aprendizado de *word embeddings* baseado nessa técnica. Nessa abordagem, cada palavra é dada como entrada para um classificador log-linear com uma camada de projeção contínua que tenta prever palavras vizinhas posteriores e anteriores a esta dentro de uma determinada distância.

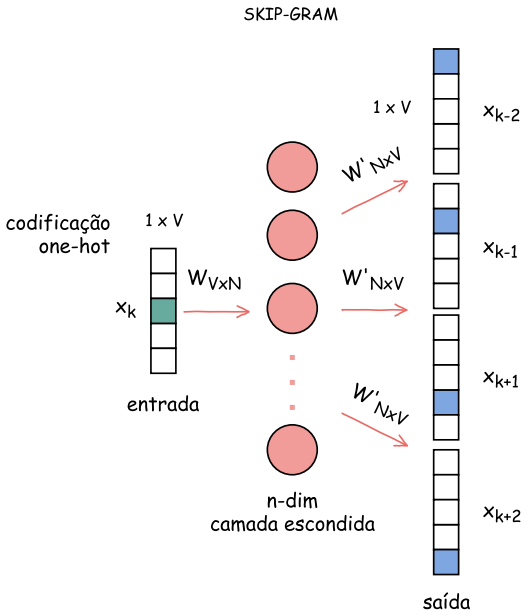

Figura 4.3: Modelo *skip-gram*.

Mais formalmente, Mikolov et al. (2013b) define que, dada uma sequência $w_1, w_2, w_3, ..., w_T$ de palavras de treinamento, o objetivo do modelo *skip-gram* é maximizar a equação

$$\frac{1}{T}\sum_{t=1}^{T}\left[\sum_{-c\leq j\leq c, j=0}\log p(w_{t+j}|w_t)\right] \tag{4.8}$$

onde c é o contexto de treinamento (que pode ser uma função da palavra central w_t). A função de probabilidade é definida com base nos parâmetros de entrada e saída v_w e $v_w^{'}$, respectivamente. Tais atributos são representações vetoriais de w. Dessa forma, a probabilidade condicional de w_O dado w_I é dada pela *softmax*

$$p(w_O|w_I) = \frac{\exp(v_{w_O}^{'\top} v_{w_I})}{\sum_{w=1}^{W}\exp(v_w^{'\top} v_{w_I})} \tag{4.9}$$

onde W é o tamanho do vocabulário. Essa formulação é custosa pois o cálculo de $\nabla \log p(w_O|w_I)$ é proporcional a W, que tende a ser grande. Mikolov et al. (2013b) propõe substituir a função *softmax* por *softmax* hierárquico, que reduz o custo do cálculo de $\log p(w_O|w_I)$ (Morin and Bengio, 2005).

4.1.4 Continuous Bag of Words

O modelo *bag of words*, ou saco de palavras, é um dos mais antigos e aplicados para representação de palavras (Harris, 1954). Nesse modelo, textos são representados como um conjunto de palavras arranjadas de maneira que a sua ordem é irrelevante, mas mantendo como informação a sua ocorrência. A partir do vocabulário formado, os documentos que compõem o *corpus* são representados como um vetor binário, onde 1 representa a ocorrência de uma palavra, e 0, a inocorrência. Por exemplo, dadas as sentenças:

Exemplo 1.

(1) A survey of user opinion of computer system response time

(2) The EPS user interface management system

obtém-se o vocabulário V = {“a”, “survey”, “of”, “user”, “opinion”, “computer”, “system”, “response”, “time”, “the”, “eps”, “interface”, “management”}. A partir do vocabulário, podemos representar as duas sentenças da seguinte maneira:

(1) [1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

(2) [0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1]

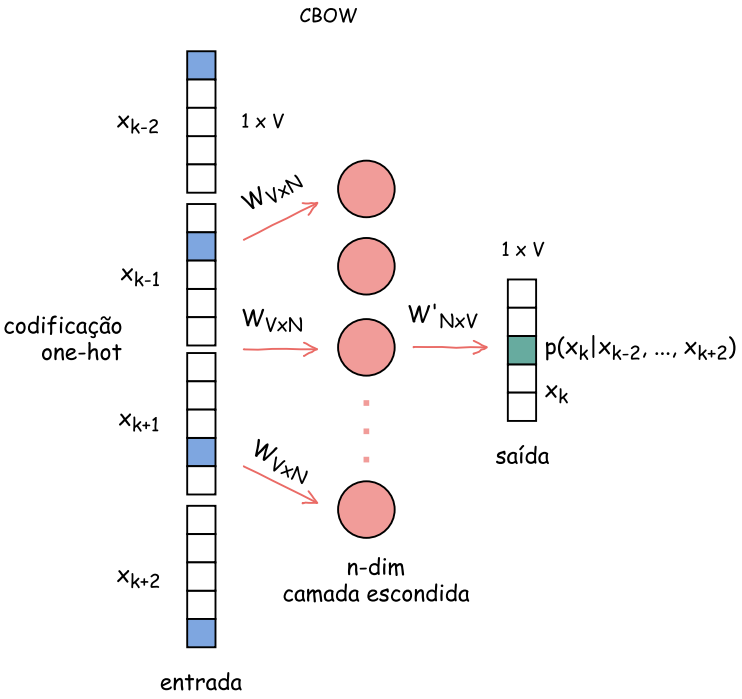

Figura 4.4: Modelo *cbow*.

Mikolov et al. (2013a) propõe uma arquitetura baseada nesse modelo, que tem como objetivo predizer a palavra seguinte com base no contexto. Similar ao *Skip-gram*, nessa arquitetura a segunda camada escondida não-linear é removida e a camada de projeção (e não apenas a matriz de projeção) é compartilhada com todas as palavras. Tal modelo é ilustrado na Figura 4.4.

4.1.5 Backpropagation

O algoritmo de *Backpropagation* (abreviação de *backward propagation of errors*) é utilizado para o aprendizado dos pesos em redes de múltiplas camadas, dada uma rede com número fixo de unidades e interconexões. Para a minimização do erro quadrático entre os valores da camada de saída e os valores esperados, utiliza-se Gradiente Descendente (Mitchell et al., 1997).

Suponhamos que tenhamos um conjunto de treinamento $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ com m exemplos de treinamento. A rede pode ser treinada utilizando Gradiente Descendente. Para um único exemplo de treinamento, a função de custo com respeito a esse exemplo é dada por:

$$J(W, b; x, y) = \frac{1}{2}\|h_{W,b}(x) - y\|^2.$$

Esta é a função de erro quadrático. A função de custo total, com relação a um conjunto de treinamento de m exemplos, é definida como segue:

$$J(W, b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W, b; x^{(i)}, y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{(l)})^2 =$$

$$= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|h_{W,b}(x^{(i)}) - y^{(i)}\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{(l)})^2.$$

O primeiro termo é a média da soma dos erros quadráticos, enquanto o segundo é o termo de regularização (*weight decay*), que tende a decrementar a magnitude dos pesos e ajuda a prevenir *overfitting*. O parâmetro de decaimento (*weight decay parameter*) λ controla a importância relativa dos dois termos. O objetivo é minimizar $J(W, b)$ como função de W e b. Para treinar a rede neural, inicializa-se cada parâmetro $W_{ij}^{(l)}$ e $b_i^{(l)}$ com valores aleatórios próximos de zero, e em seguida aplicando um algoritmo de otimização, como o Gradiente Descendente.

Uma iteração do Gradiente Descendente atualiza os parâmetros W, b como segue:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(W, b),$$

onde α é a taxa de aprendizado.

Dado um conjunto de treinamento (x, y), é necessário calcular todas as ativações ao longo da rede, inclusive o valor da hipótese $h_{W,b}(x)$. Depois, para cada nó i na camada l, deve-se calcular $\delta_i^{(l)}$, que mede a relevância do nó nos erros da camada de saída. Para cada nó de saída, $\delta_i^{(n_l)}$, onde n_l é a camada de saída, pode-se calcular diretamente a diferença entre a ativação da rede e o valor esperado. Para as camadas internas, δ_i^l é calculado com base na média dos erros dos nós que utilizam a_i^l como saída. O algoritmo de *Backpropagation* é apresentado a seguir:

(1) Calcular as ativações para cada camada da rede, inclusive a camada de saída L_{n_l}

(2) Para cada unidade de saída i na camada n_l, atribua

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{n_l}} \frac{1}{2} \|y - h_{W,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

(3) Para $l = n_l - 1, n_l - 2, \ldots, 2$

Para cada nó i na camada l, atribua

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_l - 1} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

(4) Calcule as derivadas parciais desejadas como segue:

$$\frac{\delta}{\delta W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\delta}{\delta b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}.$$

Para treinar a rede neural, pode-se repetir passos do Gradiente Descendente para reduzir a função de custo $J(W, b)$ (Ng, 2011).

Mikolov et al. (2011) propõe o uso de *Backpropagation Through Time* (BPTT) para o treinamento de redes neurais recorrentes (RNNLM). Com *truncated BPTT*, o erro é propagado através das conexões recorrentes para trás no tempo considerando um número específico de passos. Assim, a rede aprende a lembrar informações de vários passos da camada interna. Este algoritmo é descrito com mais detalhes em Werbos (1990).

4.1.6 Trabalhos Relacionados

Representação de palavras como vetores de valor real, ou *word embeddings*, tem sido estudada há bastante tempo e tem sido aplicada em muitos problemas dentro da área de Inteligência Artificial, como análise de sentimento (Maas et al., 2011) e detecção de paráfrase (Socher et al., 2011). Ao longo dos anos, muitos cientistas da computação têm proposto diferentes modelos arquiteturais para cálculo e aprendizado de tais representações. A seguir, faremos uma breve descrição dos principais trabalhos presentes na literatura.

Em Hinton (1984) é apresentado o conceito de representações distribuídas. Essa nova forma de representação é uma alternativa às representações locais, que consistem no uso de um elemento computacional para cada entidade a ser representada. Nesta abordagem é proposto um modelo em que a informação de uma entidade a ser representada envolve características distribuídas entre várias outras entidades computacionais envolvidas. Esta forma de representação foi amplamente utilizada em trabalhos posteriores.

Schütze (1993) apresenta o conceito de espaço de palavras, ou *word space*. Nesse trabalho é proposta uma metodologia, baseada em *n-gram*, para computação de representações semânticas distribuídas de palavras com base em estatísticas de coocorrências lexicais dentro de um corpo textual. Para cada palavra são considerados os contextos em que esta ocorre no texto, e a similaridade semântica é calculada através da proximidade dos vetores no espaço. Testes foram realizados em um corpus contendo 50000 (cinquenta mil) palavras. Os resultados obtidos demonstraram que palavras semanticamente relacionadas estavam próximas no espaço e que representações vetoriais podem ser utilizadas para desambiguação.

Em Bengio et al. (2003) é proposto um modelo probabilístico para computação e aprendizado de representações distribuídas de palavras. Nesse trabalho, os autores propõem um modelo neural que, dado um vocabulário V, mapeia cada palavra em um vetor real e, em seguida, computa, através de uma função, a probabilidade condicional das palavras em V, isto é, a probabilidade de ocorrência de uma palavra

em uma sequência. Desta forma, a rede neural aprende, de maneira conjunta, os vetores de características e um modelo estatístico. Os resultados obtidos mostraram-se melhores em comparação àqueles obtidos por abordagens baseadas em *n-grams*, obtendo melhor generalização. Contudo, este modelo mostrou-se mais lento para treinar e testar.

Posteriormente, Morin and Bengio (2005) propôs uma variante mais rápida do modelo acima descrito. É proposta a construção de uma árvore binária que corresponde à hierarquia das palavras em um vocabulário, onde cada palavra equivale a uma folha desta árvore e é especificada pelo caminho da raiz até esta palavra. Caso a árvore esteja balanceada, cada palavra pode ser descrita como uma sequência de decisões estocásticas binárias de ordem $log|V|$, onde V é o vocabulário.

Posteriormente, essa abordagem foi aprimorada em Mnih and Hinton (2009), em que apresentou performance superior a de modelos neurais não-hierárquicos e baseados em *n-grams*. Neste trabalho, o modelo hierárquico empregado difere do apresentado acima pelo uso de uma versão modificada do modelo log-bilinear apresentado em Mnih and Hinton (2007) para a computação das probabilidades de cada nó e pela habilidade em lidar com múltiplas coocorrências de cada palavra na árvore.

Em Huang et al. (2012) é proposto um modelo arquitetural de aprendizado em que são consideradas a polissemia e a homonímia de palavras, e que aprende representações embasadas não apenas no contexto local onde cada palavra aparece, mas também no contexto global do documento. Essa abordagem tem como objetivo resolver a limitação dos modelos que utilizam uma única representação vetorial para uma palavra, que, por conseguinte, acabam não aprendendo os diferentes significados que a mesma pode ter. Para tanto, utiliza-se um modelo multi-protótipo, que utiliza uma variedade de representações para uma mesma palavra. Experimentos realizados para este modelo mostraram desempenho superior a de modelos neurais posteriores.

Mais recentemente, Mikolov et al. (2013a) propôs dois modelos para computação de representações vetoriais contínuas de palavras a partir de um conjunto de dados de grande escala. O primeiro modelo proposto é o saco-de-palavras contí-

nuo, ou *continuous bag-of-words*, o qual prevê a palavra atual baseado no contexto. Esse modelo é assim chamado uma vez que a ordem das palavras no histórico não influencia a projeção, e diferente do modelo saco-de-palavras clássico, este utiliza representações distribuídas contínuas do contexto.

Outro modelo proposto é o *continuous skip-gram*. Similar ao saco-de-palavras contínuo, esse modelo, porém, ao invés de predizer a palavra baseada no contexto, tenta maximizar a classificação de uma palavra baseada em outra palavra na mesma sentença. Mais precisamente, para cada palavra atual, este modelo propõe-se a predizer palavras que estejam a uma certa distância em torno desta palavra. Segundo os autores, estes modelos mostraram-se capazes de treinar com corpora de ordens de magnitude muito superiores aos dos melhores modelos previamente propostos. Além disso, acredita-se que problemas como análise de sentimento e detecção de paráfrase podem ser beneficiados por estes modelos.

Com base nesses trabalhos, nota-se que *word embeddings* é uma metologia com ótimos resultados e, por conseguinte, bastante promissora.

4.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNN) são variantes dos *perceptrons* de múltiplas camadas baseadas no funcionamento do córtex visual dos animais. Criado inicialmente para problemas de visão computacional, esse modelo tem-se mostrado bastante eficiente para problemas de processamento de linguagem natural (Kim, 2014)

Segundo Johnson and Zhang (2015), "uma *CNN* é uma rede neural *feedforward* equipada com camadas de convolução intercaladas por camadas de *pooling*". As camadas de convolução consistem em unidades computacionais que representam, cada uma delas, uma região da entrada, ou *vetor de região*, como, por exemplo, uma parte de uma imagem. Cada uma dessas unidades é disposta de maneira a cobrir toda a imagem. E sobre suas entradas é aplicada uma função não-linear. Essa camada tem como objetivo identificar características da imagem, como bordas ou

curvas, por exemplo; e a rede tem como objetivo aprender filtros de convolução que são ativados sempre que encontram uma característica.

Johnson and Zhang (2014) explicam que, dada uma entrada x, uma unidade associada a ℓ-ésima região computa

$$\sigma(\mathrm{W} \cdot \mathrm{r}_\ell(\mathrm{x}) + \mathrm{b}),$$

onde $\mathrm{r}_\ell(\mathrm{x})$ é o vetor de região representando a região x na localização ℓ, e σ é uma função de ativação não-linear predefinida aplicada em cada componente. W é uma matriz de pesos, enquanto b, um vetor de vieses. Ambos são aprendidos durante o treino e são compartilhados com todas as unidades computacionais na mesma camada. Dessa forma, cada unidade gera um vetor de dimensão m onde m é o número de vetores de peso, isto é, o número de linhas de W ou neurônios.

Os vetores de dimensões m são então agregados pela camada de *pooling* e em seguida, usadas pela camada superior (um classificador linear) como *features* para classificação. Johnson and Zhang (2014) afirmam que a camada de *pooling* "essencialmente contrai a imagem mesclando *pixels* vizinhos, para que camadas superiores possam lidar com informações mais abstratas/globais".

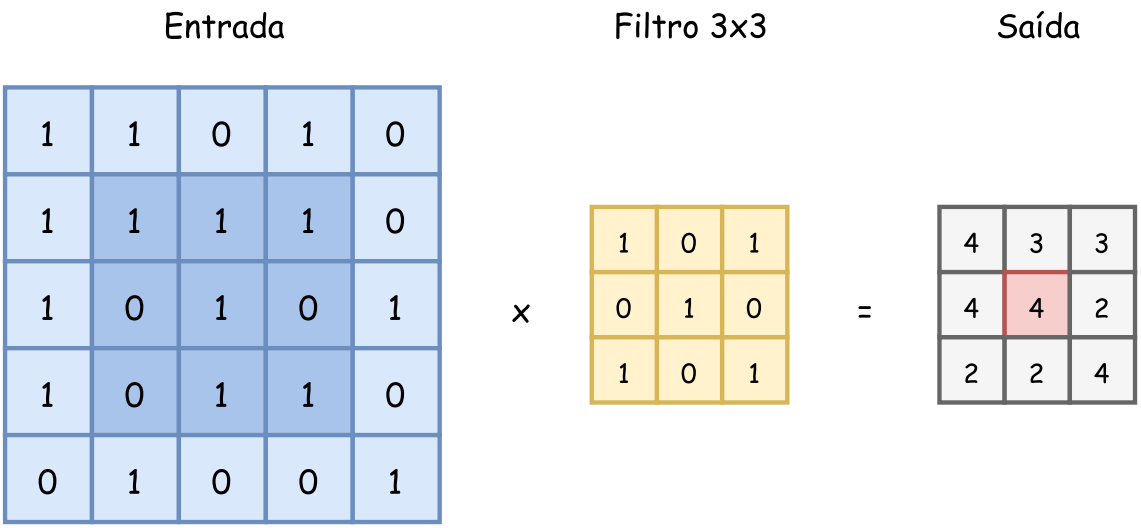

Figura 4.5: Exemplo de convolução com imagem 5×5 e filtro 3×3.

A Figura 4.5 mostra um exemplo de convolução. Neste exemplo, consideramos uma matriz de dimensões 5×5 representando uma imagem cujos *pixels* assumem

valor 0 ou 1 e um filtro (ou *kernel*) 3×3. Para cada região da entrada, é feita uma multiplicação, elemento a elemento, entre a região e o filtro, e os resultados obtidos são somados, gerando uma nova matriz de tamanho 3×3. A esta matriz dá-se o nome de *mapa de features*.

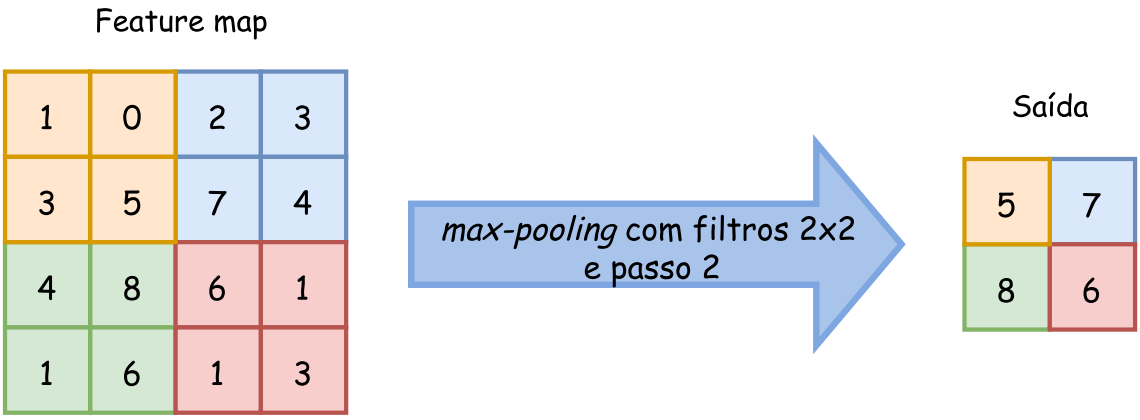

Figura 4.6: Exemplo de *pooling* com filtros 2×2.

A Figura 4.6 mostra um exemplo de *pooling* aplicado a um mapa de *features*. Neste exemplo, tomamos uma vizinhança espacial (ou janela) de tamanho 2×2. Nela, aplicamos *pooling* máximo, isto é, tomamos o maior valor dentro desse espaço. Essa janela percorre todo o mapa a cada duas células, formando uma nova matriz 2×2. Dessa forma, reduz-se a dimensionalidade do mapa, mas mantendo a informação mais relevante.

Johnson and Zhang (2014) explica que, quando aplicado a textos, uma rede CNN requer uma representação dos dados que preserve localizações internas (neste caso, a ordem das palavras) da entrada. Desta forma, dado um documento $D = (w_1, w_2, \dots)$ com vocabulário V, uma possível representação seria obtida considerando cada palavra como um *pixel*, considerando D como uma imagem com $|D| \times 1$ *pixels* e $|V|$ canais, e representando cada palavra, ou *pixel*, como um vetor de dimensão $|V|$. Por conseguinte, $\mathrm{r}_\ell(\mathrm{x})$ pode ser a concatenação de vetores, um vetor *bag of words*, ou um vetor *bag of* n-*grams*.

Enquanto aplicações com imagens utilizam tamanho fixo, documentos possuem tamanho variado. E devido a isso, é necessário reajustar a camada de *pooling* para suprir esse problema. Uma alternativa é o uso de *pooling* máximo (*max-pooling*)

sobre todo o conjunto de dados, ou seja, uma unidade associada a todo o texto. Outra proposta, é a aplicação de *pooling* k-máximo (*k-max*) dinâmico, que consiste em tomar os k valores mais altos, onde k é função do tamanho da sentença, mas novamente aplicado a todo o conjunto de dados, e a operação é aplicada a *pooling* máximo (Johnson and Zhang, 2014). A Figura 4.7 mostra a arquitetura de uma rede convolucional.

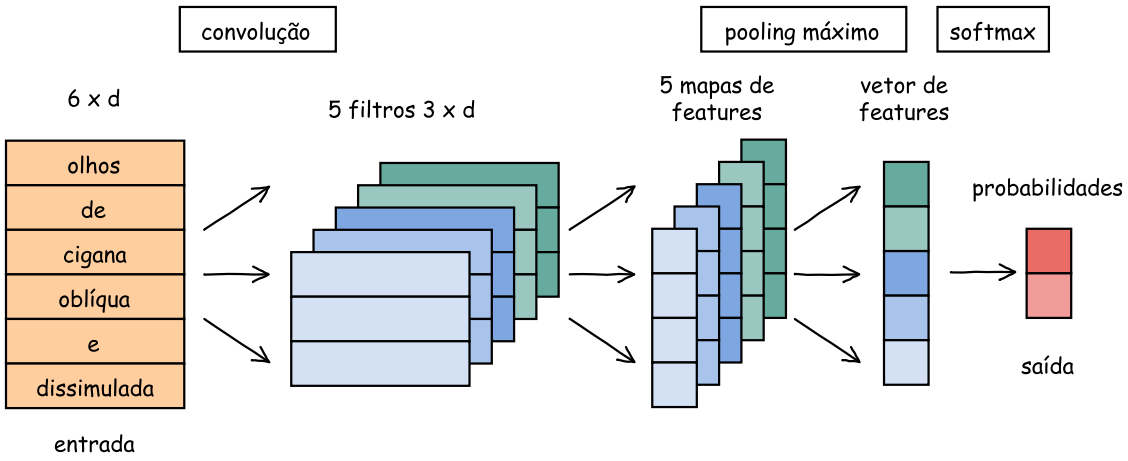

Figura 4.7: Exemplo de arquitetura de uma CNN para classificação de sentença.

Em 4.7 a entrada consiste em uma matriz formada pela concatenação de vetores de palavras de dimensão $1 \times d$. Essa entrada é escaneada por filtros de convolução de dimensão $3 \times d$. Esses filtros percorrem a imagem com *stride* 1, gerando mapas de *features* de dimensão 1×4 que são agregados e dados como entrada para a camada de *pooling*. Essa camada executa sobre cada mapa de *features* um *pooling* máximo, extraindo o valor máximo resultante da aplicação de cada filtro de convolução, gerando um novo vetor de dimensões 1×5. Esse vetor é passado para a camada de saída, que aplica *softmax* e calcula as probabilidades para cada classe.

4.2.1 Trabalhos Relacionados

Esta seção destina-se a uma revisão literária dos principais trabalhos realizados com redes neurais convolucionais aplicadas a tarefas de processamento de linguagem natural.

Kalchbrenner et al. (2014) propõem uma rede convolucional para modelagem semântica de sentenças que utiliza *pooling* k-máximo dinâmico. A rede manuseia sentenças de entrada de tamanho variado e induz um grafo de características sobre a sentença capaz de captar de forma explícita relações entre palavras a curta e longa distância. Os experimentos demonstraram que esse modelo tem alto desempenho nos problemas de classificação de sentimento e *six-way question classification* sem necessidade de *features* externas. Em relação à predição de sentimento utilizando o Twitter, obteve-se uma redução de erro 25% em relação a outros trabalhos.

Em Kim (2014) são apresentados experimentos com convolucionais treinadas com vetores de palavras pré-treinadas a partir de aprendizado não-supervisionado para classificação de sentenças. Nos experimentos realizados, quatro variações do modelo foram utilizadas. O primeiro deles consiste na inicialização aleatória das palavras, modificadas ao longo do treinamento. O segundo utiliza os vetores pré-treinados com *word2vec*. Nesse, todas as palavras são mantidas estáticas e somente os demais parâmetros do modelo são aprendidos. O terceiro modelo é o mesmo que o segundo, com o diferencial de que os vetores são refinados ao longo do treinamento através de *backpropagation*. O quarto e último modelo consiste na utilização de dois conjuntos de vetores de palavras. Cada conjunto é tratado como um "canal", e os filtros são aplicados a ambos. Um dos canais é mantido estático, ao passo que o outro é refinado ao longo do treinamento. Esse trabalho obteve bons resultados e contribuiu para evidenciar a qualidade já esperada dos vetores de palavras em tarefas de processamento de linguagem natural.

Em Johnson and Zhang (2014), os autores fazem uma análise da aplicação de redes neurais convolucionais no problema de classificação de textos, explorando a ordem das palavras. As redes convolucionais são aplicadas diretamente sobre dados textuais de alta dimensão, em oposição à abordagem tradicional, que utiliza vetores de baixa dimensão. São estudados dois tipos de *CNN*: uma adaptação direta de convolucionais utilizadas para imagens, e uma variante que emprega conversão para *bag-of-words* na camada de convolução. Na primeira abordagem, o documento de entrada é tratado como uma imagem onde cada palavra é um pixel, representado por um vetor binário cuja dimensão é o tamanho do vocabulário e o valor 1 é atribuído

apenas à posição dessa palavra no vocabulário. Deste modo, uma região de tamanho fixo do texto é representada pela concatenação dos vetores das palavras dessa região. A segunda abordagem propõe a utilização de vetores *bag-of-words* para representar a região da entrada. Neste caso, cada região é um vetor binário onde o valor 1 é atribuído à posição corresponde a cada uma das palavras presentes na sequência. Os experimentos realizados corroboraram a eficiência da metodologia em comparação a métodos do estado-da-arte e trabalhos posteriores.

Em Johnson and Zhang (2015), os autores dão continuidade ao trabalho descrito acima. Nele é proposto um *framework* de aprendizado semi-supervisionado que aprende representações de regiões de texto a partir de dados não rotulados e posteriormente integra as representações aprendidas em um treino supervisionado. Aqui, uma rede neural é treinada para prever o contexto das regiões de modo que sua camada de convolução gere vetores de características para cada região de texto, que posteriormente são usadas como entrada adicional de uma rede convolucional com aprendizado supervisionado. Essa abordagem apresentou vantagens em relação à anterior, e usando esses novos modelos, eles conseguiram uma melhor performance em relação a estudos anteriores em classificação de sentimento e classificação de tópico.

Em Zhang and Wallace (2015) é proposta uma análise de redes convolucionais de uma camada para explorar a influência das configurações do modelo em sua performance, identificando empiricamente os parâmetros cujo refinamento é necessário e aqueles menos importantes para a performance. Nesse trabalho, os parâmetros analisados foram: os vetores de palavras da entrada; o tamanho do filtro de região; o número de mapas de características; a função de ativação; a estratégia de *pooling* e a regularização. Os resultados dos experimentos revelaram que a representação das palavras impacta na performance; contudo, cada uma delas provê resultados melhores em tarefas distintas. Observou-se também que o tamanho do filtro de região e o número de mapas de características têm grande influência nos resultados. *Pooling* 1-máximo tem desempenho superior ao de outras estratégias. E a regularização tem efeito relativamente pequeno na performance do modelo.

Diante desse passeio pelos principais trabalhos presentes na literatura, pode-se ver nitidamente que as redes neurais convolucionais, embora inicialmente voltadas para problemas de visão de computacional, são modelos bastante promissores e com ótimo desempenho nas tarefas de processamento de linguagem natural.

A seguir, apresentaremos a metodologia aplicada neste trabalho e os resultados dos experimentos realizados.

Capítulo 5

Experimentos

Neste capítulo, serão apresentadas a metodologia aplicada durante os experimentos e métricas para validação, o conjunto de dados utilizado e a análise dos resultados obtidos.

5.1 A Base de Dados

O conjunto de dados aqui utilizado foi concedido pela empresa Buscapé e é composto por 11240063 (onze milhões, duzentos e quarenta mil e sessenta e três) ofertas distribuídas entre 25 categorias-pai distintas. A Figura 5.1 apresenta, através de um gráfico de barras, a distribuição das ofertas ao longo dessas categorias.

Nota-se, por meio do gráfico, uma variância significante na distribuição das ofertas entre as categorias-pai. É notável uma dominância de ofertas da categoria "Moda e Acessórios", e as categorias "Música Digital", "Tabacaria", "Arte e Antiguidade" e "Artigos Religiosos" apresentam quantidades pouco significativas de dados comparadas às demais. Como consequência, é possível que o classificador tenha menor desempenho durante a classificação de ofertas das categorias "Música Digital", "Tabacaria", "Arte e Antiguidade" e "Artigos Religiosos". Da mesma forma, algumas ofertas podem ser erroneamente classificadas, sendo atribuídas à categoria "Moda e Acessórios".

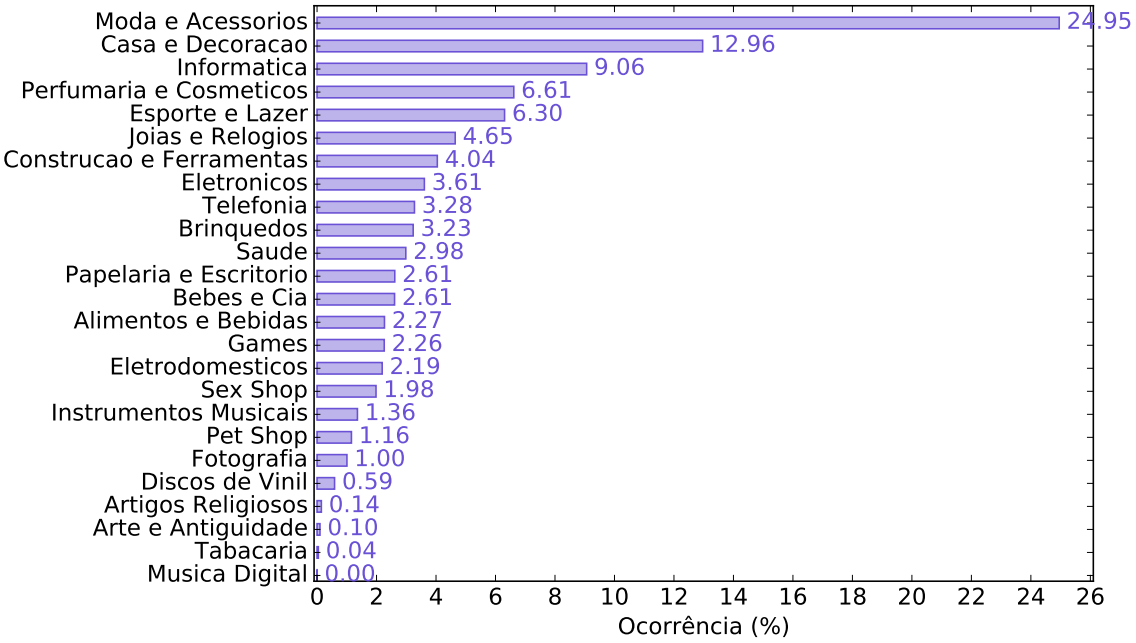

Figura 5.1: Distribuição das categorias-pai ao longo do *dataset*.

Observando o conjunto de dados, nota-se também uma grande variação no tamanho das ofertas, isto é, o número de palavras dos textos das descrições. Essa observação é importante para determinar o quanto esse aspecto pode influenciar no desempenho do algoritmo. A Figura 5.2 mostra, através de um histograma, a frequência absoluta dos tamanhos das ofertas do longo do *dataset*.

Através de uma análise do histograma e dos valores nele apresentados, descobriu-se que as ofertas possuem em média 8 palavras, aproximadamente, com valores variando de 1 a 61 e desvio padrão de aproximadamente 4 palavras. A moda encontrada foi de 6 palavras. Ofertas com muitas palavras, em geral, possuem uma quantidade grande de informações pouco relevantes para o algoritmo, apresentando *tokens* numéricos e códigos que não adicionam semântica à descrição da oferta. Além disso, a presença desses *tokens* levará potencialmente à geração de um vocabulário bastante extenso, o que pode desencadear um consumo desnecessário de memória e tempo computacional durante a execução dos algoritmos utilizados nos experimentos. Diante desse problema, optou-se aqui por substituir os *tokens* numéricos pelo termo "numero", diminuindo assim o vocabulário.

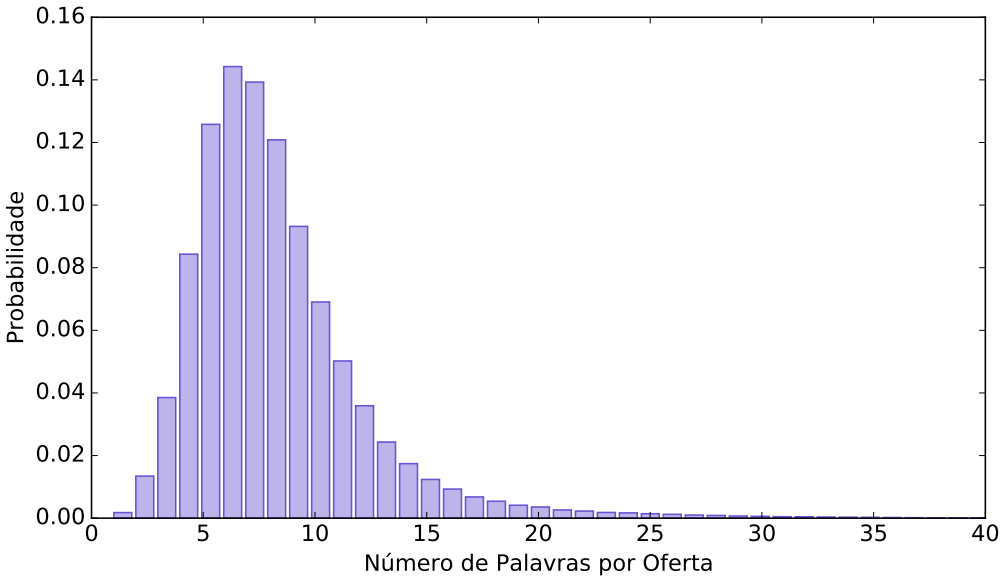

Figura 5.2: Distribuição dos tamanhos das descrições das ofertas ao longo do *dataset*. As frequências estão normalizadas no intervalo $[0, 1]$.

5.2 Metodologia

A fim de atestar a qualidade das metodologias *word embedding*, foi realizada uma bateria de experimentos de modo a comparar o desempenho da aplicação dessa técnica em comparação ao uso do classificador BOW. Foram consideradas diferentes circunstâncias e parâmetros. A metodologia utilizada é apresentada abaixo.

5.2.1 O *framework* Gensim

Para a geração dos vetores *word embedding*, optou-se aqui pela utilização do *framework* Gensim (Řehůřek and Sojka, 2010), disponível para a linguagem Python. Esse *framework*, uma abreviação de *generate similar*, do inglês, "gerar semelhante", apresenta uma coleção de *scripts* para modelagem, indexação de documentos e recuperação de similaridade, dentre eles uma versão orientada a objetos da ferramenta *word2vec* desenvolvida por Mikolov e outros (Goldberg and Levy, 2014).

A partir de um conjunto de documentos preprocessados, essa ferramenta gera, através de rede neural, modelos vetoriais multidimensionais. Por padrão, o algoritmo

de treino utilizado é o CBOW.

5.2.2 Validação dos Modelos

A fim de realizar os experimentos, optou-se nesse trabalho pela geração de três subconjuntos de dados a partir do *dataset*. Inicialmente, o conjunto foi subdivido em dois conjuntos, um contendo 90% das ofertas, e o outro, 10%. Esse último servirá como conjunto de teste. Essa divisão foi feita de forma estratificada, preservando a porcentagem de amostras para cada classe. Após isso, aplicou-se uma nova divisão, dessa vez sobre o conjunto contendo 90% da amostra inicial, de modo que os dois novos subconjuntos possuam 80% e 20% das ofertas deste. O maior servirá como conjunto de treino, enquanto o menor, de validação.

Cada experimento descrito a seguir foi realizado uma única vez sobre os conjuntos de treino e validação. Ao final de todos os experimentos, a melhor configuração encontrada para cada modelo foi enfim testada sobre o conjunto de teste.

5.2.3 Métricas

As métricas adotadas nesse trabalho foram acurácia e macro f1. A acurácia é calculada simplesmente utilizando a porcentagem de ofertas corretamente classificadas, ou seja, cuja categoria indicada pelo algoritmo seja igual ao rótulo pré-definido.

Optou-se também pela utilização do macro f1 como métrica de qualidade. O cálculo consiste na média harmônica do f1 por classe. Essa métrica é importante pois nesse problema busca-se maximilizar a qualidade de toda a classificação e não apenas de cada categoria.

5.2.4 UWE: Análise dimensional

Para analisar o impacto do tamanho dos vetores *word embedding* na qualidade do classificador UWE, realizamos uma sequência de experimentos variando o tamanho do espaço vetorial e o número de palavras utilizadas. Nesses experimentos, a dimensionalidade do espaço vetorial variou entre 40 e 400 dimensões, com incremento de

40 dimensões por execução.

Além disso, optou-se pela utilização das 3, 6 e 9 primeiras palavras da descrição das ofertas para cada uma das dimensões acima citadas. Cada oferta é então representada pela concatenação dos *embeddings* das palavras utilizadas. No caso de o número de palavras da descrição for inferior ao valor requerido, um *padding* é adicionado ao final do vetor representativo da oferta até que seu tamanho se ajuste ao desejado. Nesse caso, o *padding* é um vetor de valores iguais a zero.

Ao final, executamos outros três experimentos com as 3, 6 e 9 primeiras palavras com o classificador BOW. Separamos os melhores resultados para cada número de palavras utilizadas no classificador UWE e comparamos com os resultados do classificador BOW. Dessa forma, foram realizados 33 experimentos nessa etapa de testes.

5.2.5 BOW: Análise da engenharia de *features*

A fim de analisar a engenharia de *features* apresentada no capítulo 3, foi realizada uma série de testes com cada uma delas individualmente e um último experimento com todas.

As *features* analisadas foram: primeira palavra, bigramas, categoria na loja, categoria por preço e nome da loja. Para cada uma delas, foi realizado um único experimento utilizando os conjuntos de treino e validação. Realizamos também um experimento utilizando apenas a descrição da oferta e outro utilizando todas as *features* combinadas. Em todos eles, foram utilizadas todas as palavras da descrição da oferta. No total, foram realizados sete experimentos nessa etapa.

5.2.6 BOW: Análise da confiança

Nessa etapa, é feita uma análise da confiança do classificador BOW. Essa análise é necessária para se verificar a porcentagem das ofertas classificadas que será submetida diretamente ao SCP e o quanto será mantido para validação, que é realizada por uma equipe especializada.

Dessa forma, foi utilizado um *threshold* que determina o limite inferior da probabilidade de cada oferta pertencer de fato à categoria indicada pelo algoritmo. Em outras palavras, para o cálculo da acurácia e da porcentagem de exemplos retornados, consideramos apenas as ofertas cuja probablidade seja igual ou superior a esse valor.

Durante os experimentos, o *threshold* variou entre 10% e 90%, com incremento de 10% a cada execução. Nessa etapa, foi utilizado o classificador BOW+ftrs (i.e., que utiliza todas as *features*). Assim, realizamos aqui uma sequência de dez experimentos.

5.2.7 Parâmetros dos classificadores

Para os experimentos realizados, optamos inicialmente pela utilização do classificador com Gradiente Descendente Estocástico *SGDClassifier*, disponível na biblioteca *Scikit Learning* da linguagem de programação Python. No caso da função de custo, foi utilizado o algoritmo *Modified Huber Loss*.

Para o classificador BOW, decidimos utilizar uma matriz esparsa binária $N \times V$, onde N é o número de ofertas do conjunto de dados utilizado, e V, o tamanho do vocabulário gerado. Para cada palavra do vocabulário, atribuiu-se 1 à sua respectiva posição na matriz caso essa palavra esteja presente na descrição da oferta, e 0, caso contrário.

Para o classificador UWE, utilizamos uma matriz real $N \times D$. Cada linha da matriz é um vetor de dimensão d, gerado a partir da concatenação dos vetores de cada palavra que descreve uma oferta. O valor D é então calculado com base no tamanho global do conjunto de dados, ou seja, o tamanho da maior oferta encontrada. Dessa forma, a dimensão de uma oferta é dada por

$$D = d \times t_{global}.$$

Os identificadores das categorias-pai foram mapeados para valores numéricos inteiros entre 0 e $N - 1$, onde N é o número de categorias-pai distintas presentes no *dataset*.

Para a CNN, tomamos como parâmetro 2 épocas e taxa de aprendizado de 0,05. A janela aqui adotada foi de 5 palavras. Foram utilizados 100 filtros de convolução. O tamanho do *embedding* para as palavras da descrição da oferta foi de 100, ao passo que, para as *features*, o tamanho escolhido foi de 50. Para essa abordagem, foram usados *embeddings* aprendidos de forma não-supervisionada e *embeddings* aprendidos ao longo do treinamento do modelo supervisionado.

5.3 Resultados

Apresentamos aqui os resultados dos experimentos realizados segundo a metodologia apresentada na seção 5.2.

5.3.1 Análise da dimensionalidade

Aqui, analisamos o impacto da dimensionalidade do espaço vetorial no desempenho do classificador UWE. Para uma melhor visualização, extraímos os melhores resultados dos experimentos realizados, que podem ser vistos na Figura 5.3. É notável o crescimento da acurácia à medida em que o número de dimensões também aumenta, o que já era esperado uma vez que, quanto maior a dimensionalidade do espaço vetorial, mais características são aprendidas pelos *embeddings*. A acurácia cresce rapidamente entre 40 e 120 dimensões. E a partir desse ponto, apresenta um crescimento mais lento. Pode-se inferir então que a dimensão dos vetores influencia consideravelmente na qualidade da classificação. Contudo, esse impacto se torna menos perceptível entre espaços vetoriais com grandes dimensões.

A seguir, realizamos uma comparação entre os resultados obtidos pelo classificador UWE e o classificador BOW. Os resultados podem ser visualizados através da Figura 5.4. Pelo gráfico, vemos que o BOW teve resultados um pouco melhores que o UWE, com diferença de 1,88% na acurácia com as 9-primeiras palavras. Pode-se notar também que a diferença entre os dois modelos aumentou ao longo do eixo das abscissas. Não podemos afirmar de maneira exata o porquê desse comportamento, contudo, uma possível explicação pode estar na representação vetorial das ofertas.

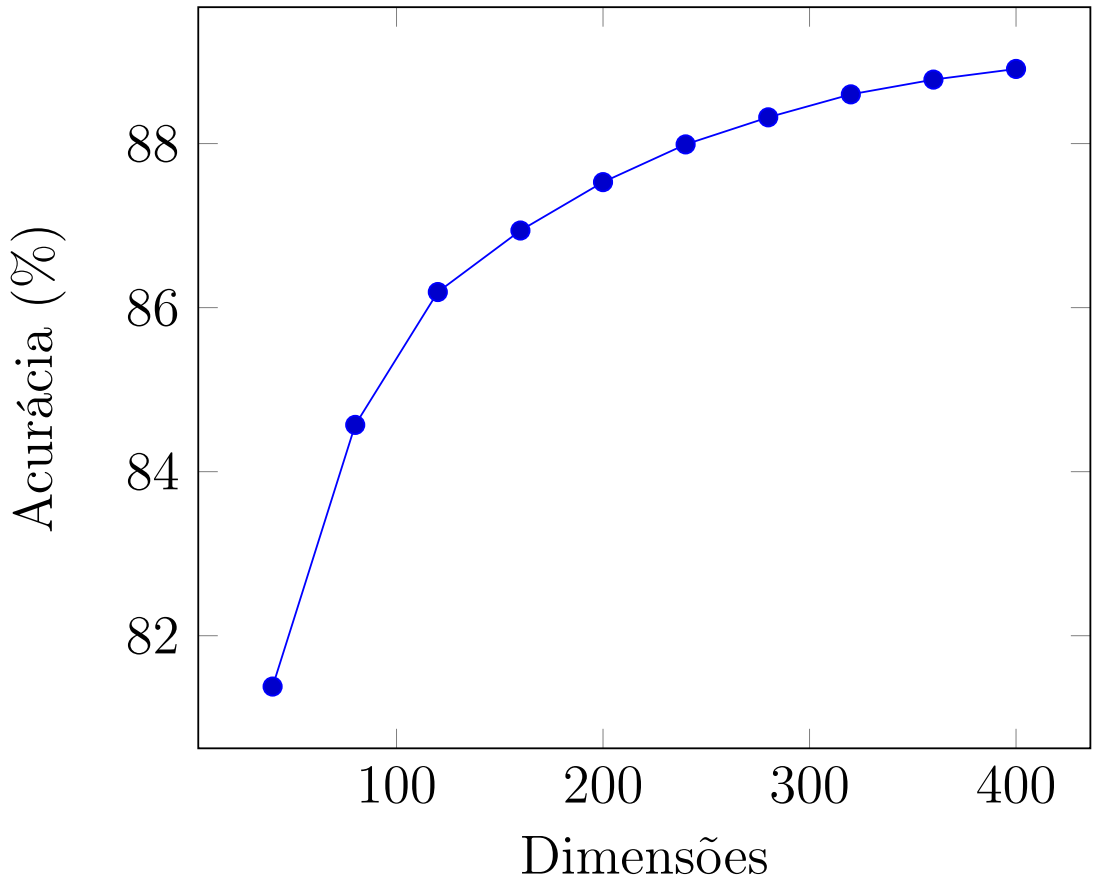

Figura 5.3: Impacto da dimensionalidade no classificador *word embedding*.

Conforme apresentado na seção 5.1, a maior parcela do conjunto de dados é composta por ofertas com 6 palavras. Além disso, como dito em 5.2.7, os vetores das ofertas são ajustados ao tamanho global. Dessa forma, a conversão de ofertas em vetores reais pode ter resultado em uma quantidade significante de vetores esparsos, isto é, com uma grande quantidade de zeros ao longo de um número grande de dimensões, na casa de milhares.

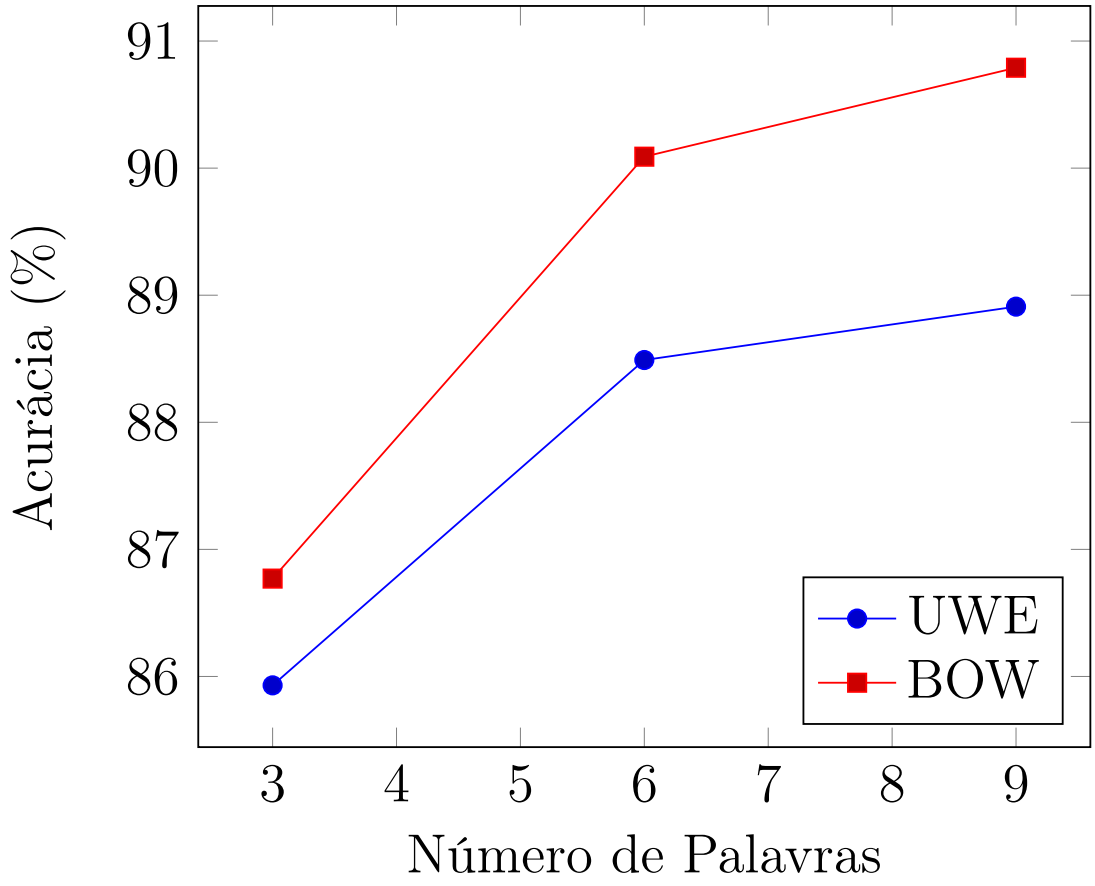

Figura 5.4: Comparação entre os desempenhos dos classificadores com *bag of words* e *word embedding*

5.3.2 Análise da engenharia de *features*

A Figura 5.5 apresenta um gráfico de barras com os valores da acurácia em relação às *features* utilizadas com o algoritmo *bag of words* aplicado sobre o conjunto de validação. Podemos notar que o nome da loja foi a *feature* que mais impactou positivamente na qualidade do classificador, com aumentos de 2,54% e 4,63% na

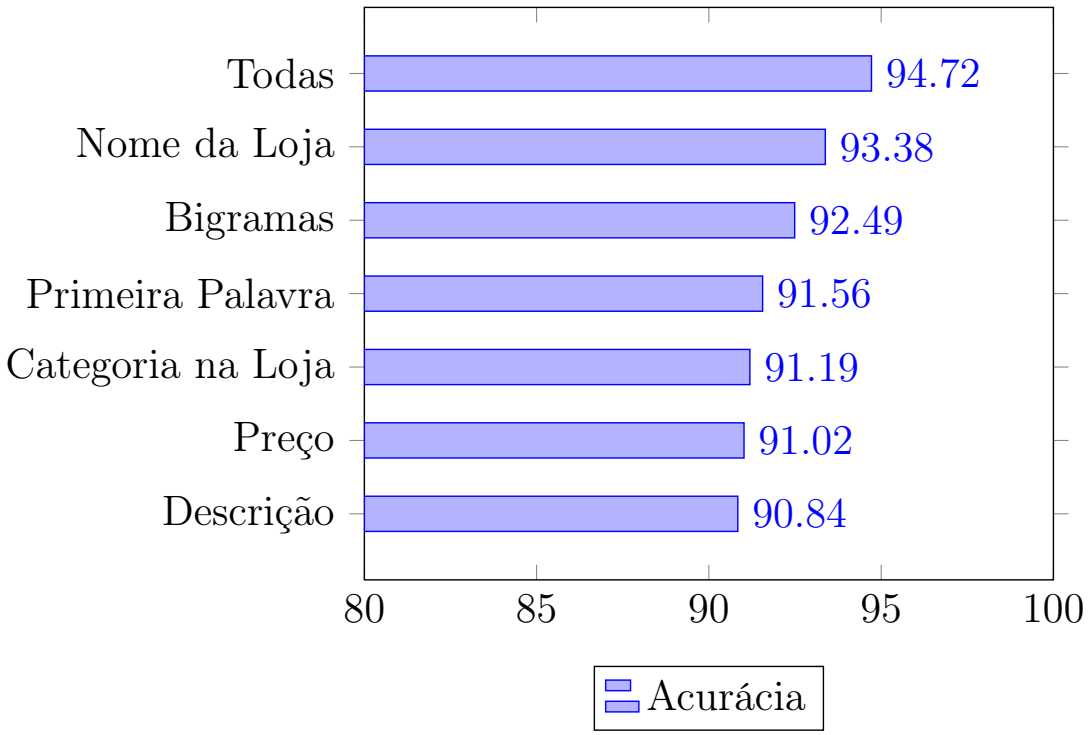

Figura 5.5: Impacto da engenharia de features no desempenho do classificador BOW.

acurácia e no macro f1, respectivamente, em relação à versão que utiliza apenas o texto da descrição da oferta. Em contrapartida, o preço mostrou-se pouco relevante, melhorando em apenas 0,18% a acurácia, e em 0,33% o macro f1. O baixo impacto desta *feature* na classificação por categorias-pai pode ser explicado pela grande disparidade nos preços dos produtos das subcategorias por elas agrupadas. O uso de bigramas aumentou em 1,65% a acurácia e em 1,23%, o macro f1, sendo então a segunda *feature* com melhor desempenho. Isso mostra que o contexto em que as palavras aparecem na oferta é um melhor parâmetro para classificação do que apenas as suas ocorrências. O uso da primeira palavra aumentou em 0,72% e 1,22% a acurácia e o macro f1, respectivamente. E por fim, a categoria na loja aumentou a acurácia em 0,35% e em 0,55% o macro f1.

5.3.3 Limiar de confiança do classificador com *bag of words*

Aqui, analisamos a confiança do classificador com todas as *features*. Essa análise é realizada levando-se em consideração a probabilidade de uma oferta ser de fato da classe indicada pelo algoritmo. Assim, estabelecemos limites inferiores de modo que

o cálculo da acurácia seja feito sobre os exemplos cuja probabilidade seja maior ou igual a esses limites. Dessa forma, temos a Figura 5.6. Através do gráfico, nota-se que a acurácia praticamente constantes até o limite inferior de 20%. Nessa Figura temos também o gráfico gerado a partir da porcentagem de exemplos retornados. Nota-se aqui um decaimento, já esperado, do número de exemplos obtidos à medida em que o limite inferior aumenta. Apesar disso, 64% dos valores preditos pelo algoritmo tem 90% ou mais de probabilidade de estarem corretos, o que corrobora a qualidade do algoritmo. Percebe-se também que, de maneria análoga à acurácia e ao macro f1 acima citados, a curva mantém-se praticamente constante até o limite inferior de 20%.

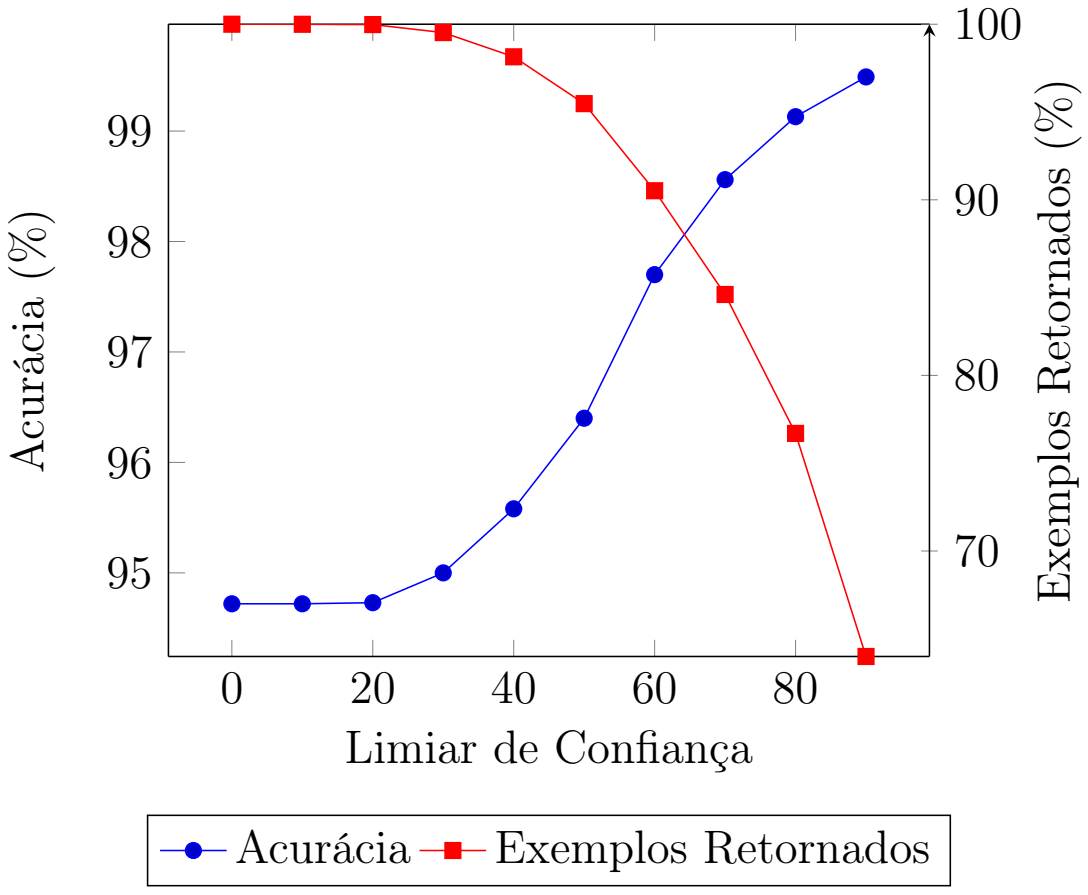

Figura 5.6: Porcentagem de exemplos retornados e curva da acurácia com diferentes valores do limiar de confiança.

5.3.4 Resultados do Classificador CNN

A Figura 5.7 apresenta a comparação entre os diferentes modelos por nós utilizados. Podemos notar que o modelo *CNN* é bastante superior aos modelos *BOW* e *UWE*, obtendo, com sua variante mais simples (sem *features*), um acréscimo de 0,79% na acurácia em relação ao modelo *BOW* com todas as *features*. Em contrapartida, a aplicação de *word embedding* não-supervisionado ao modelo *CNN* simples não demonstrou melhora significativa em relação ao modelo *BOW* simples (acréscimo de 0,25% na acurácia). Também podemos perceber que a adição de *features* (nesse caso, *nome da loja* e *categoria na loja*) ao modelo *CNN* não resultou em um ganho relevante na acurácia como ocorrido com o *BOW*. Uma possível explicação pode estar no fato de que modelos *CNN* já captarem aspectos posicionais e de vizinhança das palavras. Surpreendentemente, o modelo *UWE* teve desempenho inferior à todos os outros, inclusive o modelo *BOW* mais simples.

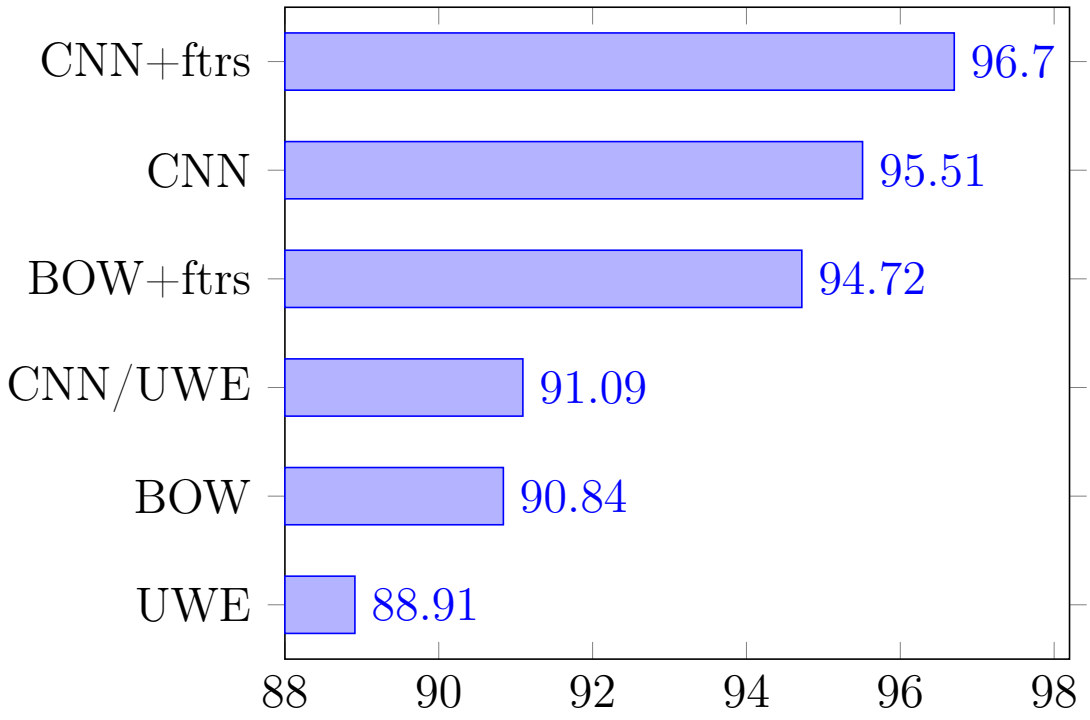

Figura 5.7: Comparação entre a acurácia dos diferentes modelos.

Na Figura 5.8 apresentamos a métrica F1 alcançada pelo modelo CNN+ftrs para cada categoria (ordenadas pelos valores F1). Comparando esses resultados com a frequência das categorias da Figura 5.1, podemos observar que a performance é

geralmente inferior para categorias menos frequentes, o que é o esperado. Mais importante, a única categoria com F1 abaixo de 80% é *Música Digital*.

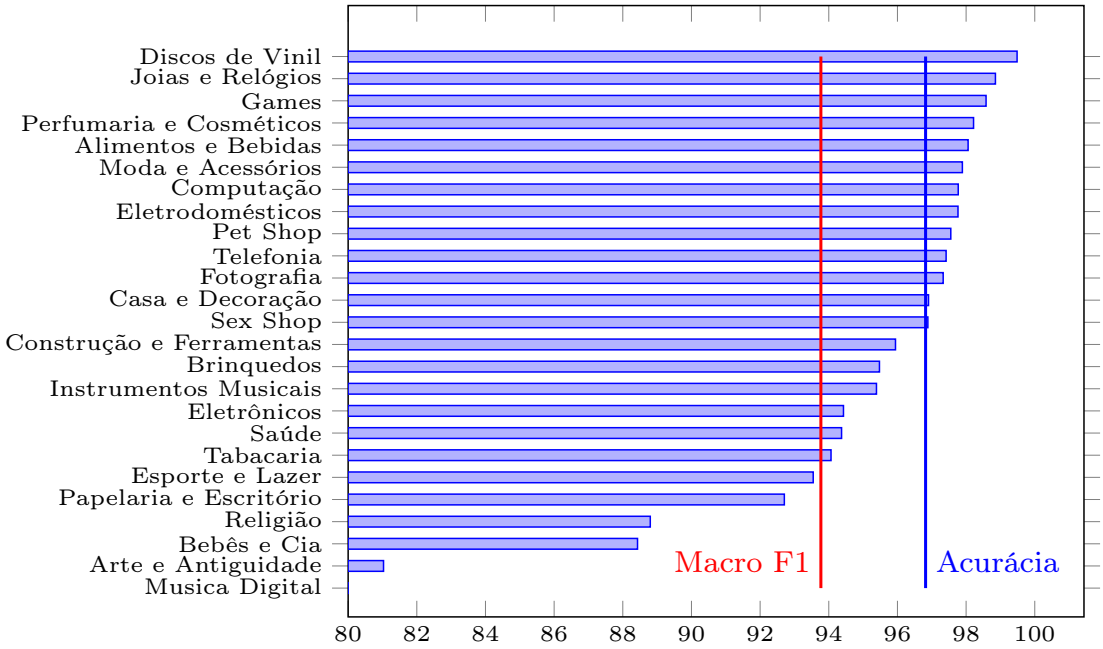

Figura 5.8: Métrica F1 (%) por categoria para o modelo CNN+ftrs.

Na Figura 5.8 incluímos duas linhas verticais indicando a acurácia e o macro F1 do modelo. O macro F1 alcançado é mais de 3 pontos inferior à acurácia. Isso se dá principalmente pelo F1 alcançado pela categoria *Música Digital*. Podemos até mesmo cortar a figura uma vez que o F1 dessa categoria é menor que 23,52%. Este valor é muito menor que o restante e não é uma performance aceitável para um sistema prático. Contudo, essa é uma categoria degenerada, uma vez que sua frequência no conjunto de dados é de aproximadamente 0,0006%. Este é um conceito muito escasso para que o modelo possa aprender sobre ele e ser avaliado. Por exemplo, se removermos essa categoria do conjunto de dados, alcançamos um macro F1 de 96,70%, o que é muito próximo da acurácia atingida.

Terminados os experimentos sobre o conjunto de validação, realizamos a execução de um último experimento, considerando aqui a melhor configuração encontrada para cada classificador. O conjunto de dados para treino consiste em 90% do *dataset*, enquanto os 10% restantes foram utilizados para teste. Os resultados podem ser visualizados na Figura 5.9.

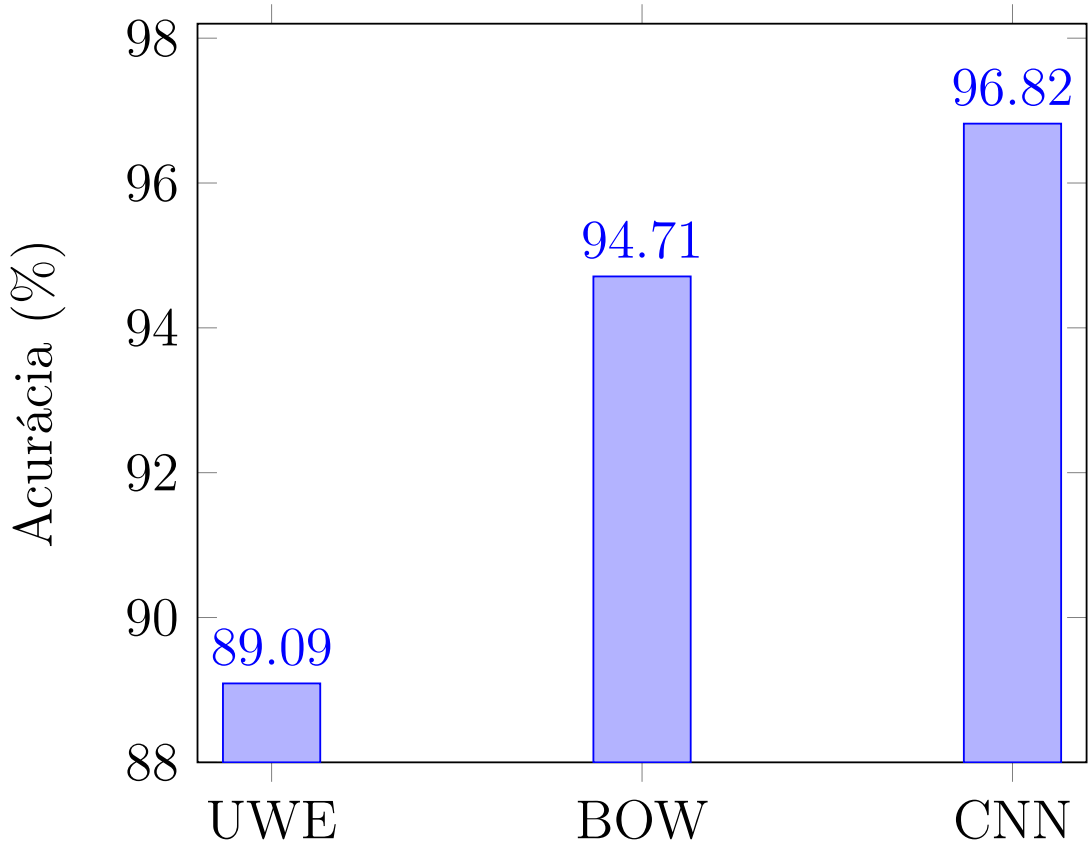

Figura 5.9: Comparação dos resultados dos melhores modelos aplicados ao conjunto de teste.

A seguir, apresentaremos as conclusões do trabalho aqui realizado.

Capítulo 6

Conclusões

6.1 Resumo do Problema Abordado

Neste trabalho, abordamos o problema de categorização de ofertas, que consiste, grosso modo, em inferir, com base nos atributos de uma oferta, a categoria à qual essa oferta pertence. Enfrentado por SCPs, esse problema exige, devido à quantidade massiva de informações coletadas diariamente, o desenvolvimento de algoritmos que possam aprender com a experiência e classificar esses produtos de forma automatizada. Contudo, é necessário que esses algoritmos sejam capazes de extrair informações através do texto que descreve tais ofertas, o que torna imprescindível que eles sejam representados de maneira compreensível à máquina.

Uma das formas mais simples de representação utilizada é o *bag of words*. Contudo, esse modelo possui desvantagens, como a perda da ordem das palavras e a possibilidade de uma mesma representação ser compartilhada por sentenças distintas. Além disso, esse método baseia-se sobretudo em informações pouco descritivas do produto, como ocorrência de uma palavra ou sua frequência no texto. Dessa forma, propomos aqui outras duas abordagens que visam o aprendizado de características de maneira mais complexa, resultando em melhores representações vetoriais e acurácia na classificação.

6.2 Resumo da Proposta

Neste trabalho, propomos a utilização de duas técnicas de aprendizado de máquina em comparação à tecnica *bag of words*. Para os experimentos, utilizamos a base de dados da Buscapé, com cerca de 11 milhões de ofertas. Para a validação dos modelos, dividimos a base em dois subconjuntos, de teste e validação, com 80% e 20%, respectivamente. E por fim, comparamos os melhores modelos de cada abordagem utilizando 90% do conjunto de dados para treino, e 10% para teste.

A primeira abordagem, *word embedding*, consiste na geração de representações vetoriais de palavras em um espaço multidimensional que preservam informações semânticas e do contexto em que cada uma dessas palavras aparece. Esse modelo, apresentado em Bengio et al. (2003), é bastante utilizado em problemas de análise de sentimento (Maas et al., 2011) e detecção de paráfrase (Socher et al., 2011). Para esse modelo, propomos uma análise da dimensionalidade do espaço vetorial na qualidade do classificador UWE em relação ao BOW que utiliza apenas a descrição da oferta.

A segunda abordagem proposta consiste na utilização de redes neurais convolucionais. Esse modelo neural utiliza camadas de convolução e *pooling* para geração de *features* utilizadas pela camada de saída para classificação. Para este modelo, comparamos a aplicação em *word embedding* não-supervisionado com *word embedding* aprendido durante o treinamento da rede. Também fazemos a comparação entre o modelo que utiliza apenas a informação da descrição da oferta com um modelo que incorpora *features* adicionais.

6.3 Resumo dos Resultados

Durante a análise da dimensionalidade do espaço vetorial no classificador *UWE*, obtivemos, no seu melhor modelo, acurácia de 88,91% e macro f1 de 79,18%. Em comparação ao melhor modelo *BOW* utilizado nesta etapa do experimento, a qualidade do primeiro foi ligeiramente inferior, com uma diferença de 1,88% na acurácia

e 2,74% no macro f1.

Em seguida, estudamos o impacto dos diferentes atributos das ofertas sobre a qualidade da classificação do algoritmo utilizando o modelo *BOW*, e pudemos constatar que a *feature* que mais influenciou nos resultados foi o nome da loja. Em relação ao modelo que utiliza apenas o texto da descrição da oferta, este obteve acurácia e macro f1 superiores, com diferenças de 2,54% e 4,63%, respectivamente.

A seguir, comparamos o desempenho do classificador *CNN* utilizando *word embedding* não-supervisionado e supervisionado. O primeiro modelo mostrou-se inferior ao segundo, com diferença de 4,42% na acurácia. Por fim, fizemos a comparação entre o modelo supervisionado, que utiliza apenas a descrição da oferta, com outro que utiliza também a informação do nome da loja e da categoria da loja. Este modelo obteve performance superior, com diferença de 1,19% na acurácia.

Com base nesses resultados, pudemos demonstrar que a abordagem *CNN* produz excelentes resultados quando aplicada à problemas que envolvem informações textuais, superando substancialmente a abordagem *BOW* tradicional. Essa qualidade se dá pela capacidade dessa abordagem de captar informações independente da posição das palavras no texto. Além disso, através do uso de *word embedding*, evitamos a esparsidade presente nos modelos *BOW*.

6.4 Principais Contribuições

A seguir, apresentamos as principais contribuições obtidas pelo presente trabalho.

- *Estudo sobre Categorização de Ofertas*

  Este trabalhou contribuiu como material de estudo em língua portuguesa para o problema de Categorização de Ofertas. Também pudemos promover uma análise desse problema utilizando dados do mundo real, tornando este trabalho uma base sólida e confiável para pesquisas futuras.

- *Análise do impacto das informações de ofertas para os classificadores*

Outra importante contribuição obtida através deste trabalho encontra-se na análise dos atributos das ofertas e sua influência nos resultados dos experimentos. Além disso, pudemos demonstrar a engenharia de *features* utilizada e validá-la, o que servirá como referência para trabalhos vindouros.

- *Comparação entre abordagens estudadas*

  Por fim, os experimentos por nós executados contribuíram para comparar as diferentes metodologias utilizadas em problemas que envolvem processamento de linguagem natural, corroborando a qualidade da metodologia *word embedding* e das redes neurais convolucionais.

6.5 Trabalhos Futuros

Uma possível direção de pesquisa é desenvolver um classificador que utilize tanto a informação textual quanto a informação visual das ofertas, i.e., a descrição e a imagem do produto. Para tanto, aplicaremos a rede neural convolucional para o aprendizado de padrões e, consequentemente, reconhecer o produto presente na imagem. Deste modo, almejamos aumentar a qualidade da classificação.

Almejemos também aplicar as metodologias aqui abordadas sobre o problema de classificação com categorias-filhas, fazendo assim um estudo a respeito da hierarquia entre os dois tipos de categoria que compõem o conjunto de dados utilizado nesse trabalho. Dessa forma, também poderemos investigar de maneira mais precisa a influência do preço do produto no resultado da classificação.

Por fim, outra direção seria dar continuidade aos experimentos com o classificador *word embedding* alimentando-o com informações externas ao domínio do problema, como a base de dados da Wikipédia, disponível online para fins acadêmicos. Assim, visamos obter melhores representações vetoriais dada à maior diversidade de contextos em que cada vocábulo existente se encontra, e, consequentemente, obter uma acréscimo na qualidade do classificador.

Referências

Sven Abels, Axel Hahn, and G Oldenburg. Reclassification of electronic product catalogs: The"apricot"approach and its evaluation results. *Informing Science: International Journal of an Emerging Transdiscipline*, 9:31–47, 2006.

Ebru Arisoy, Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28. Association for Computational Linguistics, 2012.

Samy Bengio and Georg Heigold. Word embeddings for speech recognition. In *Proceedings of the 15th Conference of the International Speech Communication Association, Interspeech*, 2014.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

Yoon Ho Cho and Jae Kyeong Kim. Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce. *Expert systems with Applications*, 26(2):233–246, 2004.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL `http://doi.acm.org/10.1145/1390156.1390177`.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1953048.2078186`.

Eli Cortez, Mauro Rojas Herrera, Altigran S da Silva, Edleno S de Moura, and Marden Neubert. Lightweight methods for large-scale product categorization. *Journal of the American Society for Information Science and Technology*, 62(9): 1839–1848, 2011.

Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

H Grabowski, R Lossack, and J Weißkopf. Datenmanagement, 2002: Datenmanagement in der produktentwicklung [data management: Data management in product development]. *München und Wien*, 2002.

David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pages 1–4, 2006.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Geoffrey E Hinton. Distributed representations. 1984.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

Rie Johnson and Tong Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in neural information processing systems*, pages 919–927, 2015.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3, 2010.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Thomas M Mitchell et al. Machine learning, 1997.

Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM, 2007.

Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252. Citeseer, 2005.

Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, 72:1–19, 2011.

Dmitry Pavlov, Ramnath Balasubramanyan, Byron Dom, Shyam Kapur, and Jignashu Parikh. Document preprocessing for naive bayes classification and clustering with mixture of multinomials. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 829–834. ACM, 2004.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

Hinrich Schütze. Word space. In *Advances in Neural Information Processing Systems 5*. Citeseer, 1993.

Holger Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Ben Wolin. Automatic classification in product catalogs. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–352. ACM, 2002.

Tak-Lam Wong, Wai Lam, and Tik-Shun Wong. An unsupervised framework for extracting and normalizing product attributes from multiple web sites. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 35–42. ACM, 2008.

Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.