PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

WILLIAN EDUARDO BECKER

UMA ABORDAGEM DE REDES NEURAIS CONVOLUCIONAIS PARA ANÁLISE DE SENTIMENTO MULTI-LINGUAL

Porto Alegre
2017

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ABORDAGEM DE REDES NEURAIS CONVOLUCIONAIS PARA ANÁLISE DE SENTIMENTO MULTI-LINGUAL

WILLIAN EDUARDO BECKER

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Rodrigo Coelho Barros

Porto Alegre
2017

Ficha Catalográfica

B396a Becker, Willian Eduardo

Uma abordagem de redes neurais convolucionais para análise de sentimento multi-lingual / Willian Eduardo Becker . – 2017.
85 p.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rodrigo Coelho Barros.

1. Redes Neurais Convolucionais. 2. Inteligência Artificial. 3. Análise de Sentimento. 4. Deep Learning. 5. PLN. I. Barros, Rodrigo Coelho. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Willian Eduardo Becker

Uma Abordagem de Redes Neurais Convolucionais para Análise de Sentimento Multi-lingual

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação, Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 24 de novembro de 2017.

BANCA EXAMINADORA:

Profa. Dra. Karin Becker (UFRGS)

Prof. Dr. Duncan Dubugras Alcoba Ruiz (PUCRS)

Prof. Dr. Rodrigo Coelho Barros (PPGCC/PUCRS - Orientador)

"Por vezes, sentimos que aquilo que fazemos não é, senão, uma gota de água no mar. Mas o mar seria menor se lhe faltasse uma gota."
(Madre Teresa de Calcutá)

AGRADECIMENTOS

A meu orientador, o professor Dr. Rodrigo Coelho Barros, pelos ensinamentos, companheirismo e empenho. Também agradeço por toda a dedicação e paciência durante os momentos difíceis. Foi um privilégio ter convivido com você durante este período.

A meu pai Darci, mãe Janete e irmã Cinthia, por todo o amor, compreensão e por sempre acreditarem em mim. Não foram poucas as vezes que suas palavras de apoio me deram forças para continuar nesta jornada. Parte deste trabalho devo à vocês. Amo muito vocês.

Aos meus colegas do Grupo de Pesquisa em Inteligência do Negócio (GPIN), pelo convívio, discussões, rodas de estudo e confraternizações. Todos vocês foram muito importantes para a realização deste trabalho, e os levarei para o resto da vida.

Ao corpo docente da PUCRS, pela interação, disposição e ensinamentos.

À PUCRS, que tornou possível a realização desta pesquisa, através do auxílio financeiro.

UMA ABORDAGEM DE REDES NEURAIS CONVOLUCIONAIS PARA ANÁLISE DE SENTIMENTO MULTI-LINGUAL

RESUMO

A utilização de redes sociais tornou-se uma atividade cotidiana na sociedade atual. Com o enorme, e ininterrupto, fluxo de informações geradas nestes espaços, abre-se a possibilidade de explorar estes dados de diversas formas. A Análise de Sentimento (AS) é uma tarefa que visa obter conhecimento sobre a polaridade das mensagens postadas, através de diversas técnicas de Processamento de Linguagem Natural, onde a maioria das soluções lida com somente um idioma de cada vez. Entretanto, abordagens que não restringem-se a explorar somente uma língua, estão mais próximas de extraírem todo o conhecimento e possibilidades destes dados. Abordagens recentes baseadas em Aprendizado de Máquina propõem-se a resolver a AS apoiando-se principalmente nas Redes Neurais Profundas (*Deep Learning*), as quais obtiveram bons resultados nesta tarefa. Neste trabalho são propostas três arquiteturas de Redes Neurais Convolucionais que lidam com dados multi-linguais extraídos do Twitter contendo quatro línguas. Os dois primeiros modelos propostos caracterizam-se pelo fato de possuírem um total de parâmetros muito menor que os demais *baselines* considerados, e ainda assim, obtêm resultados superiores com uma boa margem de diferença. O último modelo proposto é capaz de realizar uma classificação multitarefa, identificando a polaridade das sentenças e também a língua. Com este último modelo obtém-se uma acurácia de 74.43% para AS e 98.40% para Identificação da Língua em um *dataset* com quatro línguas, mostrando-se a melhor escolha entre todos os *baselines* analisados.

Palavras-Chave: Redes Neurais Convolucionais, Inteligência Artificial, Análise de Sentimento, Aprendizado Profundo, PLN.

A CONVOLUTIONAL NEURAL NETWORK APPROACH FOR MULTI-LINGUAL SENTIMENT ANALYSIS

ABSTRACT

Nowadays, the use of social media has become a daily activity of our society. The huge and uninterrupt flow of information in these spaces opens up the possibility of exploring this data in different ways. Sentiment Analysis (SA) is a task that aims to obtain knowledge about the polarity of a given text relying on several techniques of Natural Language Processing, with most of solutions dealing with only one language at a time. However, approaches that are not restricted to explore only one language are more related to extract the whole knowledge and possibilities of these data. Recent approaches based on Machine Learning propose to solve SA by using mainly Deep Learning Neural Networks have obtained good results in this task. In this work is proposed three Convolutional Neural Network architectures that deal with multilingual Twitter data of four languages. The first and second proposed models are characterized by the fact they require substantially less learnable parameters than other considered baselines while are more accurate than several other Deep Neural architectures. The third proposed model is able to perform a multitask classification by identifying the polarity of a given sentences and also its language. This model reaches an accuracy of 74.43% for SA and 98.40% for Language Identification in the four-language multilingual dataset. Results confirm that proposed model is the best choice for both sentiment and language classification by outperforming the considered baselines.

Keywords: Convolutional Neural Networks, Artificial Intelligence, Sentiment Analysis, Deep Learning, NLP.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AM – Aprendizado de Máquina

AS – Análise de Sentimento

API – *Application Programming Interface*

BOW – *Bag-of-words*

BLR – *Bayesian Logistic Regression*

CNN – *Convolutional Neural Network*

LSTM – *Long Short-Term Memory*

ME – Máxima Entropia

MNB – *Multinomial Naïve Bayes*

MLP – *Multi-layer Perceptron*

NB – *Naïve Bayes*

NLTK – *Natural Language Toolkit*

POS – *Part-of-Speech*

PLN – Processamento de Linguagem Natural

ReLU – *Rectified Linear Units*

RNA – Rede Neural Artificial

RNN – *Recurrent Neural Network*

SVM – *Support Vector Machine*

SUMÁRIO

1. INTRODUÇÃO

1.1 Motivação

Atualmente, a informação está disposta nos mais diversos lugares. O recente avanço tecnológico impulsionou o desenvolvimento de ferramentas que visam auxiliar atividades cotidianas, como tomar um táxi ou escolher um produto para compra, ou até mesmo outras atividades mais específicas e complexas, como diagnóstico médico ou análises laboratoriais.

As mídias sociais, fruto deste avanço tecnológico recente, criaram uma nova forma de interação entre as pessoas, levando esta para o mundo virtual. Desta forma, devido a grande adesão e larga alcançabilidade de seu conteúdo, as redes sociais são uma ótima forma de gerar e consumir conteúdo dos mais variados tipos e para as mais diversas aplicações ou públicos. O simples ato de escolher uma marca para comprar, ou qual filme assistir no final de semana, pode ser diretamente influenciado pelas experiências de outros usuários os quais compartilham a sua opinião sobre os produtos na rede.

Com o enorme, e ininterrupto, fluxo de informações geradas nas redes sociais, abre-se a possibilidade de explorar estes dados de diversas formas. A Análise de Sentimento (do inglês *Sentiment Analysis*) é uma tarefa que visa obter conhecimento sobre a polaridade (positiva, neutra ou negativa) das mensagens postadas, através de diversas técnicas de Processamento de Linguagem Natural (PLN). Basicamente, a Análise de Sentimento pode ser dividida em quatro etapas gerais [3]: (1) *Coleta*, fase onde os dados são extraídos da fonte; (2) *Pré-processamento*, fase a qual aplica-se uma série de processos os quais visam melhorar a qualidade dos dados; (3) *Classificação*, onde utiliza-se a abordagem escolhida (dicionário, aprendizado de máquina ou estatística/semântica) para classificar a polaridade das sentenças; e (4) *Sumarização*, fase onde os resultados obtidos na etapa anterior são totalizados e validados.

A maioria das soluções para Análise de Sentimento lida com somente um idioma de cada vez. Entretanto, abordagens que não restringem-se a explorar somente uma língua, estão mais próximas de extraírem todo o conhecimento e possibilidades dos dados. Se tratando de Redes Sociais, como o Twitter, o conteúdo está disposto nos mais diferentes idiomas, fazendo com que estratégias multi-linguais sejam de extrema valia neste contexto.

1.2 Formulação do Problema

A maioria dos trabalhos que visam resolver a Análise de Sentimentos multi-lingual apoiam-se principalmente em sistemas de traduções dos textos [4, 5, 6, 7, 8]. Neste caso, converte-se o conjunto de testes para a mesma língua do conjunto de treino ou vice-versa, deixando ambos os conjuntos na mesma linguagem, sendo possível assim realizar uma classificação mono lingual. Mesmo

que as abordagens baseadas em traduções pareçam intuitivas, dois problemas principais podem afetar a qualidade da classificação: primeiro, o problema de tradução de textos já é uma tarefa desafiadora e custosa por si só, onde uma tradução errada ou infiel ao contexto afetará direta e negativamente o resultado; segundo, a maioria dos sistemas de tradução geram somente a sua melhor combinação de resultado para um determinado texto, limitando assim a diversidade e particularidades de uma determinada língua a um mesmo resultado. Então, mesmo que a tradução dos conjuntos de treino ou de teste seja perfeita, a classificação multi-lingual poderá não ter o mesmo desempenho que uma classificação mono lingual com dados de treino e teste da mesma língua [9].

Outra desvantagem relacionada à tradução de texto para a AS multi-lingual é a disponibilidade de boas ferramentas *open-source*. A maioria das APIs gratuitas oferecem um determinado número de operações, não sendo adequadas para traduções em larga escala. Assim sendo, a adoção de APIs para a tradução de grandes volumes de dados pode acabar tornando-se cara e inviável.

Abordagens recentes baseadas em Aprendizado de Máquina propõem-se a resolver a AS apoiando-se principalmente nas Redes Neurais Profundas (*Deep Learning*), as quais obtiveram bons resultados recentemente nesta tarefa. Entretanto, o treinamento deste tipo de algoritmo, dependendo de sua configuração e tamanho do conjunto de treinamento, pode levar um tempo considerável. Mesmo com o avanço da tecnologia de hardware utilizada para processamento, e também com a utilização de GPUs (Unidade de Processamento Gráfico, do inglês *Graphics Processing Unit*), ainda é comum encontrar cenários onde o treinamento pode levar dias.

1.3 Abordagem Proposta

Tendo em vista as aplicações de Redes Neurais Convolucionais para as mais diversas finalidades, buscou-se encontrar uma arquitetura que fosse capaz de resolver a Análise de Sentimento de dados multi-linguais obtido do Twitter. Para tal, a arquitetura da CNN foi sendo evoluída com o objetivo de melhorar a sua capacidade preditiva. Desta forma, o presente trabalho apresenta três arquiteturas que alcançaram resultados bastante competitivos, superando modelos considerados *estado da arte*.

As três arquiteturas propostas utilizam uma representação a nível de caractere, porque, mesmo com um pequeno conjunto de caracteres, é possível gerar uma quantidade enorme de palavras. Como a análise é realizada em um cenário multi-lingual, este tipo de representação é altamente recomendada, pois o alfabeto de possíveis caracteres nas diferentes línguas é somente um pouco maior se comparado ao alfabeto utilizado para mapear uma única língua. Caso fosse adotada uma representação baseada em palavras, o vocabulário seria consideravelmente expressivo, e cresceria a cada nova língua que fosse adicionada à análise. Ainda, por se tratar de uma representação baseada em caracteres, esta abordagem dispensa a necessidade de pré-processamento nos dados para realizar o treinamento.

O primeiro modelo proposto intitula-se Conv-Char-R, foi inspirado originalmente naquele proposto por Zhang et al. [2], porém com algumas modificações em sua estrutura. Estas alterações na arquitetura simplificaram o seu funcionamento e agilizaram o treinamento, conseguindo alcançar um resultado competitivo se comparado aos demais classificadores.

A segunda arquitetura de CNN proposta, a Conv-Char-S, possui uma arquitetura ainda mais simples que o modelo anterior, contento três elementos básicos: uma camada convolucional, uma camada de *max-overtime-pooling* e uma camada densamente conectada contendo um neurônio para cada classe (positiva e negativa). A camada convolucional realiza a operação de convolução com 256 filtros deslizantes de tamanho 7xn, onde n é o tamanho do vocabulário. A camada seguinte da rede realiza a operação de *max-overtime-pooling*, a qual mantém as *features* mais importantes que são passadas para a camada seguinte. Nesta camada aplica-se uma função *dropout* com probabilidade de 0.9. A última camada da rede, a qual possui somente dois neurônios e é densamente conectada com a camada anterior, realiza uma função *Softmax*, onde é computada a probabilidade da instância pertencer a cada classe.

O terceiro modelo apresentado, a NNLS, é uma arquitetura de CNN multitarefa, capaz de realizar a Análise de Sentimento e também a Identificação da Língua. Esta é a primeira abordagem que utiliza Redes Neurais Convolucionais com representação a nível de caractere em um cenário multi-lingual para realizar estas duas tarefas de classificação. Neste modelo utiliza-se três fluxos, cada um com uma ou mais camadas convolucionais, onde é realizado um aprendizado hierárquico nos dados de entrada.

1.4 Resultados

Para a validação dos modelos propostos foram executados diversos experimentos envolvendo um *dataset* de *tweets*, analisando individualmente as diferentes línguas contidas neste e também o cenário multi-lingual. Para tal, foram testados diversos algoritmos, como o SVM com diversas combinações de representações, CNN utilizando *word-embeddings* como representação, e também uma arquitetura mais complexa de CNN, a qual utiliza representação a nível de caractere, além da LSTM.

A Conv-Char-R obteve uma acurácia bem competitiva se comparada aos demais *baselines*, alcançando 0.695 de acurácia e 0.722 de F-measure, praticamente os mesmos resultados da CNN com representação a nível de caractere e que possui uma estrutura mais complexa e com muito mais parâmetros treináveis. O segundo modelo proposto, a Conv-Char-S, alcançou uma acurácia de 0.721 e F-measure de 0.752 no conjunto multi-lingual, superando assim todos os *baselines* testados, incluindo modelos considerados estado da arte em tarefas de AS, como LSTM. A arquitetura que utiliza CNN e *word-embeddings* obteve o segundo melhor desempenho, alcançando um acurácia de 0.715 e F-measure de 0.750, valores ligeiramente menores que a Conv-Char-S. A última arquitetura proposta, a NNLS, considerando uma classificação multitarefa, obteve o melhor resultado preditivo,

com uma acurácia de 74.43% para AS e 98.40% para Identificação da Língua, superando o segundo melhor *baseline* por uma boa margem.

Também foram analisadas outras características do treinamento dos modelos, buscando identificar aqueles que necessitam um maior recurso de máquina para realizar o seu treinamento, bem como aqueles que demandam mais tempo neste processo. A Conv-Char-S foi a arquitetura que menos gerou parâmetros treináveis e também a que menos memória exigiu, totalizando $1225\times$ menos parâmetros e utilizando $\approx 4\times$ menos memória que o segundo melhor *baseline*. Ainda, este modelo obteve o menor tempo para treinamento por época, necessitando até $\approx 4.5\times$ menos tempo que os demais modelos analisados.

1.5 Contribuições

O presente trabalho apresenta três arquiteturas de Redes Neurais Convolucionais distintas. A aplicação dos modelos propostos, conciliando com uma representação a nível de caractere, em um cenário multi-lingual, mostra-se uma ótima escolha. Desta forma, não se faz necessário treinar um algoritmo para cada diferente língua, fato que tornaria o treinamento muito mais custoso. Ainda, em cenários de redes sociais, é comum que uma mensagem contenha palavras de outras línguas. Caso fosse adotado um classificador para cada língua, ocorrências como esta não receberiam o devido tratamento.

A arquitetura Conv-Char-R é a primeira a utilizar representação a nível de caractere e Redes Neurais Convolucionais para realizar a Análise de Sentimento considerando dados multi-linguais. Outro modelo apresentado, a Conv-Char-S, mostra uma arquitetura de Rede Neural Convolucional com um número muito reduzido de parâmetros, e que mesmo assim apresenta um ótimo desempenho na tarefa de Análise de Sentimento em cenário multi-lingual. Também, por considerar uma representação a nível de caracteres, esta abordagem não necessita de uma fase de pré-processamento dos dados. Outra contribuição deste modelo diz respeito à sua excelente capacidade de treinamento em cenários com recursos de hardware mais limitados. A Conv-Char-S possui $\approx 90\times$ menos parâmetros e consome $5\times$ menos memória que as arquiteturas baseadas em *word-embeddings*. Estas características resultam em um tempo de treinamento até $\approx 4.5\times$ menor que os demais *baselines*.

A NNLS mostrou-se o melhor modelo dentre todos aqueles apresentados neste trabalho, alcançando resultados superiores com uma boa margem. Esta arquitetura introduz o conceito de aprendizagem hierárquica na representação dos caracteres de entrada, realizando posteriormente uma classificação multitarefa. Esta é a primeira arquitetura de Rede Neural Convolucional multitarefa com representação a nível de caractere para a Análise de Sentimento e Identificação da Língua em cenários multi-linguais que se tem conhecimento.

1.6 Estrutura do Trabalho

No Capítulo 2 será apresentado o referencial teórico, o qual discorre sobre Análise de Sentimento, representação de informações e Redes Neurais. A abordagem proposta será apresentada no Capítulo 3. No Capítulo 4 serão apresentados os diferentes tópicos envolvendo a realização dos experimentos, bem como os resultados destes. Por fim, no Capítulo 5 serão apresentadas as conclusões dos experimentos e trabalhos futuros.

2. REFERENCIAL TEÓRICO

2.1 Análise de Sentimento

Saber o que outras pessoas pensam sobre algo sempre foi importante para a maioria das pessoas durante o processo de tomada de decisão [10]. Comprar um produto ou uma marca, ou ainda visitar um determinado lugar, são exemplos de atividades que podem ser realizadas com base em experiências relatadas por outras pessoas. Empresas também procuram saber o que os usuários pensam sobre seus produtos, visando melhorar os serviços oferecidos e consequentemente melhorar seu posicionamento no mercado [11].

A ascensão das mídias sociais, como blogs e redes sociais tem despertado um grande interesse, em especial na tarefa de Análise de Sentimentos (AS). A Figura 2.1 endossa tal situação ao exibir o crescimento na quantidade de buscas pelo termo "Sentiment Analysis" no buscador Google desde 2005. Com a proliferação de opiniões, avaliações, recomendações e outras formas de expressão *online*, o conteúdo opinativo transformou-se em uma espécie de moeda virtual para empresas que desejam comercializar os seus produtos, identificar novas oportunidades e gerenciar suas reputações. Como as empresas estão cada vez mais interessadas em automatizar o processo de pesquisa sobre o que está sendo falado sobre elas, a compreensão das conversas nas mídias sociais, a fim de identificar conteúdo relevante, que permita tomar ações específicas, tem levado as empresas investirem cada vez mais recursos em pesquisas relacionadas à AS.

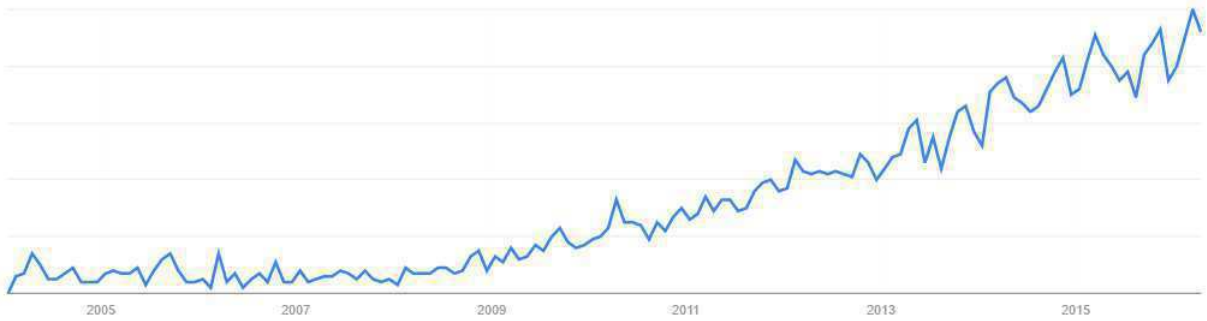


Figura 2.1 – Pesquisa do termo "Sentiment Analysis" no Google Trends[1].

Segundo Liu [12], a Mineração de Opinião (ou Análise de Sentimento) é uma área de pesquisa recente que congrega pesquisas de Mineração de Dados, Aprendizado de Máquina (AM), Recuperação de Informação, Linguística e Processamento de Linguagem Natural (PLN). Seu objetivo é analisar a opinião das pessoas, identificando seus sentimentos, avaliações, atitudes e percepções relacionadas a um determinado alvo a partir de uma linguagem escrita [12]. Formalmente, uma opinião pode ser definida como correspondente a uma quíntupla ($\mathbf{e}_i$, $\mathbf{a}_{ij}$, $\mathbf{s}_{ijkl}$, $\mathbf{h}_k$, $\mathbf{t}_l$), proposta por Liu [12], onde:

[1] http://google.com/trends

- $\mathbf{e}_i$: é o nome de uma entidade;
- $\mathbf{a}_{ij}$: é um aspecto da entidade $\mathbf{e}_i$ (opcional);
- $\mathbf{s}_{ijkl}$: é a polaridade do sentimento sobre aspecto $\mathbf{a}_{ij}$ que tem como alvo a entidade $\mathbf{e}_i$;
- $\mathbf{h}_k$: é o detentor do sentimento (i.e. quem expressou o sentimento), também chamado de fonte de opinião;
- $\mathbf{t}_l$: é o instante no qual a opinião foi expressa por $\mathbf{h}_k$.

Uma definição mais simplificada diz que uma opinião basicamente é composta por dois elementos principais: um **alvo** e um **sentimento** sobre esse alvo [12]. Um alvo pode ser representado por uma entidade, indivíduo, tópico ou marca. Já um sentimento é representado por elementos subjetivos, que podem ser emoções, avaliações e experiências expressas pelo autor a respeito de um determinado alvo. A polaridade de um sentimento significa se ele quer dizer algo positivo ou negativo, ou ainda ser neutro caso não seja possível definir se este é positivo ou negativo.

2.1.1 Etapas

O processo de AS pode ser definido genericamente através das seguintes tarefas: coleta do conteúdo, pré-processamento dos dados, classificação do sentimento e sumarização dos resultados. Esta arquitetura é apresentada na Figura 2.2.

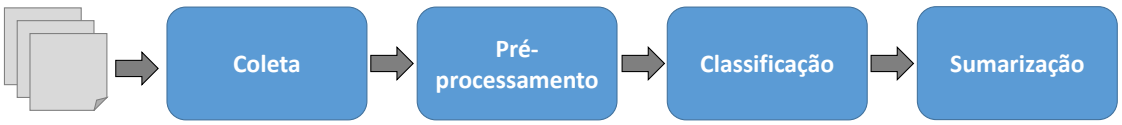


Figura 2.2 – Etapas da Análise de Sentimento.

- **Coleta:** Nesta etapa é realizada um busca em fontes diversas, tais como artigos em sites, comentários, anúncios, documentos, redes sociais dentre outras. Neste passo é importante selecionar somente mensagens relacionadas ao alvo a ser analisado, descartando as demais. Ao trabalhar com dados de redes sociais, por exemplo, é normal que diferentes assuntos sejam discutidos simultaneamente. Assim sendo, se faz necessária a criação de mecanismos de filtragem, o qual deverá manter somente o conteúdo relevante à análise.
- **Pré-processamento:** A tarefa de pré-processamento inclui uma série de rotinas, processos e métodos necessários para melhorar a qualidade dos dados e alcançar melhores resultados na definição do sentimento das mensagens. Esta etapa se faz ainda mais importante quando se deseja trabalhar com dados informais, como mensagens de redes sociais, por exemplo. Neste

ambiente são compartilhadas mensagens que não seguem rigorosamente as regras gramaticais, comprometendo o resultado final. Ainda, outro problema inerente ao processamento de textos é a alta dimensionalidade dos dados. Assim sendo, existem diversas técnicas de pré-processamento de dados que podem ser aplicadas para amenizar estes problemas e facilitar a análise, tais como: remoção de *stop-words*, correção ortográfica, lematização, *stemming* dentre outros.

- **Classificação**: A tarefa de classificação é etapa mais importante do processo de AS, pois nesta etapa é atribuída a polaridade ou orientação da opinião, determinando se ela é positiva, neutra ou negativa. Segundo Tsytsarau e Palpanas [13], as abordagens propostas para classificar a polaridade de um texto podem ser divididas em três grandes grupos:

  1. Abordagem baseada em Dicionário: este tipo de abordagem utiliza-se de um dicionário (também chamado de léxico) de palavras que possuem uma polaridade definida, sendo esta positiva ou negativa. Para classificar a polaridade de uma palavra encontrada no texto, primeiro se faz necessário anotar as sentenças, definindo a qual classe gramatical cada palavra pertence. Em seguida, deve-se identificar no texto as palavras que possuem algum sentimento (e.g. adjetivos) e então consultar esta palavra no dicionário para definir sua polaridade. Ainda, pode-se utilizar um léxico contendo a polaridade de cada palavra medidas em uma escala, onde a definição do sentimento da sentença dá-se por meio do cálculo da média entre a polaridade das palavras presentes.

  2. Abordagem baseada em Aprendizado de Máquina: A abordagem baseada em Aprendizado de Máquina visa descobrir automaticamente regras gerais em grandes conjuntos de dados, de maneira que se possa encontrar informações implicitamente representadas [14]. Atualmente, a maioria das abordagens baseadas em AM para PLN consistem em *tokenizar* um conjunto de caracteres em estruturas como palavras, sentenças e parágrafos, e então aplicar algum algoritmo de classificação estatística nestas estruturas [15].

  3. Abordagens Estatísticas e Semânticas: a abordagem Estatística baseia-se na premissa que palavras que expressam alguma opinião frequentemente são encontradas juntas em uma sentença. Assim sendo, se uma palavra ocorre frequentemente ao lado de uma palavra positiva, então provavelmente ela também será positiva [14]. Assim sendo, caso uma determinada palavra apareça diversas vezes próxima de uma palavra *positiva* ou *negativa*, pode-se assumir que a polaridade entre estas é compartilhada. Entretanto, quando uma palavra aparece em igual frequência entre as classes, pode-se classificá-la como *neutra*. A abordagem Semântica assemelha-se com a Estatística, diferenciando-se somente pelo uso de uma métrica para cálculo da distância entre as palavras[13]. Neste caso, as palavras que estão semanticamente próximas são classificadas com a mesma polaridade.

- **Sumarização:** Etapa focada na validação e apresentação dos resultados. A forma mais simples de sumarizar os resultados é através de gráficos, onde os resultados são sumarizados em totais e dados estatísticos, facilitando a sua visualização e entendimento.

2.1.2 Níveis de Análise

Textos e sentenças possuem diversas informações, algumas delas fáceis de serem encontradas e extraídas, enquanto outras demandam relacionamentos complexos de palavras para serem compreendidas em sua totalidade. Na literatura existem diferentes níveis de granularidade que tratam problemas de AS. Conforme Liu [12], a decisão do nível a ser utilizado em uma aplicação irá depender do grau de informação que deseja-se extrair da mesma. Dessa forma, os níveis de análise aplicados na AS estão divididos em três principais: **documento**, **sentença** e **aspecto**.

- **Documento:** Este nível de análise realiza a classificação da polaridade de um documento por inteiro, verificando se a opinião expressa neste é positiva, negativa ou neutra [16]. Esta granularidade é adequada quando o documento trata de uma única entidade [14], não sendo aplicável quando mais de um alvo é mencionado. A Figura 2.3 apresenta um exemplo de mensagem onde esta análise poderia aplicada.

Figura 2.3 – Opinião de um usuário sobre um determinado hotel.

- **Sentença:** Classifica a polaridade do sentimento expresso em cada sentença de um documento. Normalmente, esse nível é utilizado quando deseja-se analisar o sentimento de alvos distintos em um único documento ou para realizar uma análise detalhada dos sentimentos expressos sobre um alvo específico. Também pode-se dividir a mesma sentença em diversas cláusulas (e.g. "A areia é linda, mas a água é suja.") [17]. Cabe ressaltar que, em geral, postagens e comentários em mídias sociais seguem um padrão de sentenças curtas. A Figura 2.4 apresenta uma mensagem compartilhada em uma rede social, onde a análise a nível de sentença poderia ser aplicada.

- **Aspecto:** Também chamado de *feature level*, neste nível é realizada uma análise com granularidade mais fina, com foco na opinião expressa e não interessando os construtos utilizados para expressá-la (e.g. documento, sentença, oração). O termo *aspecto*, também chamado de característica, é um nível de análise que permite identificar o sentimento de um alvo sob

Figura 2.4 – Comentário de um usuário sobre a comida de um hotel.

diferentes perspectivas, considerando seus diferentes atributos. A Figura 2.5 ilustra um exemplo de um comentário realizado no Twitter, onde é possível identificar o sentimento do autor da postagem para diferentes aspectos de um hotel. Neste exemplo um usuário qualifica um determinado hotel, considerando três aspectos: atendimento, café da manhã e acomodações, além de expor sua opinião sobre o hotel, considerando-o uma "ótima opção"e um "excelente Hotel".

Figura 2.5 – Opinião de um usuário sobre um hotel e seus aspectos.

Devido ao aumento do conteúdo opinativo das experiências de usuários sobre produtos e serviços, houve uma grande demanda por aplicações que obtivessem uma granularidade maior de cada mensagem ou comentário, aplicando a análise a nível de aspecto. Muitas empresas desejam monitorar a experiência de seus usuários quando utilizando seus produtos. Por exemplo, empresas fabricantes de celulares podem recorrer à opinião de seus usuários quando uma atualização em seu produto é lançada, buscando identificar em quais aspectos do aparelho (câmera, bateria, design e outros) esta atualização teve impacto positivo ou negativo. Em cenários como estes a análise a nível de aspecto seria mais assertiva.

2.1.3 Desafios e Aplicações

Recentemente, a AS tornou-se um importante tópico de pesquisa em áreas como AM e PLN, motivados principalmente pelos desafios e dificuldades envolvidos nesse processo. Em tarefas de classificação textual, grande parte do tempo é empregado na etapa de pré-processamento. Além disso, com o crescimento das redes sociais online, pesquisas em AS se deparam com um problema em particular: o “internetês”. Em redes sociais, por exemplo, é comum que mensagens contenham

erros gramaticais, abreviações, contrações e outras alterações na estrutura textual. Essas variações prejudicam a capacidade de generalização dos dados e aumentam o espaço dimensional, exigindo que estratégias de pré-processamento apropriadas sejam aplicadas.

Um problema inerente ao processamento de textos é a alta dimensionalidade dos dados. A coleção de dados utilizada para a mineração de textos, por menor que seja, pode facilmente conter milhares de termos, muitos deles redundantes e desnecessários, tornando lento o processo de extração de conhecimento e prejudicando a qualidade dos resultados. A fim de contornar este problema, deve-se ter uma estratégia de pré-processamento bem definida, visando a redução da dimensionalidade dos dados, facilitando a montagem da representação estruturada para a classificação do sentimento.

A Análise de Sentimento tornou-se muito importante recentemente, principalmente devido ao crescimento das redes sociais. Atualmente é comum que o consumidor, além de realizar suas compras online, também utilize portais de *reviews*, blogs e fóruns de discussão para compartilhar experiências de consumo e considerações sobre produtos e serviços adquiridos. Acompanhando este crescimento, muitas empresas passaram a monitorar seus produtos e também seus usuários, visando obter informações sobre a aceitação ou rejeição de seus produtos, bem como moldar seu negócio de maneira que seus consumidores tenham uma melhor experiência. Na literatura existem diversos trabalhos que aplicam técnicas de AM em dados opinativos [18, 19, 16]. Além de explorar *reviews* de produtos, alguns estudos também analisam o sentimento relacionados à determinadas instituições, hotéis e restaurantes, indicando que esta é uma área capaz de atingir diversos segmentos de mercado, auxiliando na descoberta de conhecimento e na tomada de decisões.

As mídia sociais possibilitam ainda outros tipos de trabalhos, como o monitoramento do desempenho dos candidatos, quando em época de campanha eleitoral [20], ou até mesmo durante debates políticos. Entre outras aplicações, destacam-se o reconhecimento de sentimento do público em grandes eventos [21] ou eventos esportivos [22], suporte na previsão de flutuações no mercado financeiro [23] e ainda o monitoramento da saúde pública através das mensagens compartilhadas nas redes sociais [24].

Pesquisadores do Facebook realizaram um experimento [25] explorando as emoções presentes nas mensagens postadas na sua rede social. Neste experimento os pesquisadores utilizaram o LIWC[2] [26] para definir a polaridade das postagens em sua rede social e limitar a exibição de postagens com conteúdo apenas negativo na linha do tempo de um determinado grupo de usuários e apenas positivo na linha do tempo de outro grupo. O objetivo era verificar a ocorrência de contágio emocional em redes sociais, ou seja, definir o quanto postagens negativas ou positivas influenciam no humor e nas futuras postagens de quem as lê. Este tipo de estudo reforça a capacidade de

[2]O LIWC é uma ferramenta (disponível em http://liwc.wpengine.com) bem estabelecida e utilizada em diversas áreas, criada com o objetivo de classificar palavras emotivas em diferentes categorias, e contou com o aval de psicólogos, sociólogos e linguistas durante seu desenvolvimento. Ela possui um dicionário léxico de aproximadamente 4500 palavras e raízes de palavras, fazendo parte de oitenta categorias. Todas as palavras de raiva, por exemplo, estão incluídas em pelo menos duas categorias: emoções negativas e emoção geral. As categorias foram definidas e populadas com base em buscas em diversos dicionários, questionários e listas feitas por pesquisadores. Para a correta definição, três juízes independentes julgaram cada uma das palavras e definiram em qual categoria ela deveria estar.

entender o comportamento humano e também as interações dos usuários baseando-se na análise do conteúdo postado em mídias sociais.

2.1.4 Análise de Sentimento multi-lingual

A maioria dos sistemas de AS construídos até hoje foram concebidos para tratar informações de somente uma língua (Análise de Sentimento mono lingual), tendo principalmente documentos em inglês como alvo. Mesmo sendo inglês a língua mais utilizada na internet, a fatia de usuários desta engloba somente um total de 26.3% dos usuários[3], justificando assim a importância da criação de ferramentas para trabalhar com as demais línguas. Nos últimos anos houve um crescimento em aplicações que fossem capazes de lidar com dados de duas ou mais línguas. Grande parte desta motivação originou-se no desejo de empresas em ter uma visão global da aceitação de suas marcas e/ou produtos, não limitando-se somente ao conteúdo gerado em uma determinada língua.

Uma abordagem que não se restringe a analisar dados de somente uma linguagem é capaz de, após a coleta, analisar um total muito maior de dados. Esta premissa ganha mais força quando as informações são provenientes de redes sociais, como o Twitter e Facebook. Nestes espaços, os quais são bem democráticos no que tange a língua utilizada, existe um universo bem expressivo de dados nas mais variadas línguas.

A maioria dos trabalhos que visam resolver a AS multi-lingual realiza uma das abordagens abaixo apresentadas:

- Traduzir os documentos da linguagem original para inglês e então realizar a AS [27];
- Criar um léxico contendo palavras subjetivas em cada linguagem a ser considerada na análise [28].

A utilização de métodos para AS multi-lingual é completamente dependente de recursos oriundos da tradução de documentos ou de léxicos. Assim sendo, a própria tradução de documentos já pode ser considerada uma tarefa difícil e desafiadora, seja por sua complexidade ou mesmo pelo tempo necessário para a sua realização. Além disso, o aprendizado destes métodos ocorre em *features* na linguagem original, e quando estas informações são confrontadas com a linguagem alvo estes métodos tradicionais podem não funcionar [29].

Nos dois tópicos a seguir serão apresentadas as principais características das abordagens tradicionalmente aplicadas na tarefa de Análise de Sentimento multi-lingual.

- **Abordagens baseadas em Léxico:** nesta abordagem não supervisionada a classificação do sentimento é realizada com base no cálculo da polaridade de uma determinada sentença ou documento, utilizando-se de um dicionário (léxico) o qual contém palavras em uma determinada língua e a magnitude do sentimento atrelado à esta. Primeiramente realiza-se uma varredura

[3] http://www.internetworldstats.com/stats7.htm - 30 de Junho de 2016.

em todos os documentos, ignorando *stopwords* ou palavras objetivas, visando identificar nas sentenças as palavras contidas no léxico, para então calcular a polaridade de cada documento. Após esta verificação é possível obter a polaridade ζ_d de cada documento d somando todas as pontuações ζ_t de cada palavra em t em todas as sentenças s deste documento, *i.e.*,

$$\zeta_{(d)} = \sum_{S \in d} \sum_{t \in S} \zeta_d \tag{2.1}$$

O valor resultante do cálculo do sentimento do documento d é então utilizado para definir a qual classe c este pertence, *i.e.*,

$$c_{(d)} = \begin{cases} \text{positiva} & \text{se } (\zeta_d - \epsilon) \geq 0, \\ \text{negativa} & \text{se } (\zeta_d - \epsilon) < 0, \end{cases} \tag{2.2}$$

onde ϵ é um valor (*bias*) que aumenta a possibilidade do documento d ser da classe positiva. Este *bias* é opcional e pode ser útil pois, conforme apresentado por Taboada et al. [30], as pessoas têm uma tendência a escreverem textos visivelmente negativos mesmo estes contendo algumas palavras positivas (e.g., *"este carro já foi bom, hoje não é mais"*).

Nesta abordagem, também é frequente a realização de um tratamento especial para casos de negação, intensificadores ou frequência de uma palavra em uma sentença. Quando aplica-se algum tratamento deste tipo, a polaridade não é analisada a cada palavra de uma sentença, e sim considerando um contexto prévio e\ou posterior, de tamanho a ser definido, para o cálculo do *score* de sentimento da palavra em questão.

- **Abordagens baseadas em Corpus:** este tipo de abordagem apoia-se primordialmente no uso de sistemas de traduções de textos. Para tal, primeiramente realiza-se a tradução do conjunto de teste para a mesma língua do conjunto de treino (ou vice-versa), podendo assim aplicar uma classificação mono lingual.

  Caso fosse possível, a tradução manual seria a forma mais indicada para a realização desta tarefa. Entretanto, cenários de AS facilmente consideram milhares (muitas vezes até milhões) de instâncias, esta tarefa precisa ser automatizada. Para tal, existem diversas ferramentas disponíveis: Google Translate[4], Bing Translator[5] e Moses [31] são alguns exemplos.

  Abordagens baseadas em tradução (supervisionadas) normalmente sofrem de dois problemas gerais. Um deles é que sistemas de tradução sempre apresentam somente o melhor resultado, descartando detalhes e particularidades de cada documento. Isto pode gerar uma perda na qualidade da tradução, diminuindo assim a acurácia da classificação do sentimento. Outro fator que pode influenciar negativamente na classificação é a diferença entre a distribuição das palavras na linguagem original e na alvo, onde palavras que ocorrem muito em uma

[4] http://translate.google.it/
[5] http://www.microsofttranslator.com/

língua podem não aparecer muito na outra. Neste casos, a classificação multi-lingual terá um resultado pior do que classificadores mono lingual.

2.2 Representação de Informações

Segundo Aggarwal-Zhai [32], o processo clássico de mineração assume que os dados estão contidos em bases de dados concretas e estruturadas, onde os atributos dos objetos são conhecidos e o pré-processamento normalmente está focado na integração, limpeza e normalização dos dados. Por outro lado, em tarefas de AS onde os dados não possuem uma estrutura bem definida e normalmente contêm muito conteúdo irrelevante para a análise, faz-se necessária uma etapa preparatória visando eliminar qualquer informação que possa interferir na representação das informações, para a posterior atribuição do sentimento.

Algoritmos de aprendizado para classificação não são capazes de processar diretamente os documentos de texto em sua forma original, ou seja, é necessário transformar letras para números. Para tal, são executadas atividades de tratamento e padronização da coleção de textos, seleção dos termos (palavras) mais significativos e, por fim, representação da coleção textual em um formato estruturado que preserve as características necessárias aos objetivos definidos na etapa de identificação do problema [33]. Dessa forma, durante a etapa de pré-processamento, os documentos devem ser convertidos para um modelo de representação estruturado, descrevendo os dados da forma mais efetiva possível. Normalmente, os documentos são representados por vetores de características, ou seja, um vetor de atributos. Logo, a estrutura de um documento nada mais é que a representação de seu vetor de características em um espaço dimensional. Nas duas próximas seções deste capítulo serão apresentadas técnicas de pré-processamento e diferentes formas de representação de informações para a AS.

2.2.1 Pré-processamento de dados

Antes de realizar o processo de classificação, uma das principais etapas para aquisição de um modelo preditivo adequado refere-se ao processo de limpeza de dados e seleção de atributos. A fase de pré-processamento do texto é aquela onde um arquivo de texto cru, normalmente apresentado como uma sequência de bits digitais é reconhecido como uma sequência bem definida de unidades linguisticamente importantes. No mais baixo nível, caracteres representando os grafemas do sistema de escrita da linguagem; evoluindo, unidades lexicais consistindo de um ou mais caracteres; por fim, sentenças contendo uma ou mais palavras [34]. De modo geral, o processo de seleção de atributos caracteriza-se pela identificação de um subconjunto de atributos relevantes à classificação. Além de ser importante em diversas tarefas de mineração de dados, a seleção de atributos torna-se necessária em tarefas de classificação textual, uma vez que esse tipo de cenário pode gerar um grande volume de atributos, muitos deles irrelevantes para o processo preditivo [32].

A tarefa de pré-processamento basicamente consiste em uma série de rotinas, processos e métodos necessários para melhorar a qualidade dos dados e alcançar melhores resultados preditivos. Esta etapa é fundamental para as tarefas de PLN, uma vez que as unidades resultantes serão utilizadas em todas as etapas subsequentes do processo. Na mineração de textos, o pré-processamento é ainda mais importante, pois a maioria das técnicas de aprendizado não processam textos em sua forma original, sendo necessário convertê-los em uma representação que seja aceita por esses algoritmos [33]. Nesta etapa o objetivo é remover ruídos e elementos textuais irrelevantes objetivando melhorar a qualidade dos dados para a classificação.

As estratégias empregadas no pré-processamento de textos dependem do tipo de aplicação e do nível de análise que se deseja realizar. Uma diferença importante entre os textos de mídias sociais e mídias tradicionais, como jornais e notícias, é a variação na qualidade do conteúdo. O principal desafio associado ao processamento de conteúdo proveniente de mídias sociais é a grande variação da qualidade da escrita: alguns textos possuem estrutura formal, seguindo a norma culta da linguagem; e outros possuem baixa qualidade, tanto na sua estrutura quanto na gramática, por vezes utilizando termos difíceis para interpretação. Este tipo de escrita faz com que as tarefas de tratamento e classificação sejam mais complexas do que ambientes formais [35].

- **Filtragem:** esta técnica consiste em utilizar expressões regulares para eliminar elementos textuais que não são importantes na mineração de opinião. Os elementos são filtrados considerando as características da aplicação. Nesta etapa ocorre a eliminação das *stopwords*, que são os termos que não acrescentam à representatividade da coleção, ou que sozinhos nada significam. Artigos, pronomes e advérbios são exemplos de palavras que são eliminadas nesta etapa. O conjunto de *stopwords* é conhecido como *stoplist*. A grande vantagem da eliminação dos termos irrelevantes é a redução de 40 a 50% da extensão dos documentos [36]. Joachims [37] sugere a remoção de termos frequentes em mais de 80% dos documentos ou com frequência menor ou igual a três. Conforme Manning et al. [38], essa eliminação reduz significativamente a quantidade de termos, diminuindo o custo computacional das próximas etapas da análise.

  Alguns estudos disponibilizam listas de *stoplist* que podem ser utilizadas na criação de mecanismos para aplicação nesta etapa do pré-processamento. A lista mais utilizada, a qual é recomendada pelo Laboratório de Recuperação de Informação de Massachusetts em Amherst, pode ser encontrada no trabalho de Lewis [39]. Esta lista foi criada para utilização na língua inglesa e é composta por 266 termos. Para a língua portuguesa pode-se utilizar a lista disponibilizada no trabalho de Neves [40].

- **Tokenização:** utiliza regras e expressões regulares para dividir o texto em *tokens*, que são unidades básicas de uma linguagem. Normalmente esses *tokens* são palavras, pontuações e outros marcadores utilizados na linguagem.

  O tratamento da pontuação tem diferentes enfoques - enquanto deseja-se manter os limites das sentenças, normalmente as marcas de pontuação internas da sentença são descartadas

[41]. A especificação do que deve ser considerado como uma palavra (*token*) é complexa; a principal informação utilizada é a ocorrência de um espaço em branco - espaço, tabulação ou o início de uma nova linha - mas mesmo este sinal não é necessariamente confiável [41]. Essa segmentação possui como regra geral a utilização do conjunto "ponto final – espaço em branco – letra maiúscula" [42]. Diversas abordagens são utilizadas com o objetivo de desambiguar a ocorrência do ponto final, aumentando a quantidade das unidades lexicais analisadas antes e após o ponto. Palmer-Hearst [43] desenvolveram um sistema com esse pressuposto, o qual atingiu o expressivo resultado de 99% de acerto com documentos não estruturados do Wall Street Journal.

As idiossincrasias, tais como a utilização do ponto final para determinar a parte fracionária de um número, ou uma abreviação, por exemplo, devem ser contornadas para uma correta unitização do texto. Goldsmith [44] apresenta diversos métodos para segmentação de texto em *tokens*, mesmo para as linguagens onde o espaço em branco não é o delimitador clássico, como o chinês e japonês, ou naquelas onde, em alguns casos, não existe o delimitador. Lista, ainda, várias abordagens para aprendizado morfológico não supervisionado, ou seja, sem a necessidade de *corpus* anotado para treinamento prévio do algoritmo.

- ***Speller*:** realiza a correção ortográfica das palavras, auxiliando no desempenho de outras ferramentas de processamento, tais como *Parsers*[6] e *Taggers*[7]. O principal objetivo desta etapa é identificar os erros cometidos quanto à ortografia (palavras que não constam do léxico dessa língua ou usadas em contexto impróprio), para então sugerir alternativas prováveis e ortograficamente corretas a cada erro identificado. Ferramentas de correção são bastante empregadas em aplicações que consideram dados de redes sociais, pois erros gramaticais e ortográficos são muito comuns e tendem a aumentar a dimensionalidade dos atributos.

  O CoGrOO[8] é um exemplo de corretor ortográfico, o qual foi desenvolvido por alunos da Universidade de São Paulo, para a língua portuguesa. A ferramenta realiza a correção ortográfica, detectando erros de grafia nas palavras (por exemplo, a palavra "sugeito" com a letra "g", em vez de "sujeito" com a letra "j"), e também a correção gramatical, detectando erros nas relações entre as palavras (por exemplo, na sentença "Os menino estudam demais.", é detectado o erro na concordância entre o artigo "os" e o substantivo "menino"). Para a língua inglesa, um software muito difundido e utilizado para a correção gramatical é o OpenNLP[9].

[6] Ferramenta utilizada para realizar a verificação semântica e sintática das palavras de um determinado conjunto de texto. Normalmente a estrutura resultante deste *parsing* é uma árvore, a qual apresenta apresenta as associações entre as unidades (palavras) presentes neste texto, bem como as relações funcionais que unem estes grupos de unidades.

[7] Ferramenta utilizada para anotar a classe gramatical de um determinado conjunto de texto.

[8] Disponível em "http://cogroo.sourceforge.net/".

[9] Biblioteca para a linguagem de programação Java, baseada em Aprendizado de Máquina, e desenvolvida para o processamento de textos. Suporta tarefas como tokenização, segmentação de sentença, etiquetagem morfo-sintática, extração de entidades nomeadas, extração de sintagmas, análise sintática e resolução de correferência, dentre outras. Esta ferramenta está disponível em "http://opennlp.apache.org/".

- ***Stemmer*:** forma de normalização textual que visa a redução das palavras à sua raiz (*stem*). Por exemplo, "Jardim", "Jardineiro" e "Jardinagem" são reduzidos para "Jardi". Este tipo de técnica reduz a dimensionalidade dos dados, pois sua aplicação garante que diferentes palavras sejam mapeadas a seu tronco morfológico primitivo. A aplicação desta técnica não só reduz o número de entradas, como também aumenta a ocorrência do radical nos documentos [3]. Entretanto, a aplicação de técnicas de *stemming* geram uma redução na precisão das buscas, já que não se é possível verificar a ocorrência de determinada palavra nos documentos. Em tarefas de classificação de documentos estas variações morfológicas são importantes, pois aumentam o poder de discriminação entre os documentos.

  Os algoritmos de eliminação da variação morfológica mais conhecidos para a língua inglesa são Porter [45], Lovins [46] e Paice [47]. O algoritmo Porter tem sido muito utilizado nos últimos anos, sendo possível encontrar em sua página oficial[10] diversas implementações em diferentes linguagens de programação. Para a língua portuguesa existem dois algoritmos bastante utilizados: Snowball[11] e Orengo [48]. Apesar destes algoritmos serem aplicados com o intuito de melhorar o desempenho de sistemas de Recuperação de Informações e Mineração de Textos, alguns estudos [49, 50] mostraram que a redução de afixos em uma coleção de documentos não apresenta uma melhora significativa no desempenho da tarefa de mineração aplicada, realizando somente uma redução no espaço de representação de dados.

  Além disso, ao realizar *stemming* deve-se ter cuidado com *overstemming* e *understemming*. *Overstemming* ocorre quando a cadeia de caracteres extraída não é um sufixo, mas sim parte do radical. Por exemplo, a palavra "gramática", após o processamento, reduz-se para "grama", o que não representa o seu radical, que é "gramat". *Understemming* ocorre quando o sufixo não é removido totalmente. Por exemplo, a palavra "referência", após o processamento reduz para "referênc", ao invés de "refer", que é o radical correto [51].

- **Lematização:** o processo de lematização consistem em substituir as palavras pela sua forma canônica, sendo o singular masculino em substantivos e o infinitivo em verbos. Exemplificando, as palavras "gato", "gata", "gatos" e "gatas" são todas formas do mesmo lema "gato". Igualmente, as palavras "tiver", "tenho", "tinha" e "terei" são do mesmo lema "ter". Assim como o *Stemmer*, a lematização auxilia na redução da dimensionalidade dos atributos.

  Bird et al. [52] explicam como realizar essa tarefa utilizando a biblioteca de programação NLTK[12], para qualquer idioma com base de treinamento disponível. Erjavec e Dzeroski [53] desenvolveram um método estatístico para lematização de unidades lexicais para o idioma esloveno. Para palavras conhecidas, o *framework* reportou 98.6% de acurácia. Para pala-

[10] Disponível em "http://tartarus.org/martin/PorterStem.mer/".

[11] Disponível em "http://snowballstem..org/".

[12] NLTK (*Natural Language Toolkit*) é uma biblioteca aberta de desenvolvimento para Python, criada em 2001 como parte de um curso no Departamento de Ciência da Computação e Informação, da Universidade da Pensilvânia, visando facilitar aplicações de PLN. Desde então ele tem sido suportado e expandido, servindo de base para vários projetos. Esta biblioteca contém uma grande quantidade de código, dados e documentação, todos disponíveis em "http://nltk.org/".

vras não conhecidas o resultado foi de 92% de efetividade, o que é um resultado bastante significativo.

Após apresentadas as principais estratégias de pré-processamento que são aplicadas em tarefas de PLN, a Tabela 2.1 apresenta um comparativo sobre a utilização destas em diferentes trabalhos.

Tabela 2.1 – Uso de diferentes técnicas de pré-processamento de dados em tarefas de PLN.

Trabalho	Ano	Fonte	Domínio	Técnica
Grefenstette [54]	1993	Enciclopédia Grolier	Genérico	*Speller* Filtragem
Ciaramita [55]	2002	Jornal Wall Street	Genérico	*Speller* Tokenização Filtragem
Pang-Lee [16]	2002	IMDb	*Review* de filmes	Filtragem Tokenização
Pekar [56]	2004	British National Corpus	Genérico	Filtragem Lematização *Stemming*
Go et al. [57]	2009	Twitter	Genérico	Filtragem Tokenização
Pak-Paroubek [58]	2010	Twitter	Genérico	Filtragem Tokenização
Wang et al. [20]	2012	Twitter	Eleições	Filtragem Tokenização

2.2.2 Tipos de Representação de Informações

Na literatura, diversos modelos são empregados para representar dados não estruturados em tarefas de classificação de sentimento [59, 60, 16]. Em abordagens de AM [61, 16], a definição dos modelos é a chave central para se obter resultados de qualidade na predição dos dados. Genericamente, existem duas maneiras de se representar informações para algoritmos de aprendizado de máquina: *representação local* e *representação distribuída*. Estes dois tipos de representação serão apresentados a seguir. Esta seção será encerrada com a apresentação da Tabela 2.3, trazendo um comparativo entre diversos trabalhos que utilizaram diferentes formas de representação de informações, bem como algumas outras particularidades destes.

- **Representação Local:** Neste tipo de representação cada elemento é representado por dimensões específicas para descrever cada elemento do domínio, resultando assim em uma representação esparsa. Neste esquema utiliza-se um vocabulário com tamanho *n* contendo todos os elementos que deseja-se mapear. O tamanho desse vocabulário varia de acordo com o *corpus* ou linguagem utilizada. Abaixo são apresentadas diferentes estratégias utilizadas na montagem deste tipo de representação:

  1. ***N*-gramas**: um *n*-grama é uma unidade formada por uma sequência de *n tokens*, os quais podem ser caracteres, letras ou palavras. Este tipo de representação pode ter um

bom resultado na tarefa de classificar opiniões pois, através deste método, é possível incorporar à análise um determinado contexto de palavras. Muitas vezes este tipo de representação é utilizado como parte do conjunto de atributos utilizados em tarefas de classificação [16]. Abbasi [62] propõe a utilização de *n*-gramas para diferentes bases como palavras, letras, dígitos e etiquetas morfossintáticas, mostrando que este tipo de atributo é bastante informativo, e também importante, na captura de estilos de linguagem em textos positivos e negativos. É comum a utilização de uma representação híbrida de *n*-gramas, a qual pode conter unigramas (*tokens* únicos), bigramas (conjunto de dois *tokens*) e trigramas (conjunto de três *tokens*), ou ainda outra combinação de *tokens*. Na Tabela 2.2 são apresentados exemplos de representação *n*-grama a nível de palavra e caractere.

Tabela 2.2 – Exemplos de estruturas *n*-gramas para a sentença "*ser ou não ser*".

Estrutura	Palavra	Caractere
Unigramas	ser, ou, não	s, e, r, o, u, n, ã, o, _
Bigramas	ser ou, ou não, não ser	se, er, r_, _o, ou, _s, u_, _n, nã, ão, o_
Trigramas	ser ou não, ou não ser	_se, ser, er_, r_o, _ou, ou_, u_n, _nã, não, ão_, o_s

Uma das dificuldades quando utiliza-se *n*-gramas como representação é a definição do número de *tokens* que represente bem as informações das sentenças que serão analisadas. Normalmente utiliza-se bigramas quando o conteúdo a ser analisado é pequeno e trigramas quando o conjunto é grande (com milhões de palavras).

Pang et al. [16] reportaram que unigramas superam o resultado de bigramas em tarefas de AS. Neste trabalho, a utilização de unigramas alcançou um resultado de 82.9% de acurácia, ao passo que a utilização de bigramas resultou em uma acurácia de 77.1%, ambos os testes utilizando uma base de *review* de filmes, SVM como classificador. Contrariando os resultados apresentados por Pang et al. [16], Dave et al. [63] comprovam que, sob algumas configurações e aplicando algumas técnicas no trato dos dados, bigramas e trigramas atingem um melhor resultado do que unigramas. Neste experimento foi utilizado SVM para realizar a classificação de *reviews* de produtos, provenientes do site da Amazon[13] e CNET[14]. O resultado reportado foi uma acurácia de 87.2% com a utilização de bigramas e 81.1% com unigramas, ambos aplicando o algoritmo SVM como classificador. Pesquisas anteriores, como Pang e Lee [10] e Wiebe et al. [64], mostram que a utilização de 4-gramas ou mais é redundante, causando ruído e diminuição da eficiência do processo.

Outro ponto que pode ser explorado quando da montagem de estruturas *n*-gramas é a influência da posição em que a palavra se encontra em uma sentença. Mukherjee e Bhattacharyya [65] sugerem que palavras que aparecem em certas posições no texto

[13] http://www.amazon.com/
[14] http://www.cnet.com/

carregam mais sentimentos ou peso do que palavras que aparecem em outras posições. Esta afirmação apoia-se no processo de Recuperação de Informação, onde as palavras que aparecem em determinados tópicos, tais como títulos e resumos, têm um peso maior do que aquelas que aparecem no corpo do documento. Um exemplo que ilustra esta situação de influência no sentimento pode ser verificado no texto abaixo:

> *"Nosso passeio deveria ter sido bom! Reservamos um hotel muito bonito e agradável, com uma linda vista e com um atendimento aceitável, e ainda com uma comida que parecia ser deliciosa. A distância entre o hotel e a cidade era curta, então estaríamos bem dispostos para conhecer o lugar. Porém a viagem foi péssima, cansativa e estressante!"*

No exemplo acima, mesmo o texto contendo muitas palavras positivas, presentes em quase todo o texto, o sentimento geral pode ser classificado como negativo. Isto ocorre porque as últimas palavras presentes no texto são cruciais na definição do sentimento do mesmo. De maneira geral, palavras pertencentes às *n* primeiras sentenças e às *m* últimas sentenças em um texto têm um peso crucial na determinação do sentimento. Pang e Lee [16] e Kim e Hovy [66] são exemplos de trabalhos que aplicaram esta ponderação espacial na codificação dos vetores de atributos.

2. ***Bag-of-words*:** o nome deste modelo praticamente explica como esta representação funciona, ou seja, a representação é criada como uma "sacola" de palavras, não considerando nenhuma relação semântica entre as palavras, somente a sua ocorrência ou não no texto. Segundo Bifet e Frank [61], o texto passa a ser um conjunto de palavras sem levar em consideração a ordem em que estas palavras ocorrem no texto.

   Nesta representação, cada palavra é colocada unicamente nesta "sacola", e em seguida, para cada sentença, deve-se percorrer este vetor de palavras, colocando 1 quando esta palavra está presente na sentença e 0 quando está ausente, conforme apresentado na Figura 2.6(a). Outro modo de montar esta estrutura é utilizar um contador para cada palavra, ou seja, em cada sentença deve-se contar quantas vezes cada palavra aparece, conforme apresentado na Figura 2.6(b). Para montar os exemplos de representação mostrados na Figura 2.6 foram utilizadas as três sentenças abaixo:

   (a) *"Meu carro é azul e meu avião é vermelho."*

   (b) *"Comprei meu carro vermelho e meu carro azul ontem."*

   (c) *"Ontem choveu."*

   Na literatura, este é um dos modelos mais utilizados para representação de palavras [67, 68, 69, 70]. Entretanto, este tipo de representação avalia somente a ocorrência ou não de uma palavra no texto, ou seja, não considera nenhuma relação semântica entre elas, analisando somente a presença ou não das palavras no texto. Assim sendo, frases que possuem significados diferentes podem ter a mesma representação se as mesmas palavras forem utilizadas. Este tipo de problema pode ser identificado nas sentenças abaixo:

Sentença / palavra	meu	carro	é	azul	ontem	e	comprei	choveu	avião	vermelho
Sentença 1	1	1	1	1	0	1	0	0	1	1
Sentença 2	1	1	0	1	1	1	1	0	0	1
Sentença 3	0	0	0	0	1	0	0	1	0	0

(a) *Bag-of-words* por presença.

Sentença / palavra	meu	carro	é	azul	ontem	e	comprei	choveu	avião	vermelho
Sentença 1	2	1	1	1	0	1	0	0	1	1
Sentença 2	2	2	0	1	1	1	1	0	0	1
Sentença 3	0	0	0	0	1	0	0	1	0	0

(b) *Bag-of-words* por frequência.

Figura 2.6 – Exemplo de *bag-of-words* por presença e *bag-of-words* por frequência.

(a) *"Contrariando o surreal João, este filme é estranho mas adorável."*

(b) *"Contrariando o surreal mas adorável João, este filme é estranho."*

O sentimento geral da primeira frase pode ser considerado positivo, ao passo que a segunda frase apresenta conotações de sentimento negativo. Abordagens baseadas em *bag-of-words* não fariam esta distinção, pois o conjunto de palavras seria o mesmo para ambas as sentenças e as duas frases seriam classificadas com a mesma polaridade.

3. **Representação a nível de caractere:** este é o tipo de abordagem com o mais baixo nível possível, ou seja, o aprendizado é feito utilizando as menores partículas que são utilizada na montagem de textos ou sentenças: as letras, numerais e pontuação. Este modelo utiliza um alfabeto de tamanho *m* para a linguagem de entrada, e então cada caractere é quantificado na forma de 1-para-*m*. Conforme apresentado por Zhang et al. [71], este tipo de representação não considera nenhum conhecimento sobre palavras, frases e sentenças ou qualquer outra estrutura semântica ou sintática associadas à linguagem. Este tipo de representação é particularmente útil quando se deseja realizar tarefas que cunho sintático, pois os padrões encontrado nos caracteres das palavras estão mais relacionados à sua função sintática [72]. A Figura 2.7 apresenta um exemplo deste tipo de representação.

   A principal evidência que motiva a utilização deste tipo de representação em tarefas de PLN é a facilidade que este modelo tem de lidar com o problema de "palavras desconhecidas", o qual é enfrentado pelas demais representações que baseiam-se em palavras. Basicamente, este problema refere-se à aparição de palavras que não estão mapeadas no vocabulário que se está analisando. Representações a nível de caractere possuem facilidades ao contornar este problema, pois o vocabulário possível de caracteres é minúsculo

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	.
m	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

Figura 2.7 – Representação a nível de caractere para a sentença "*meu carro azul.*".

se comparado ao universo possível de palavras. Entretanto, a montagem de estruturas neste nível é bastante desafiadora, pois as relações entre formas (caracteres) e funções (sintática e semântica) na linguagem podem ser bastante obscuras [72].

- **Representação Distribuída:** Nesta forma de representação, cada palavra é representada por um vetor de valores contínuos. O tamanho (número de dimensões) destes vetores utilizados para representar as palavras é um hiper parâmetro a ser definido. Esta estratégia possibilita que o vocabulário seja representado inteiramente por uma matriz que é calculada anteriormente (ou durante) à execução do algoritmo em questão. Deste modo, é possível selecionar a representação distribuída de cada palavra a partir do produto interno entre a sua representação local e essa matriz.

  1. ***Word-Embeddings:*** a representação conhecida como *word-embeddings* consiste basicamente em representar uma palavra através de um vetor de pesos em um espaço *n*-dimensional, criando assim uma representação densa da mesma. Além disso, palavras que aparecem em contextos semelhantes tendem a se concentrar em uma mesma região espaço-vetorial. Ainda, esta representação possibilita a aplicação de álgebra vetorial simples para buscar relações entre determinadas palavras.

     Os valores utilizados na representação de cada vetor de palavra são inicializados de forma aleatória. No metódo *word2vec* [73], por exemplo, os vetores de cada palavras são randomicamente inicializados com valores entre $[-\frac{1}{2d}, \frac{1}{2d}]$, onde d representa o número de dimensões utilizados na representação.

     A matriz que representa o vocabulário é calculada em uma etapa de treinamento não-supervisionado, onde um determinado número de atributos (nº de dimensões) é aprendido. Segundo Collobert [74], esses atributos aprendidos correlacionam as palavras através de propriedades semânticas e sintáticas. Neste treinamento basicamente busca-se

maximizar o *log* da verossimilhança da probabilidade de uma determinada palavra dado o seu contexto local, segundo a Fórmula 2.3. Uma vez aprendida, essa matriz pode ser refinada durante o aprendizado da tarefa em questão através do algoritmo *backpropagation*.

$$L(\theta_1, \theta_2) = \sum_{i=1}^{|NP|} \log P(w_i | w_i^{contexto}) \tag{2.3}$$

onde $w_i^{contexto}$ denota o contexto local da palavra w_i, $|NP|$ é o número de palavras de todo o conjunto de dados e θ_1 and θ_2 são, respectivamente, a representação *word-embedding* e os parâmetros na RNA [75]. O modelo mais difundido deste tipo de representação é o *word2vec* [73], o qual vem atraindo a atenção da comunidade acadêmica [76, 77].

Após apresentadas as principais formas de se representar informações para tarefas de PLN, a Figura 2.8 exemplifica de forma simplificada as diferenças entre representação local e distribuída, ao passo que a Tabela 2.3 apresenta um comparativo sobre a utilização destas representações em diferentes trabalhos.

Representação Local

	carro	azul	meu
meu	0	0	1
carro	1	0	0
azul	0	1	0

Representação Distribuída

	D1	D2	D3	D4
meu	0.5	1.9	0.1	0.8
carro	1.4	0.3	0.2	1.2
azul	0.6	1.0	1.9	0.0

Figura 2.8 – Comparativo entre Representação Local (BOW) e Distribuída (nº de dimensões definido aleatoriamente) para a sentença "*meu carro azul*". No exemplo, destaca-se o vetor utilizado para representar a palavra "azul".

2.3 Redes Neurais Artificiais

Redes Neurais são estruturas computacionais que procuram simular ou reproduzir em pequena escala o funcionamento do cérebro humano [80]. Este tipo de algoritmo pretende simular o sistema nervoso, o qual contém um conjunto de neurônios que se comunicam através de axônios e dendritos. Os dendritos são ramificações que recebem sinais elétricos de outros neurônios, que são transmitidos pelo axônio. A ligação entre um axônio e um dendrito é conhecida como sinapse. As sinapses são, portanto, as unidades que medeiam as interações entre os neurônios [80], podendo estas ser excitatórias ou inibitórias.

Conforme apresentado em [81], as Redes Neurais Artificiais (RNA ou RN) são sistemas computacionais distribuídos compostos de unidades (neurônios) de processamento simples, densa-

Tabela 2.3 – Uso de diferentes tipos de representação de informações em experimentos de AS.

Trabalho	Ano	Classificador	Fonte	Domínio	Representação
Pang e Lee [16]	2002	NB ME SVM	IMDb	*Review* de filmes	Unigramas Bigramas Adjetivos POS
Go et al. [57]	2009	NB ME SVM	Twitter	Genérico	Unigramas Bigramas POS
Pak e Paroubek [58]	2010	MNB	Twitter	Genérico	Unigramas Bigramas
Wang et al. [20]	2012	NB	Twitter	Eleições	Unigramas
Smailović et al. [78]	2014	SVM	Twitter	Genérico	Bigramas
Kim [79]	2014	CNN	Amazon e RottenTomatoes	*Review* de produtos e filmes	Word-Embedding
Barnaghi et al. [22]	2015	BLR	Twitter	Copa do Mundo 2014	Unigramas Bigramas
Severyn e Moschitti [1]	2015	CNN	Semeval 2015	Genérico	Unigramas
Zhang et al. [71]	2015	CNN	Amazon	*Review* de produtos	Caractere
Esta Dissertação	**2017**	**CNN**	**Twitter**	**Genérico**	**Caractere**

mente interconectadas. Estes neurônios, os quais computam funções matemáticas, estão dispostos em uma ou mais camadas, interligados por um grande número de conexões. Na maioria das arquiteturas, estas conexões, que simulam as sinapses biológicas, possuem pesos associados, os quais ponderam a entrada recebida por cada neurônio da rede. Os pesos podem assumir valores positivos ou negativos, dependendo do comportamento da conexão ser excitatório ou inibitório. Os pesos têm seu valores ajustados em um processo de aprendizado e codificam o conhecimento adquirido pela rede [82].

Geralmente, uma Rede Neural básica possui a seguinte estrutura:

- Uma camada de entrada, composta por várias unidades, de acordo com o número de atributos (*features*) para representar os dados;
- Uma ou mais camadas intermediárias, compostas por neurônios responsáveis pela definição das relações não lineares entre as unidades de entrada e saída;
- Uma camada de saída que fornece a resposta do sistema;
- Ligações entre as várias camadas, responsáveis pela propagação dos sinais entre as camadas.

Esta arquitetura pode ser identificada na Figura 2.9, onde são apresentados à rede três valores de entrada [x1, x2, x3], uma camada intermediária contendo quatro neurônios [h1, h2, h3, h4], as conexões [w] que ligam a camada intermediária às entradas e à saída, finalizando com uma unidade de saída [y].

O neurônio pode ser considerado o elemento processador de uma Rede Neural. Cada neurônio gera uma saída com base na combinação dos sinais de entrada recebidos dos neurônios os quais este está conectado. A saída de um neurônio é, na maio parte dos modelos, o resultado de uma função de ativação aplicada a soma ponderada dos dos valores de entrada.

A Figura 2.10 apresenta uma arquitetura simples de um neurônio artificial. Na imagem, os valores de entrada x sofrem o estímulo dos pesos w. Em seguida, os valores resultantes desta

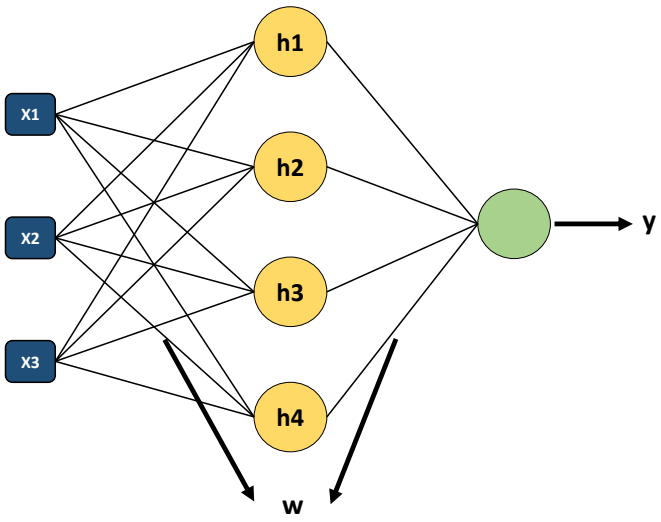


Figura 2.9 – Estrutura básica de uma Rede Neural.

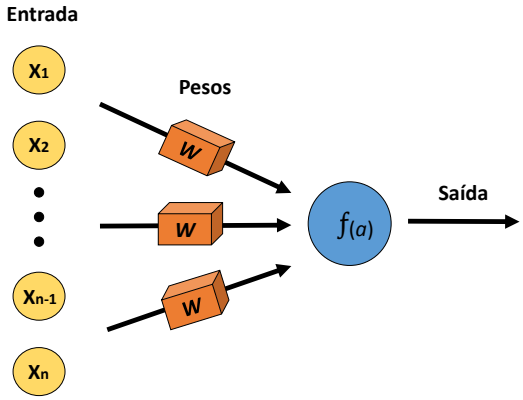


Figura 2.10 – Neurônio artificial.

combinação são computados conforme a Fórmula 2.4 e o resultado é aplicado em uma função de ativação f(a).

$$a = \sum_{i=1}^{x} x_i w_i \tag{2.4}$$

Conforme apresentado por Haykin [83], existem três tipos de funções de ativação que são mais utilizadas: Função Linear em partes (Eq. 2.5), Função Limiar ou Sinal (Eq. 2.6) e Função Sigmóide (Eq. 2.7).

- **Função Linear em partes:** Este tipo de função pode ser considerado uma aproximação de um amplificador não-linear, conforme apresentado na Figura 2.11(a).

$$f(a) = \begin{cases} 1 & \text{se } a \geq \frac{1}{2} \\ a & \text{se } \frac{1}{2} > a > -\frac{1}{2} \\ 0 & \text{se } a \leq -\frac{1}{2} \end{cases} \tag{2.5}$$

- **Função Limiar (Sinal):** Baseia-se na propriedade tudo-ou-nada, proposta por McCullock e Pitts [84], possuindo uma fronteira de decisão abrupta. A saída do neurônio assume valor 0 quando o seu resultado for negativo e 1 caso contrário. A Figura 2.11(b) apresenta o gráfico desta função.

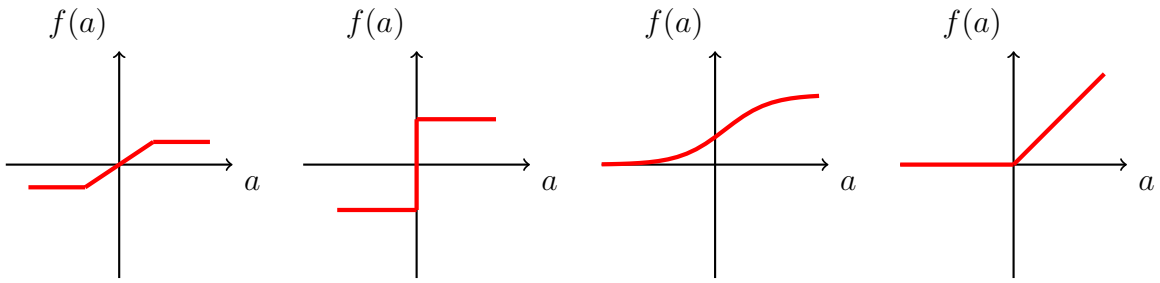
$$f(a) = \begin{cases} 1 & \text{se } a \geq 0 \\ 0 & \text{se } a < 0 \end{cases} \tag{2.6}$$

- **Função Sigmóide:** É uma das funções de ativação mais utilizada na construção da RNA. Possui uma forma de "s", monotonamente crescente, que exibe um balanceamento adequando entre comportamento linear e não-linear. Este tipo de função representa uma aproximação contínua e diferenciável da função limiar. O gráfico desta função pode ser visualizado na Figura 2.11(c).

$$f(a) = \frac{1}{1 + e^{-a}} \tag{2.7}$$

- **Função ReLU:** Proposta por Nair e Hinton [85], esta função de ativação tornou-se a mais utilizada em arquiteturas modernas de RNA. Basicamente, a ReLU zera entradas inferiores a zero e retorna o mesmo valor caso receba uma entrada positiva, conforme apresenta a Figura 2.11(d).

$$f(a) = \begin{cases} 0 & \text{se } a \leq 0 \\ a & \text{se } a > 0 \end{cases} \tag{2.8}$$



(a) Linear em partes (b) Limiar (c) Sigmóide (d) ReLU

Figura 2.11 – Exemplos de funções de ativação de um neurônio.

2.3.1 Perceptron e *Multi-layer* Perceptron

O modelo de RNA construído somente com um neurônio, conhecido como Perceptron, possui diversas limitações. Conforme apresentado por Minsky [86], este tipo de algoritmo não consegue classificar dados que não são linearmente separáveis, como o operador XOR. Assim sendo,

desenvolveu-se outra topologia de rede, com mais neurônios e camadas, conhecida como *Multi-layer* Perceptron (MLP), a qual consegue lidar com dados não separáveis linearmente. Conforme apresentado por Hornik et al. [87], uma MLP é um aproximador universal, pois consegue aproximar qualquer função com um certo grau de acurácia[15].

Em uma MLP, a primeira camada é composta pelos valores de entrada da rede, seguida por uma ou mais camadas *escondidas* e finalizando com camada de saída da rede. O modelo mais simples de uma MLP consiste em: cada camada l é totalmente conectada com as camadas l-1 e l+1, produzindo um vetor de saída $y^{(l)}$, dado o vetor de saída da camada anterior $y^{(l-1)}$. A saída de uma camada é calculado aplicando uma função de ativação para todos os neurônios desta camada, conforme apresentado na Equação 2.4. Alguns autores classificam a camada de entrada como sendo a primeira camada da rede (l_1), entretanto outros consideram que o conjunto de entrada não faz parte da rede (l_0). No decorrer deste trabalho serão apresentadas estruturas onde os dados de entrada não fazem parte da estrutura da rede, ou seja, os dados de entrada localizam-se na camada l_0.

2.3.2 Treinamento

O treinamento de uma RNA busca encontrar o conjunto de pesos que classificarão as instâncias de testes com o menor erro possível. Para isto, um dos algoritmos de aprendizado mais difundido é o *backpropagation* [88], o qual corrige os pesos da rede buscando aproximar o valor de saída e um valor esperado. Basicamente, o treinamento de uma MLP divide-se em duas fases: *Forward* e *Backward*. Conforme Carvalho e Ludemir [89], podemos definir estes passos da seguinte maneira:

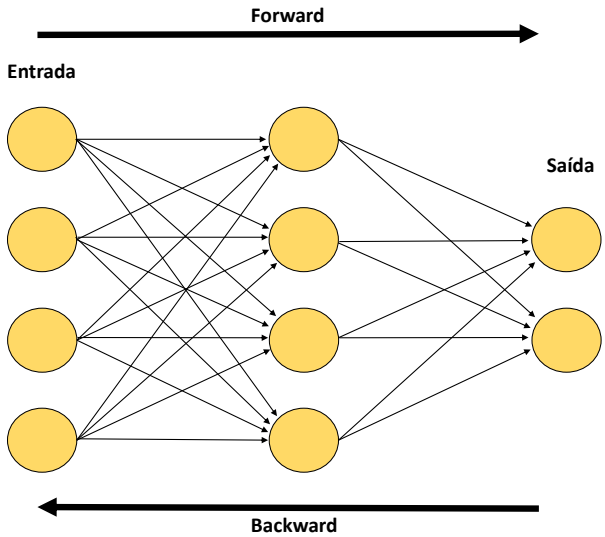


Figura 2.12 – Treinamento de uma MLP com *backpropagation*.

[15] Relação de proximidade entre o resultado esperado e o real valor obtido na execução de um determinado algoritmo.

- **Fase *forward*:**

  1. A entrada é apresentada à primeira camada da rede (l_0);
  2. Para cada camada l_i ($i > 0$):
     2.1. Os valores de saída do neurônio da camada l_i irão alimentar a entrada da camada l_{i+1}, ou seja, serão os valores de entrada para a próxima camada;
  3. Os valores provenientes da última camada serão comparados com os valores desejados;

- **Fase *backward*:**

  1. Partindo da última camada (l_n) da rede até a primeira (l_0):
     1.1. Os neurônios artificiais da camada l_a (camada atual) devem ajustar seus pesos a fim de reduzir o erro;
     1.2. O erro de um neurônio das camadas intermediárias ($l_{1,n-1}$), será calculado utilizando os erros retropropagados dos neurônios pertencentes às camadas seguintes, os quais estão conectados a ele, ponderando-se pelo peso das conexões entre eles;

  O *backpropagation* procura minimizar o erro obtido pela rede realizando ajustes nos pesos e limiares para que estes atendam às coordenadas de pontos mais baixos da superfície de erro [89]. A diminuição do erro de uma rede indica que a mesma está mais próxima da configuração ideal (convergência). Esta tarefa pode ser realizada através do método de gradiente descendente.

- Algoritmo ***backpropagation*:**

  1. Inicializar a rede (pesos e parâmetros);
  2. Repetir até atingir critério de parada[16]:
     2.1. Para cada instância do conjunto de treino:
        2.1.1. Calcular a saída (S) da rede utilizando a fase *forward*;
        2.1.2. Comparar a saída (S), calculada no passo anterior, com as saídas desejadas;
        2.1.3. Realizar atualização dos pesos utilizando a fase *backward*;

Após o processamento de todas as instâncias de treino e finalizada a atualização dos pesos, conclui-se a execução de uma **época**[17] do treinamento. Um dos diversos parâmetros que devem ser definidos previamente no *backpropagation* é o α (*alpha*). Este parâmetro regula a taxa de atualização dos pesos sinápticos. Caso o valor definido para α seja muito pequeno, o treinamento demorará mais. Ainda, dependendo dos valores iniciais dos pesos, o treinamento poderá ficar preso em um mínimo local. Dentre as diversas variações do *backpropagation*, as mais conhecidas são: Quickprop [90], Levenberg-Marquardt [91], *momentum* de segunda ordem [92] e Rprop [93].

[16] O critério pode ser o atingimento de um erro pré-definido ou a execução do algoritmo por um número pré-definido ciclos.

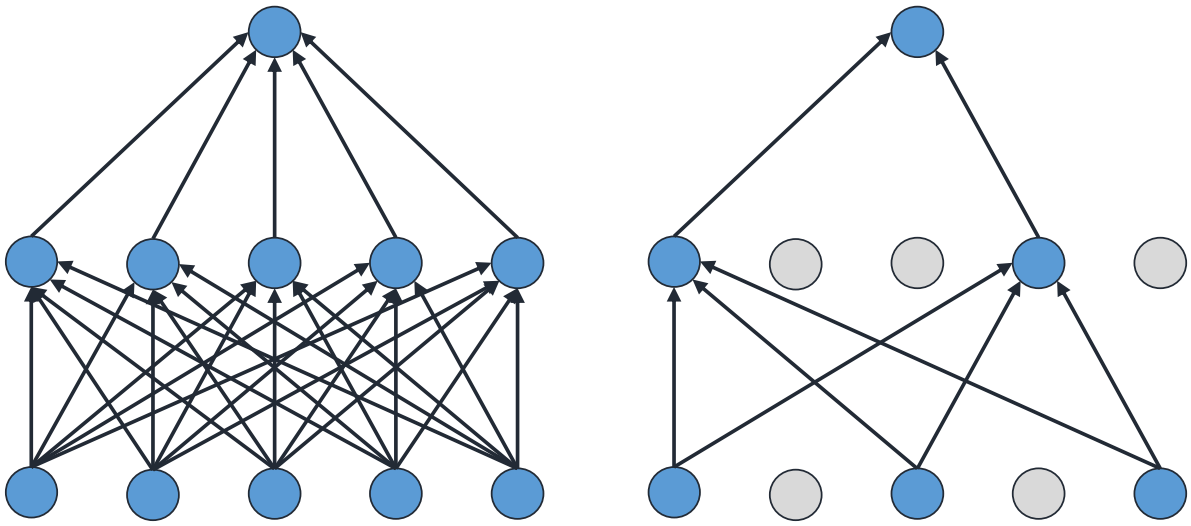
[17] Uma época consiste no intervalo correspondente a apresentação de todos os vetores de treino à camada de entrada de uma MLP.

2.3.3 Regularização

Por décadas, o treinamento de Redes Neurais *profundas*, com muitas camadas *escondidas* e com muitas conexões, poderia não alcançar o resultado esperado. O insucesso no treinamento era frequente pois ocorriam problemas como *overfitting*[18], convergência lenta e outros problemas [94]. Deste modo, diversas técnicas foram desenvolvidas e testadas. Abaixo são apresentadas algumas das principais alternativas para contornar estes problemas.

1. *Dropout*: Recentemente, uma técnica conhecida como *dropout* [95] demonstrou uma melhora significativa em diversas tarefas como a classificação de imagens [96]. Basicamente, uma função *dropout* consiste em definir alguns neurônios para zero, respeitando uma taxa probabilística p (normalmente varia entre 0.25 e 0.50, ou pode ter valores ainda mais agressivos como 0.75 ou 0.80) para que isto ocorra, durante a fase de treinamento de uma RNA. Na fase de teste, esta operação não ocorre e o valor de cada neurônio é calculado normalmente.

   A aplicação desta desta técnica cria a ideia de treinar diversas sub-arquiteturas derivadas da arquitetura original. Ao desligar um neurônio, todas as suas conexões também são temporariamente desligadas, sejam estas de saída ou entrada. Desta forma, a informação pode percorrer diferentes segmentos (neurônios e suas conexões) da RNA durante a fase de treinamento. Na Figura 2.13 é possível visualizar um exemplo de como funcionaria esta técnica em uma RNA.



(a) Funcionamento de uma Rede Neural sem aplicação de *dropout*.

(b) Funcionamento de uma Rede Neural com aplicação de *dropout*.

Figura 2.13 – Funcionamento da técnica *dropout* em uma Rede Neural com uma camada escondida. Os neurônios azuis estão ativos, ao passo que os neurônios cinzas representam os inativados pela função.

[18] Este problema ocorre quando o modelo fica muito ajustado ao conjunto de treino, passando a não generalizar bem os dados.

Conforme apresentado por Srivastava et al. [97], uma RNA que utiliza uma camada com *dropout* é capaz diminuir substancialmente o erro de generalização numa grande variedade de tarefas de classificação se comparado ao treinamento com outros métodos de regularização. De forma geral, o efeito gerado pela aplicação desta técnica é tornar a rede menos sensível a pesos específicos, fazendo com que esta fique mais suscetível a melhorar seu poder de generalização, evitando assim o *overfitting* dos dados. Este cenário é confirmado no trabalho de Kim [79], onde este demonstra que a aplicação de uma função *dropout* em uma Rede Neural Convolucional pode aumentar o seu desempenho de 2% a 4% sem causar *overfitting*.

2. *ReLU*: Durante muito tempo, a utilização de funções sigmoidais era um consenso no que tange aplicações de RNAs. Porém, Nair e Hinton [85] propuseram o uso de uma alternativa mais simples, a função de ativação retificada linear (Equação 2.8). Assim sendo, os neurônios que usam essa função de ativação são chamados de unidades retificadoras lineares (ReLU, do inglês *Rectified Linear Units*). Basicamente, a ReLU zera entradas inferiores a zero e retorna o mesmo valor caso receba uma entrada positiva. Redes Neurais Convolucionais que utilizam a função ReLU têm um tempo de treinamento reduzido se comparado ao uso da função tangente hiperbólica [96].

Na próxima seção deste capítulo será apresentada uma topologia de RNA que faz parte do conjunto de algoritmos conhecidos como Redes Neurais *profundas*[19].

2.3.4 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Networks*) foram propostas por LeCun et al. [98, 99]. Esta arquitetura, que é uma variação da MLP, foi inspirada no funcionamento do córtex visual, onde os neurônios das regiões iniciais são responsáveis por detectar formas geométricas simples na imagem, como cantos e bordas, ao passo que os neurônios das regiões finais detectam formas gráficas mais complexas. Este processo se repete através das regiões até que os neurônios da região final têm a atribuição de detectar características de mais alto nível de abstração, tais como faces ou objetos específicos [100].

Este tipo de rede é utilizada quando se deseja encontrar informações importantes que estão implícitas em um conjunto de dados, através de operações de Convolução e *Pooling*. O objetivo deste processo é diminuir a quantidade de parâmetros que devem ser aprendidos pelo classificador, pois, conforme apresentado por Hinton et al. [95], esta redução é importante para diminuir a complexidade e o custo do treinamento. A Figura 2.14 exemplifica a arquitetura de uma CNN.

- **Camada Convolucional:** Uma camada de convolução é composta por diversos neurônios, sendo cada um destes responsável por aplicar um filtro em um determinado pedaço da matriz

[19] Redes Neurais *profundas* são diferentes tipos de RNA, as quais normalmente contam com muitas camadas *escondidas* e realizam diferentes operações em diversas partes de sua estrutura.

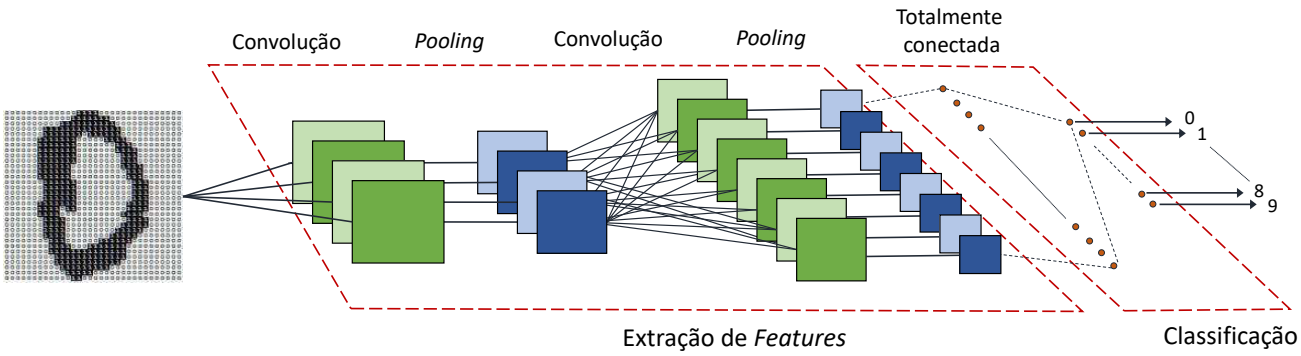


Figura 2.14 – Arquitetura de uma CNN.

de entrada. Uma convolução consiste na aplicação de uma série de "filtros" deslizantes em toda a extensão da matriz de entrada, e o resultado da aplicação destes filtros é conhecido como *feature map* [98]. De modo geral, podemos considerar os parâmetros das camadas convolucionais como um conjunto de n filtros capazes de aprender. Estes filtros, os quais não precisam ser muito grandes, percorrem toda a matriz de entrada, O tamanho de cada deslocamento de um filtro é chamado ***stride***. Diferenciando-se da arquitetura de uma MLP clássica, onde cada neurônio é conectado a todos os neurônios da camada anterior, em uma CNN apenas um subconjunto de entradas é conectado a cada neurônio.

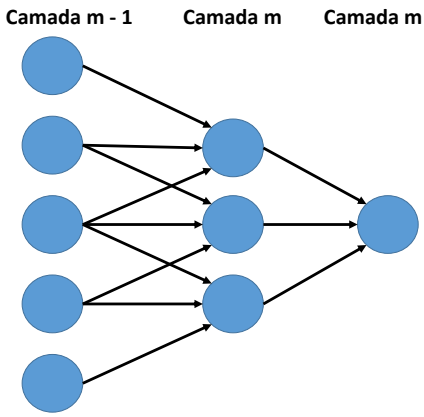


Figura 2.15 – Mapeamento de neurônios em um filtro convolucional.

Conforme mostrado na Figura 2.15, para cada filtro, cada neurônio da camada *m* é conectado somente à um subconjunto de tamanho fixo da camada *m*-1. O valor do neurônio da camada *m* é obtido através da soma de todos os resultados da multiplicação de cada neurônio de *m*-1 coberto pelo filtro e um parâmetro fixo. Logo, cada filtro aplica a mesma matriz de pesos em toda a extensão dos dados de entrada. Os valores dessa matriz de pesos também serão aprendidos durante o treinamento (via *backpropagation*) considerando a média dos gradientes dos pesos compartilhados.

- **Camada *Pooling*:** Uma camada de *pooling* implementa uma função não-linear de subamostragem com o objetivo de diminuir a dimensionalidade e capturar pequenas invariâncias [101].

Basicamente, esta técnica reduz a dimensionalidade de um mapa de característica fornecido como entrada e produz outro mapa de característica, criando assim uma espécie de resumo do primeiro. A cada filtro, é selecionado o maior valor presente (*max-pooling*, vide Equação 2.9) ou então é calculada a média dos valores presentes (*average-pooling*, vide Equação 2.10). A Figura 2.16 ilustra como esta operação ocorre. Ainda, conforme apresentado por Collobert et al. [74], é possível manter somente o maior valor para cada filtro, em um processo intitulado pelos autores como *max-overtime pooling*.

$$g(u) = \ max(u_1, u_2, ..., u_n) \tag{2.9}$$

$$g(u) = \ \frac{\sum_i^n u}{n} \tag{2.10}$$

A aplicação de *pooling* agiliza o treinamento e reforça a robustez da CNN em relação à posição e tamanho das características mais importantes dos dados de treinamento. Assim sendo, mesmo que as informações determinantes na definição da classe de uma instância estejam em uma determinada posição da matriz de entrada ou tenham um tamanho diferente entre as instâncias, a CNN conseguirá capturar estes atributos e classificar corretamente os dados.

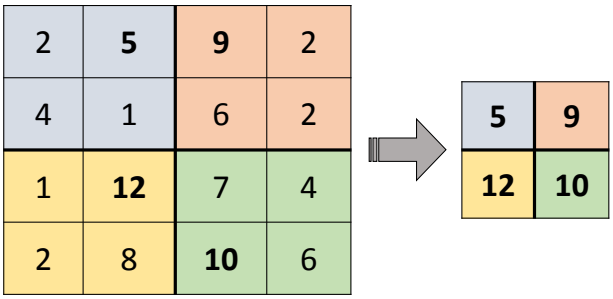


Figura 2.16 – Exemplo de *max-pooling* com deslocamento 2x2 sem sobreposição.

3. ABORDAGEM PROPOSTA

Neste Capítulo serão apresentados os modelos propostos. Primeiramente será introduzida a representação a nível de caractere, que foi o método escolhido para apresentar os dados à CNN. Na sequência serão apresentadas as três arquitetura de CNN propostas, bem como suas parametrizações. Por fim, serão apresentados os trabalhos relacionados. Como não existe nenhuma abordagem que utilize representação a nível de caractere com Redes Neurais Convolucionais para realizar a Análise de Sentimento de dados multi-linguais, foram elencados diferentes trabalhos que, de alguma forma, possuem determinados pontos convergentes com os modelos propostos, seja pela representação, pela tarefa ou pelo algoritmo utilizado.

3.1 Representação de Informações

Já é consenso que uma boa e correta representação dos dados influencia diretamente no resultado da classificação. Desta forma, faz-se necessário desprender um certo tempo na preparação dos dados. A aplicação de uma etapa de pré-processamento de dados visa eliminar possíveis ruídos, os quais podem distorcer a forma como os dados serão estruturados e então apresentados ao algoritmo classificador.

Conforme apresentado na Seção 2.1.3, quando se utiliza dados oriundos de redes sociais, enfrenta-se um problema característico deste tipo de ambiente: o internetês. Em abordagens que usam representações a nível de palavras, normalmente aplica-se uma etapa de pré-processamento. Nesta etapa procura-se eliminar ruídos e deixar o conjunto de dados o mais uniforme possível. Como não se respeita regras na escrita destas mensagens, as palavras podem ser escritas de formas diferentes, seja por abreviaturas ou mesmo erros ortográficos. Desta forma, a mesma palavra pode estar representada de diferentes formas, o que aumenta o espaço dimensional dos atributos.

Para representar os dados na arquitetura de CNN proposta, foi utilizado o tipo de representação onde considera-se as menores partículas utilizadas na formação de palavras e textos: os caracteres. Recentemente este tipo de representação ganhou destaque no trabalho de Zhang et al. [2], o qual será apresentado na Seção 3.3, onde utilizou-se estas estruturas juntamente com Redes Neurais Convolucionais para diferentes tarefas de PLN.

Basicamente, nesta abordagem todos os caracteres $c_i \in \{c_1, c_2, ..., c_n\}$ da mensagem $\mathcal{T}$ são mapeados para uma matriz binária de tamanho $m \times \eta$, onde $\mathcal{V}$ é o alfabeto com n caracteres. A Figura 3.1 apresenta como seria a representação $\vec{2d}$ para a sentença "*I can do it*".

No presente trabalho foram utilizadas mensagens extraídas da rede social Twitter. Uma característica desta plataforma é possuir um limite máximo de 140 caracteres para cada mensagem postada. Assim sendo, quando utiliza-se mensagens do Twitter em abordagens com representação a nível de caractere define-se o tamanho de entrada para $m = 140$. Nas instâncias que possuem um

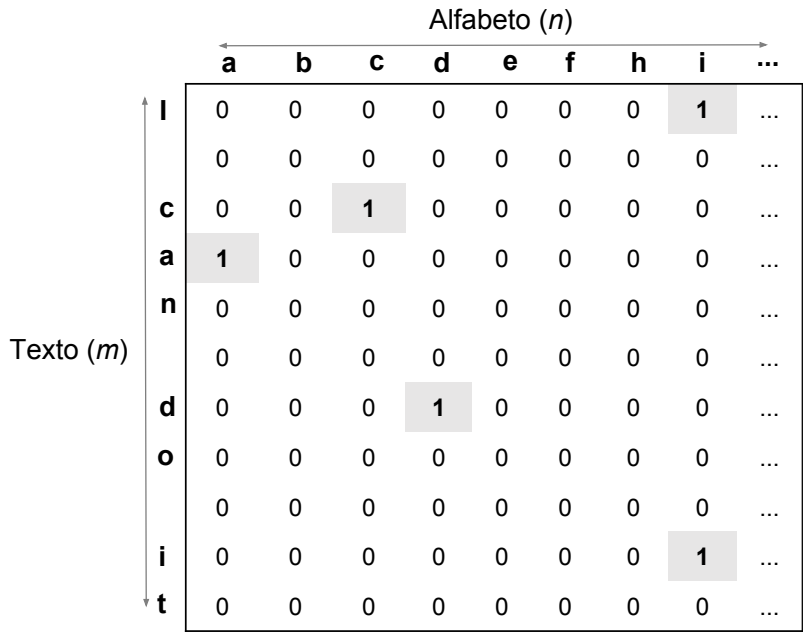


Figura 3.1 – Representação a nível de caractere para a sentença "*I can do it*".

tamanho inferior a 140 caracteres, preenche-se com zeros até que esta atinga o tamanho definido para m.

Uma das vantagens deste tipo de abordagem se comparada a abordagens a nível de palavra, além de não ser necessário realizar uma etapa de pré-processamento de dados, é o tamanho do espaço dimensional dos atributos. Quando opta-se por uma abordagem que utiliza palavras para representação das informações, o tamanho do vocabulário cresce a medida que são adicionadas novas palavras neste conjunto. Ainda, quando deseja-se utilizar dados de diferentes línguas, o tamanho do vocabulário de palavras é ainda maior, fazendo com que sejam necessários mais recursos de hardware e também um tempo maior para treinamento. Se tratando de abordagens a nível de caractere, a quantidade possível de caracteres é drasticamente menor do que a quantidade de possíveis palavras. Em outras palavras, mesmo com um limitado conjunto de caracteres, é possível estruturar um número expressivo de diferentes palavras.

Para representar as informações para posterior apresentação à Rede Neural Convolucional foi utilizado um alfabeto contendo 26 letras, 10 números e 34 símbolos, totalizando 70 caracteres únicos. Foi escolhido o mesmo conjunto de caracteres utilizados no trabalho de Zhang et al. [2], o qual pode ser visualizado na Figura 3.2.

a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9
- , ; . ! ? : ´ ‘ ` “ / \ | _ @ # $ % ^ & * ~ + - = < > ( ) [ ] { }

Figura 3.2 – Alfabeto considerado na representação das informações.

Com a possibilidade de se gerar uma expressiva quantidade de palavras com um limitado conjunto de caracteres, esta abordagem torna-se de grande valia mesmo em cenários contendo mais de uma língua. Isto é possível pois existem línguas que compartilham a maioria de seus caracteres. Português, inglês, espanhol, alemão ou tagalo são exemplos de línguas que compartilham a mesma estrutura sintática e quase a totalidade de seus caracteres. Entretanto, no caso de considerar na montagem da representação algumas línguas que são um pouco mais complexas, como japonês ou árabe, em uma análise multi-lingual, o espaço vetorial de representação será maior. Neste cenário, o vocabulário necessário para cobrir todos os caracteres dos textos e mapeá-los para uma representação matricial será maior, podendo assim, afetar o tempo de treinamento do modelo.

3.2 Modelos

3.2.1 Conv-Char-R

Recentemente, Zhang et al. [2] alcançaram bons resultados utilizando Redes Neurais Convolucionais para a Análise de Sentimento. Este trabalho foi utilizado como inspiração para a definição da arquitetura aqui proposta. Entretanto, buscou-se simplificar a arquitetura apresentada pelos autores e estabelecer um conjunto ainda mais enxuto de parâmetros e configurações, de forma que a qualidade na classificação fosse mantida e/ou que esta fosse minimamente afetada.

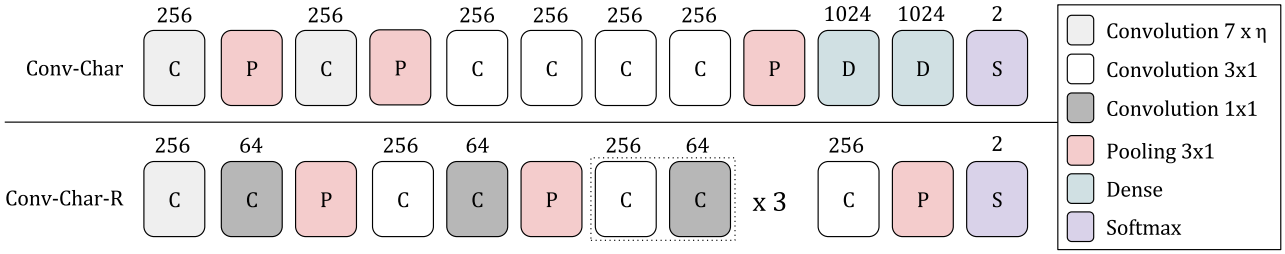


Figura 3.3 – Arquiteturas com representação a nível de caractere: Conv-Char[71] e **Conv-Char-R**.

Este modelo, o qual foi dado o nome de **Conv-Char-R**, simplificando a denominação *Convolutional Charater Reduced*, é baseado no uso de filtros convolucionais 1×1 para aumentar a não-linearidade do modelo ao reduzir a dimensionalidade dos tensores (neurônios), resultando em um menor número de parâmetros. A modificação realizada na arquitetura proposta por Zhang et al. [2] consiste na adição de convoluções de tamanho 1×1 após cada camada convolucional, exceto na última camada. Outra mudança consiste na remoção das camadas totalmente conectadas (densas), onde os neurônios oriundos da camada de *pooling* são mapeados para as duas classes consideradas na análise. A conexão entre a camada de *pooling* e a camada de saída é totalmente conectada, porém sem a existência de uma camada densamente conectada entre ambas. Também foi realizada uma pequena modificação na segunda camada convolucional, adotando-se filtros 1×3 ao invés dos

originais filtros de tamanho 1×7. Este ajuste na segunda camada convolucional implica em $\approx 2.4\times$ menos parâmetros para serem aprendidos nesta camada. O modelo proposto, bem como aquele em que o desenvolvimento da Conv-Char-R foi inspirado, podem ser observados na Figura 3.3.

3.2.2 Conv-Char-S

Com base na simplicidade e desempenho do modelo proposto por Kim [79], onde com somente uma camada convolucional e uma camada de *pooling* conseguiu-se alcançar resultados expressivos, novos testes foram realizados buscando simplificar ainda mais a arquitetura. Assim sendo, foi criada uma nova arquitetura, ainda mais simples que a Conv-Char-R. Não considerando a camada de entrada da Rede como parte desta, a nova arquitetura de Rede Neural Convolucional proposta possui basicamente três elementos: uma **camada convolucional**, uma **camada de *max-overtime pooling*** e uma **camada de saída**. O modelo proposto intitula-se **Conv-Char-S** (Figura 3.4), representando a aglutinação das palavras *Convolutional*, *Character* e *Small*. O modelo recebe este nome pois a sua estrutura contém as seguintes características:

1. *Convolutional*: possui uma camada convolucional, seguida de uma camada de *pooling*.

2. *Character*: a representação de dados utilizada é a nível de caractere.

3. *Small*: possui uma estrutura muito enxuta e simplificada se comparada às demais arquiteturas de CNN, podendo assim ser classificada como "pequena".

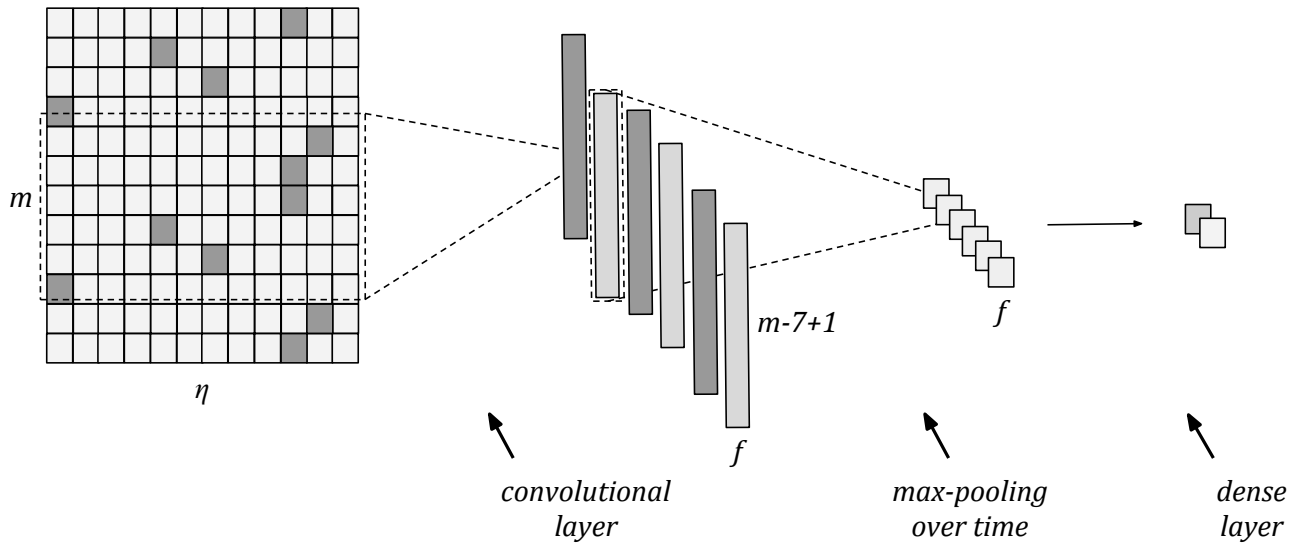


Figura 3.4 – Arquitetura da **Conv-Char-S**, composta basicamente por três componentes: (1) uma camada convolucional com f campos receptivos 7xn; (2) uma camada de *max-pooling-overtime* para seleção das *features* mais importantes das sentenças; (3) uma camada totalmente conectada que realiza um mapeamento linear da camada de *pooling* para a probabilidade de cada classe.

A camada de entrada da CNN, a qual foi explicada na seção anterior, recebe uma matriz de entrada de tamanho $m \times n$, onde m representa cada caractere de uma determinada instância e n é o alfabeto. Como esta arquitetura foi aplicada em dados extraídos do Twitter, e que possuem um tamanho máximo de 140 caracteres, este foi o valor definido para m. O alfabeto, de tamanho $\eta =$ 70, foi apresentado anteriormente na Figura 3.2. Assim sendo, a matriz de entrada possui tamanho fixo de 140 x 70.

A segunda camada da Conv-Char-S realiza operações convolucionais. Para a definição do número de filtros a serem utilizados nesta camada, a rede foi executada com diversas combinações de números de filtros e taxa para regularização (*dropout*). Nestes experimentos, buscou-se encontrar uma combinação de número de filtros e regularização que obtivesse um resultado aceitável sem comprometer a simplicidade da rede (número de parâmetros).

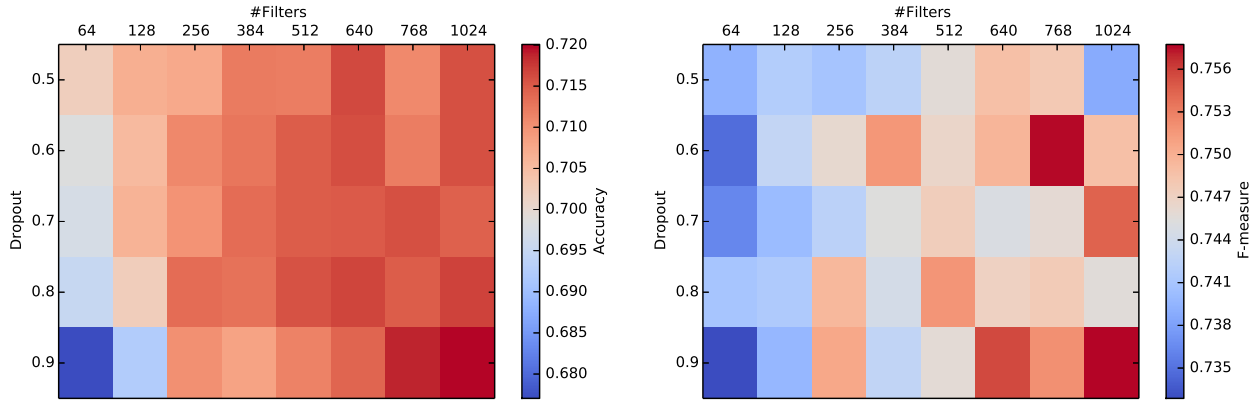


Figura 3.5 – Desempenho da Conv-Char-S no conjunto de validação com diversas variações de arquitetura. À esquerda é apresentado o impacto do *dropout* e o número de filtros considerando a acurácia obtida. À direita pode-se visualizar o impacto dos mesmos componentes considerando a métrica F-measure.

Conforme observado na Figura 3.5, o número de filtros é diretamente proporcional à acurácia do modelo, onde quanto maior o número de filtros convolucionais utilizados, maior é a acurácia obtida. Em média, os modelos com mais de 512 filtros convolucionais obtiveram resultados ligeiramente melhores, entretanto esta diferença não foi superior a 3% (variando entre 0.73 e 0.76), o que não justificaria a sua utilização devido ao maior número de parâmetros treináveis. Mesmo as arquiteturas com mais de 512 filtros obtendo os melhores resultados, nesta arquitetura foi definido uma quantidade de filtros $f = 256$, pois a utilização deste número de filtros apresentou um resultado satisfatório no quesito Quantidade de Filtros *versus* Resultado. A escolha deste número de filtros mantém a principal característica deste modelo, que é possuir um baixo número de parâmetros treináveis. O filtro deslizante possui tamanho fixo de 7x70. Desta forma, a cada 7 caracteres do texto de entrada é gerado um *feature map*, conforme apresenta a Figura 3.6. Nesta camada foram utilizados neurônios ReLU, os quais tiveram a sua matriz de pesos inicializados conforme descrito por He et al. [102].

Após a camada convolucional encontra-se a camada de *pooling* (Figura 3.7), a qual tem por objetivo reduzir o número de *features* que precisam ser aprendidas pelo algoritmo e otimizar o

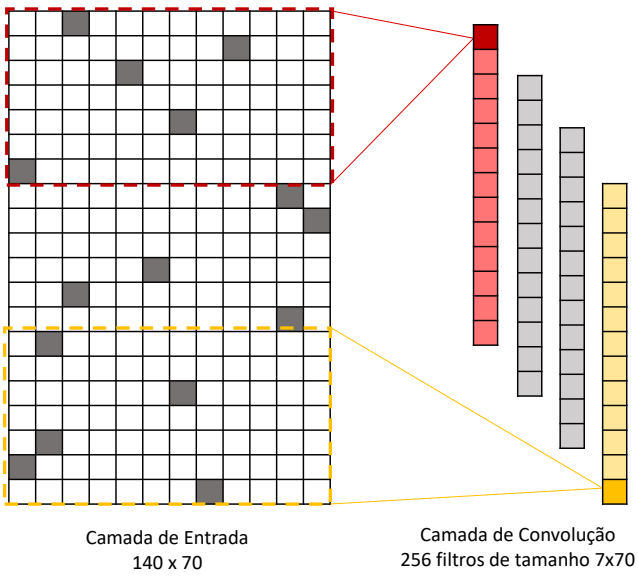


Figura 3.6 – Camada convolucional do modelo proposto. O tamanho da matriz de entrada e o número de filtros convolucionais foram reduzidos para facilitar a visualização.

tempo de treinamento. Nesta camada, é aplicada uma operação de *max-overtime pooling*, a qual foi proposta por Collobert et al. [74], e ganhou recente notoriedade em aplicações envolvendo PLN devido aos bons resultados apresentados no trabalho de Kim [79]. A aplicação deste tipo de *pooling* faz com que o número de *features* de saída desta camada seja exatamente proporcional ao número de filtros aplicados na camada convolucional anterior. Logo, somente o maior valor de cada filtro é mantido e propagado para a próxima camada da rede, e todos os demais valores são descartados na fase de treinamento. Como a **Conv-Char-S** utiliza 256 filtros convolucionais, a saída da camada de *pooling* é igual a 256 neurônios.

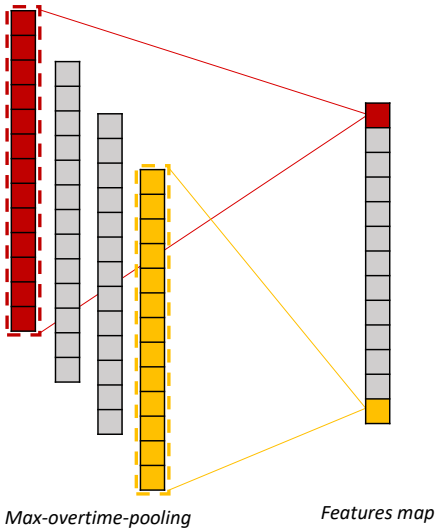


Figura 3.7 – Camada de *pooling* do modelo proposto. O número de filtros apresentados foi reduzido para facilitar a visualização.

A arquitetura proposta diferencia-se por não possuir uma ou mais camadas totalmente conectadas antes da camada de saída, item característico da maioria das Redes Neurais profundas.

A utilização de uma ou mais camadas totalmente conectadas após as operações de convolução e *pooling* visa melhorar o treinamento ao criar mais conexões entre os neurônios, fazendo com que o poder de generalização da RN aumente. Entretanto, com mais parâmetros para serem aprendidos, mais recursos computacionais são necessários para se realizar o treinamento desta arquitetura, fato que buscou-se evitar quando da elaboração da arquitetura proposta.

Após a camada de *pooling*, é realizada uma operação de *dropout*, o qual possui uma probabilidade de 0.9. A definição desta taxa de probabilidade se deu com base nos experimentos apresentados na Figura 3.5. As *features* que não são eliminadas pela função *dropout* são inseridas diretamente na camada de saída. Nesta camada de saída, onde aplica-se uma função de ativação *Softmax* (Eq. 3.1), é calculada a probabilidade da instância pertencer à classe positiva ou negativa. A instância é atribuída à classe que possuir um valor maior desta probabilidade.

$$\mathcal{E}(\hat{y}, \mathcal{P}) = -\sum_{i=1}^{C} [y_i log(\hat{y}_i)] \tag{3.1}$$

onde $\hat{y}$ é a probabilidade gerada pela camada *Sofmax*, e $\hat{\mathcal{P}} = argmax(\hat{y})$.

Nos testes envolvendo o modelo proposto utilizou-se uma taxa de aprendizado de $\alpha = 10^{-3}$. O treinamento da arquitetura foi feito utilizando a regra de atualização de pesos Adam [103], considerando *mini-batches* de 128 instâncias.

3.2.3 NNLS

Buscando aumentar o poder preditivo e também o leque de aplicações, foi proposta uma arquitetura de CNN multitarefa, a qual é capaz de aprender duas tarefas ao mesmo tempo: Análise de Sentimento multi-lingual e Identificação de Linguagem (tradução literal do inglês *language identification*). O modelo, denominado NNLS (abreviação de *Neural Network Language Sentiment*), consiste em três fluxos que realizam operações convolucionais nos dados de entrada utilizando três tamanhos de filtros diferentes. Desta forma, o aprendizado obtido através dos caracteres de entrada é feito de uma maneira explicitamente hierárquica.

A utilização do aprendizado hierárquico através das convoluções possibilita o modelo a identificar fragmentos da semântica textual que são importantes para a análise de sentimento, ao mesmo tempo que provém importantes informações acerca da estrutura sintática de cada língua considerada. Logo, ao forçar a CNN a detectar a linguagem, pode-se facilitar o aprendizado de características específicas para a classificação do sentimento de cada língua. Ao definir uma aplicabilidade multitarefa para o modelo, busca-se encontrar uma combinação de pesos que possam funcionar para ambas as tarefas, possivelmente aplicando um efeito regularizador na classificação.

Para o modelo multitarefa proposto, foram criadas duas arquiteturas distintas: a NNLS-v1 e NNLS-v2, ambos os modelos sendo testados com 128 e 200 filtros em cada camada convolucional. A primeira arquitetura (Figura 3.8) utiliza três camadas convolucionais paralelas que *convolvem* (do

inglês *convolve*) a entrada aplicando três tamanhos de filtros diferentes (similar a abordagem de Kim [79] e Szegedy et al. [104]). Esta característica possibilita a arquitetura a aprender com base em representações de trigramas dependendo do tamanho das palavras.

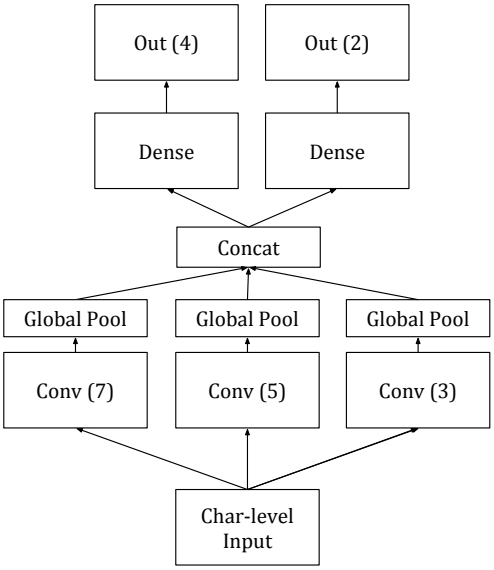


Figura 3.8 – Arquitetura da NNLS-v1.

A arquitetura NNLS-v2 (Figura 3.9) é bastante similar ao modelo anterior, porém, conforme Szegedy et al. [105] as convoluções $7 \times \eta$ são substituídas por três convoluções $3 \times \eta$, e as camadas $5 \times \eta$ são substituídas para duas camadas convolucionais com filtros de tamanho $3 \times \eta$. A ideia por trás desta modificações consiste em utilizar os mesmos campos receptivos ao passo que aumenta a não-linearidade e reduz o número de parâmetros.

A primeira camada de ambos os modelos aplica uma convolução $3 \times \eta$ nos valores de entrada, aumentando os campos receptivos e a não-linearidade, sendo assim responsável pelo aprendizado de média granularidade. Em outras palavras, a aplicação desta estratégia pode ser encarada como uma maneira sofisticada de aprender n-gramas. A segunda camada convolucional possui a função de aprender informações de alta granularidade (e.g., bigramas, dependendo do tamanho das palavras). A última camada convolucional é responsável pelo aprendizado de mais baixo nível dos dados de entrada, buscando encontrar a existência de pequenas sequências de caracteres (e.g., *:), :/, :-D, omg, wtf*, etc), as quais podem ser determinantes para a correta classificação da polaridade de uma sentença.

Após cada sequência de camadas convolucionais é aplicada uma operação de *max-pooling*, a qual seleciona as *features* mais importantes das de cada dimensão temporal. Esta estratégia de *pooling* resulta em vetores $\mathbf{f} \in \mathbb{R}^k$, onde k é o número de filtros da respectiva sequência de camadas convolucionais. Desta forma, a arquitetura proposta gera três vectores $\mathbf{f}$ independentes. A representação final $\mathcal{F}$ é gerada pela concatenação dos três vetores $\mathbf{f}$, explicitamente juntando as as representações aprendidas hierarquicamente pelos três fluxos convolucionais.

A última parte da arquitetura consiste em dois fluxos, cada um contendo contendo duas camadas, uma totalmente conectada e uma camada para os valores de saída. Esta estratégia de

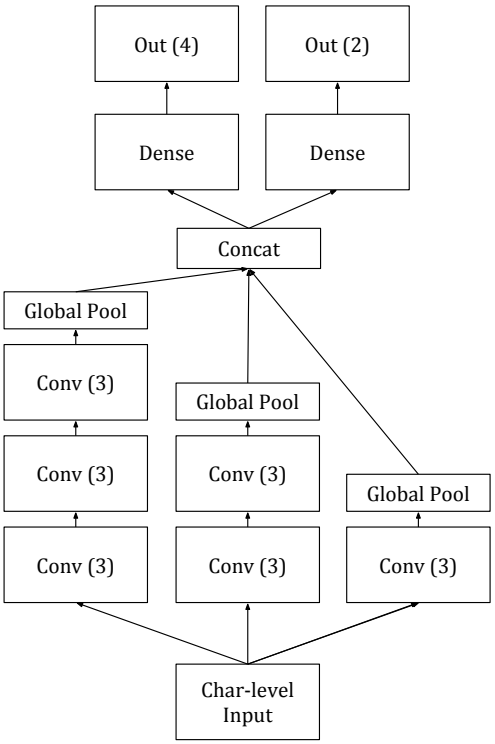


Figura 3.9 – Arquitetura da NNLS-v2.

aplicar dois fluxos, um responsável pela classificação do sentimento e o outro pela classificação da língua, aumenta a não-linearidade do modelo necessária para o aprendizado das *features* para ambas as tarefas. Os neurônios das camadas densamente conectadas, tanto do modelo NNLS-v1 quanto NNLS-v2, aplicam uma função ReLU. A camada de saída é responsável por mapear as representações obtidas pela camada totalmente conectada para cada classe de ambas as tarefas. Logo, a primeira saída da rede é responsável pela identificação da língua, e a segunda saída classifica o sentimento das instâncias. Ambas as camadas de saídas aplicam uma função Softmax, gerando os *scores* de probabilidades para cada classe.

3.3 Trabalhos Relacionados

No trabalho de Narr et al. [106] foi realizada uma Análise de Sentimento de quatro diferentes línguas: inglês, alemão, francês e português. Primeiramente, foram coletados aproximadamente 800 milhões de *tweets*, e então os dados de treino foram rotulados seguindo a mesma abordagem utilizada no trabalho de Pak e Paroubek [58], onde a polaridade de cada mensagem foi definida pela presença de *emoticons* positivos ou negativos. Para execução dos experimentos foi utilizado o algoritmo Naïve Bayes disponível na biblioteca NLTK[1], e os dados foram apresentados ao classificador em estruturas *n*-gramas.

[1]Biblioteca aberta de desenvolvimento para a linguagem Python, disponível em "http://nltk.org/"

Para montar o conjunto de testes do modelo proposto, os autores separaram aproximadamente 12.000 *tweets*, os quais foram manualmente rotulados através da plataforma Mechanical Turk[2] da Amazon. Os dados foram anotados considerando a premissa de que cada *tweet* não pode conter mais de uma polaridade ao mesmo tempo. Os *tweets* neutros foram descartados tanto do conjunto de treino quanto de teste.

Para o *dataset* multi-lingual foram separados 75.000 *tweets* (definido pelo número total de instâncias da língua com a menor quantidade de dados) de cada língua para treinamento do classificador. Os resultados apontam para uma acurácia de 71.5% no experimento envolvendo o conjunto de dados multi-lingual. Ainda, segundo os autores, a representação de dados que utiliza bigramas não obteve um melhor resultado se comparado à representação unigramas, não justificando assim a sua adoção devido ao maior custo de treinamento.

Severyn e Moschitti [1] realizaram a Análise de Sentimento de *tweets* utilizando Redes Neurais Convolucionais *profundas*. Neste trabalho foi utilizada uma arquitetura de CNN simples (Figura 3.10), com somente uma camada convolucional seguida de uma camada de *max-pooling* e finalizando com uma camada *softmax* para classificação. Os autores justificam a utilização de uma arquitetura tão simples com base no trabalho de Kim [79], o qual apresenta resultados que indicam que uma CNN simplificada pode funcionar tão bem quanto uma CNN com uma arquitetura mais complexa. Os resultados deste trabalho alcançaram uma acurácia de 84.79% na classificação de sentimento em sentenças.

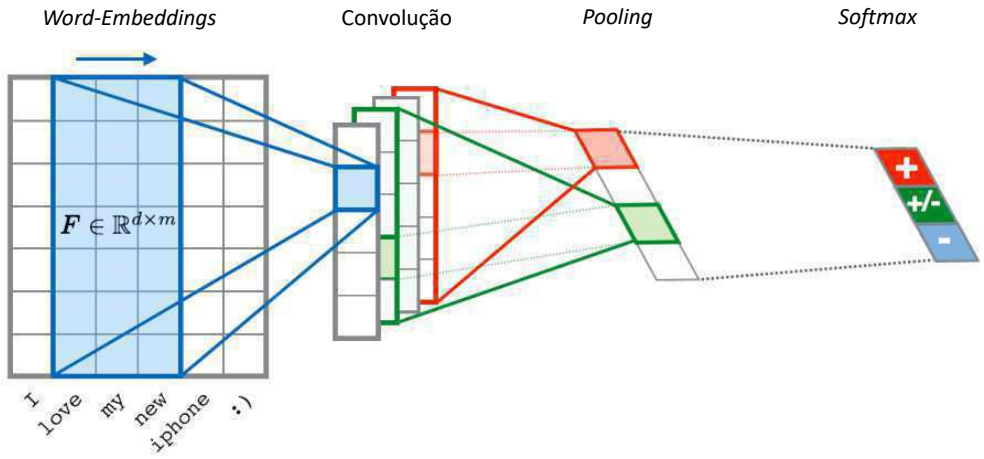


Figura 3.10 – Arquitetura de CNN utilizada por Severyn e Moschitti [1].

Recentemente, Zhang et al. [2] conseguiram bons resultados utilizando CNN e aprendizado a nível de caractere em diversas tarefas como: Classificação de Ontologias, Análise de Sentimento e Categorização de Texto. Para realizar estes experimentos, foram criados dois tipos de CNN: uma grande e uma pequena, ambas com 9 camadas escondidas (6 camadas de convolução e 3 camadas totalmente conectadas), diferenciando-se somente pelo tamanho dos *frames* em cada rede (Figura 3.11). Neste experimento também foram aplicadas 2 funções *dropout* [95] com probabili-

[2]Amazon Mechanical Turk: "http://mturk.com".

dade de 0.5. Para treinamento, foi utilizando o gradiente descendente estocástico (SGD, do inglês *Stochastic Gradient Descent*) por 3 épocas.

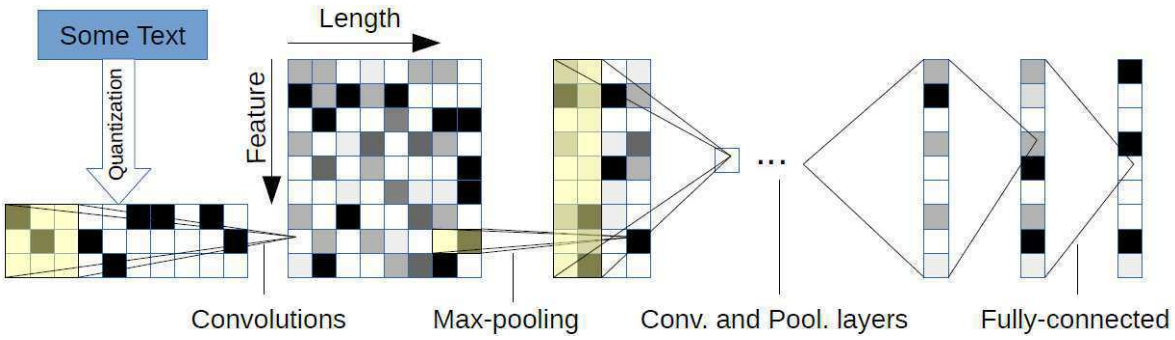


Figura 3.11 – Arquitetura de CNN utilizada por Zhang et al. [2] com representação a nível de caractere.

A Análise de Sentimento foi realizada no *dataset* de *reviews* da Amazon[3], o qual contém mais de 34 milhões de comentários, com 5 classes (valores entre 1 e 5, onde 1 é muito negativo e 5 é muito positivo). Para construir o *corpus*, foram selecionados somente comentários com tamanho entre 100 e 1014 caracteres. Para que fosse realizada uma classificação binária do sentimento, os comentários com polaridade 2 foram convertidos para 1 e os que possuíam polaridade 4 foram convertidos para 5 e ainda foram removidos do *dataset* os comentários com polaridade 3 (neutra). Este processo resultou em um conjunto de treino com 3.600.000 instâncias (1.800.000 de cada polaridade) e 400.000 instâncias para teste (200.000 de cada polaridade). Para construir a matriz de entrada da CNN foi utilizado um alfabeto de 70 caracteres, constituído por 26 letras, 10 números e outros 34 caracteres. Como as sentenças possuíam o tamanho máximo de 1014 caracteres, este foi o tamanho escolhido para a entrada da CNN e, quando o tamanho de uma sentença era menor, foram acrescentados zeros até esta alcançar o tamanho desejado. Os resultados deste experimento são apresentados na Tabela 3.1:

Tabela 3.1 – Resultados dos experimentos, em termos de acurácia, realizados no trabalho de Zhang et al. [2].

Modelo	Treino	Teste
CNN pequena	96.03%	94.50%
CNN grande	97.57%	94.49%

Outro trabalho o qual utilizou CNN e aprendizado a nível de caractere foi realizado por Santos e Gatti [107]. Neste trabalho foram utilizados dois *datasets*, um contendo *reviews* de filmes (SSTb) e outro contendo dados provenientes do Twitter (STS). A arquitetura proposta continha duas camadas convolucionais, seguidas de duas camadas totalmente conectadas, juntamente com a utilização algoritmo SGD para treinamento (minimização do erro). Os resultados apontam para uma acurácia de 85.7% no *corpus* SSTb, ao passo que no *corpus* STS este valor ficou em 86.4%, ambos realizado uma classificação binária (positiva ou negativa). Os autores argumentam que o

[3]https://www.amazon.com/

modelo introduzido é capaz de lidar com o contexto da negação tão bem quanto o método proposto por Socher et al. [108], o qual utiliza Redes Neurais Recursivas para AS.

Dentre os diferentes trabalhos relacionados apresentados anteriormente, a Tabela 3.2 apresenta um comparativo das principais características de cada um destes trabalhos:

Tabela 3.2 – Comparativo das principais características dos trabalhos relacionados.

Trabalhos Relacionados					
Artigo	**Ano**	**Algoritmo**	**Representação**	**Análise**	**Fonte**
Narr et al. [106]	2012	Naïve Bayes	Unigrama e Bigrama	Multi-lingual	Twitter
Santos-Gatti [107]	2014	CNN	Caractere	Mono lingual	Twitter
Severyn-Moschitti [1]	2015	CNN	*Word-embeddings*	Mono lingual	Twitter
Zhang et al. [2]	2015	CNN	Caractere	Mono lingual	Amazon
Esta Dissertação	**2017**	**CNN**	**Caractere**	**Multi-lingual**	**Twitter**

4. EXPERIMENTOS

Nesta seção serão apresentadas as características do ambiente experimental, bem como as informações acerca do *dataset* utilizado nos experimentos. Também serão apresentados os algoritmos utilizados para validação do modelo proposto na Seção 3, bem como os resultados obtidos nos cenários avaliados.

Atualmente, LSTM (do inglês *Long Short-Term Memory*) é o tipo mais bem sucedido de arquitetura de RNN (do inglês *Recurrent Neural Network*), sendo responsável por diversos modelos considerados estado da arte [72]. A inclusão de Redes Neurais Convolucionais no *baseline* apoia-se no seu sucesso em tarefas de reconhecimento de imagens [109, 110], vídeos [111] e, obviamente, em aplicações de PLN [112, 79, 2]. Por fim, o algoritmo SVM possui grande aplicação em tarefas de PLN e já é um algoritmo consolidado no meio, o que justifica a sua seleção como *baseline*.

4.1 Ambiente Experimental

O treinamento de Redes Neurais profundas normalmente é feito em GPUs (Unidades de Processamento Gráfico, do inglês *Graphics Processing Unit*) pois estas necessitam de um menor tempo de execução se comparado à CPUs (Unidades Centrais de Processamento, do inglês *Central Processing Unit*). Desta forma, todos os experimentos foram executados em uma GPU Tesla K40, a qual possui 12 GB de memória, utilizando-se a biblioteca de desenvolvimento *Theano*[1] [113]. Ainda, visando uma rápida e objetiva prototipação do modelo, foi utilizado o *framework* de desenvolvimento Keras [114], o qual simplifica a programação através da disponibilização de módulos e funções para o desenvolvimento de Redes Neurais profundas. A hierarquia dos componentes utilizados para desenvolver e avaliar o modelo proposto ficou distribuída conforme apresenta a Figura 4.1.

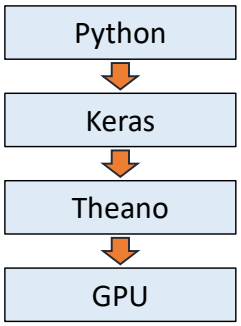


Figura 4.1 – Hierarquia dos componentes utilizados no desenvolvimento do modelo proposto.

[1] *Theano* é uma biblioteca para a linguagem Python que permite uma eficiente definição, otimização e avaliação de expressões matemáticas que envolvem vetores multidimensionais.

4.2 Métricas

Os resultados foram avaliados segundo duas métricas conhecidas em tarefas envolvendo PLN. A acurácia (Eq. 4.1) é uma medida que avalia o resultado de um treinamento, considerando o valor real de uma instância e o valor predito pelo algoritmo. Outra métrica avaliada é a F-measure (Eq. 4.4), a qual representa a média harmônica entre *Precision* (Eq. 4.2) e *Recall* (Eq. 4.3). Tanto a acurácia quanto a F-measure estão em uma escala de 0 e 1, onde quanto maior o valor, melhor é o desempenho.

$$\textit{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.1}$$

onde VP é o total de Verdadeiros Positivos, VN são os Verdadeiros Negativos, FP representa os Falsos Positivos e FN são os Falsos Negativos.

$$Precision = \frac{VP}{VP + FP} \tag{4.2}$$

$$Recall = \frac{VP}{VP + FN} \tag{4.3}$$

$$F\text{-measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.4}$$

Para a definição dos resultados avaliados, excetuando o SVM, o qual foi executado somente uma vez para cada conjunto, cada modelo foi executado 5 vezes, tanto no *dataset* multi-lingual quanto nos conjuntos separados por língua. Assim sendo, o valor apresentado para cada métrica consiste na média desta nas cinco execuções de cada algoritmo. Nos experimentos que apresentam informações relacionadas ao tempo de treinamento, a unidade de medida apresentada é segundos (s). Todos os experimentos foram executados tendo os mesmos recursos de hardware disponíveis (memória, uso de CPU/GPU, etc) e todos os modelos foram treinados segundo o pseudocódigo 4.1.

4.3 *Dataset*

Para a validação do modelo proposto foi utilizado o conjunto de dados de Mozetič et al. [115], o qual contém dados obtidos do Twitter. Este *dataset* contém *tweets* de 13 diferentes línguas, os quais estão anotados em três classes distintas: positiva, neutra e negativa. Contendo aproximadamente 1.6 milhões de tweets, esta é a maior fonte para treinamento e validação de aplicações multi-linguais disponível publicamente no presente momento.

O endereço de acesso aos dados pode ser obtido no artigo. As instâncias estão separadas por língua em 13 arquivos no formato de texto (".txt"). Em cada um destes arquivos está o

Algorithm 4.1 Pseudocódigo do treinamento utilizado nos experimentos.

```
max_epochs = 100
threshold = 20
best_accuracy = 0
count = threshold
for epoch in max_epochs do
    while count > 0 do
        train model()
        current_accuracy = model.accuracy
        if current_accuracy >= best_accuracy then
            count = threshold
            best_accuracy = current_accuracy
            model.save_weights()
        else
            count -= 1
        end
    end
end
model.load_weights()
model.test()
return metrics
```

código identificador da mensagem na plataforma (*TweetId*), a classe a qual o texto foi classificado (*HandLabel*) e ainda o identificador da pessoa que rotulou a mensagem.

Para realizar o download dos dados, foi utilizada a API (Interface de Programação de Aplicativos, do inglês *Application Programming Interface*) do Twitter, informando o identificador *TweetId* da mensagem a qual desejava-se acessar. Infelizmente nem todos os identificadores retornam a referida mensagem, pois estas informações não estavam mais disponíveis na plataforma. Desta forma, o total de *tweets* obtidos difere consideravelmente do total informado no artigo de Mozetič et al. [115].

Devido à diferenças semânticas e sintáticas entre as 13 línguas, somente foram considerados dados de quatro línguas: inglês, espanhol, português e alemão. Ainda, todos os *tweets* da classe neutra foram descartados, reduzindo a classificação a um problema binário. A Tabela 4.1 apresenta os totais de instâncias de cada língua presentes no conjuntos de treino, teste e conjuntos de validação. O *dataset* intitulado *Multi-lingual* nada mais é do que a junção de todas as línguas em um mesmo conjunto.

Tabela 4.1 – Distribuição das instâncias no *dataset* utilizado nos experimentos.

	Treino		**Validação**		**Teste**		
Língua	**Negativa**	**Positiva**	**Negativa**	**Positiva**	**Negativa**	**Positiva**	**Total**
Inglês	7,784	7,645	1,131	1,146	2,200	2,264	22,170
Alemão	7,502	10,727	1,063	1,544	2,057	2,944	25,837
Português	17,170	12,202	2,427	1,745	4,990	3,560	42,094
Espanhol	8,211	18,491	1,189	2,574	2,372	5,251	38,088
Multi-lingual	40,667	49,065	5,810	7,009	11,619	14,019	128,189
% do total	89,732 (70%)		12,819 (10%)		25,638 (20%)		128,189 (100%)

4.4 *Baselines*

Nesta seção serão apresentados os parâmetros utilizados na montagem dos algoritmos utilizados nos experimentos. Nas arquiteturas de LSTM e também CNN com representação distribuída foi utilizado um vocabulário de 157.000 palavras únicas gerado a partir do *dataset* multi-lingual, não utilizando assim *embeddings* pré-treinados.

Um dos pontos comuns nas implementações de Redes Neurais Convolucionais e LSTM foi a utilização da regra *Adam* [103] para atualização dos gradientes, com *mini-batches* de 128 instâncias, durante a fase de treinamento. Ainda, foi utilizado *dropout* somente para regularização das camadas totalmente conectadas, e todos os modelos foram treinados utilizando o *log da verossimilhança* (do inglês *Log-likelihood*) como *Loss* na camada que aplica uma função *Softmax* em dois neurônios (positivo e negativo).

4.4.1 LSTM

Bengio et al. [116] relatam que uma Rede Recorrente (RNN, do inglês *Recurrent Neural Network*) tem ciclos em seu grafo que torna possível o armazenamento de informações sobre entradas passadas por um tempo não fixado previamente, dependendo dos pesos da rede e dos valores de entrada. Esta capacidade não é verificada em RNAs não recorrentes (estáticas). Entretanto, esta influência de uma entrada na camada intermediária e, posteriormente na camada de saída, tende a crescer exponencialmente ou é diluído enquanto é propagado pelas conexões recorrentes da RNN. Este problema é conhecido como Problema da Dissipação do Gradiente (do inglês *Vanishing Gradient Problem*) e pode ser visualizado na Figura 4.2.

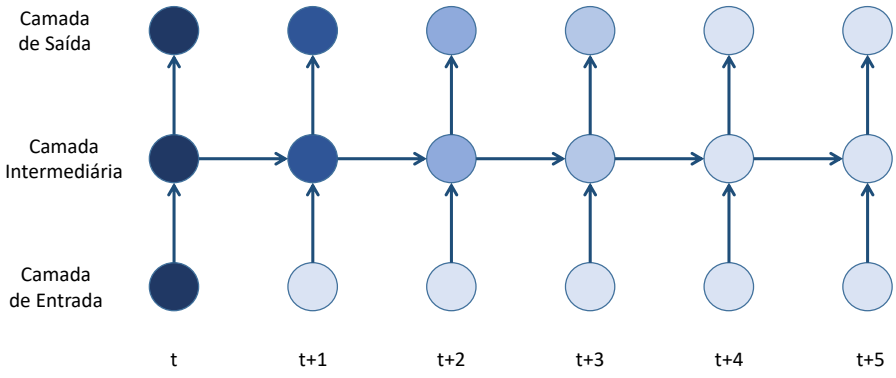


Figura 4.2 – Problema da Dissipação do Gradiente em uma RNN. Os nós representam neurônios e seu conteúdo representa a sensibilidade para a entrada, quanto mais escuro o nó, mais sensível à entrada azul escura.

Hochreiter e Schmidhuber [117] propuseram um modelo que modifica a arquitetura de uma RNN criando a rede Memória de Curto-Prazo Longa (LSTM, do inglês *Long Short-Term*

Memory). Este tipo de RNA pode ser treinada para aprender ou também esquecer memórias prévias, dependendo dos valores de entrada corrente e o estado atual. A única diferença estrutural da LSTM para uma RNN é que os elementos da camada intermediária deixam de ser neurônios com funções de ativação e tornam-se blocos de memória.

A grande ideia por traz da LSTM é a introdução das representações de estado (ou neurônios de memória) que preservam os gradientes com o decorrer do tempo [72]. O acesso a estas células de memória é controlado pelos portões (do inglês *gates*). Durante o treinamento da rede, a cada valor de entrada (estado), o portão define a intensidade da entrada que deverá ser esquecida e o quanto deverá ser repassada à célula de memória. A Figura 4.3 ilustra este funcionamento.

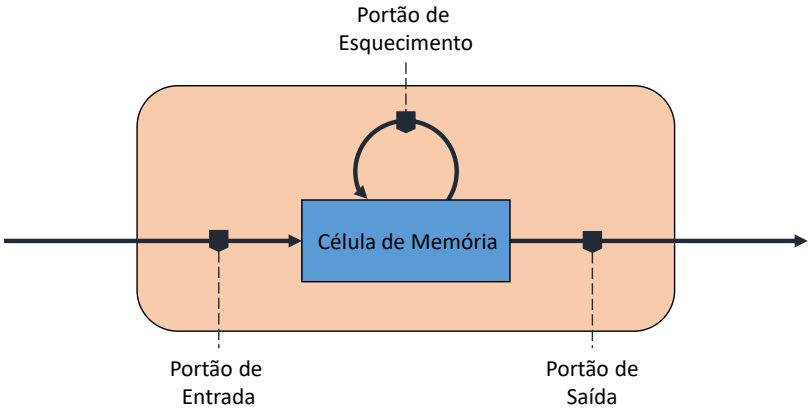


Figura 4.3 – Célula de memória de uma LSTM.

Conforme definido por Goldberg [72], o portão $\mathbf{g} \in [0, 1]^n$ é um vetor de valores na escala [0, 1] que é multiplicado com outro vetor $\mathbf{v} \in \mathbb{R}^n$. O resultado desta operação é adicionado a outro vetor. Os índices em **v** que são próximos a 1 em **g** têm a sua passagem permitida, ao passo que os valores próximos a zero são bloqueados (esquecidos).

Essa capacidade das LSTMs em abrir ou fechar o portão de esquecimento (do inglês *forget gate*) aumenta a sua robustez ao Problema da Dissipação do Gradiente. Quando o *forget gate* está ativo, ou seja, a sua ativação está próxima de 1, o gradiente não irá dissipar. Como a ativação do *forget gate* nunca será maior que 1, o gradiente não irá explodir. Contrariando a Figura 4.2, onde ocorre o Problema da Dissipação dos Gradientes, a Figura 4.4 ilustra como os mesmos são tratados em uma LSTM.

- **Parâmetros utilizados nos experimentos:** Na implementação deste algoritmo foi utilizada a arquitetura padrão de LSTM, utilizado portões de entrada e esquecimento. Para encontrar o número de neurônios ψ da camada escondida na LSTM e a taxa de *dropout* γ foi aplicado a busca em grade (do inglês *grid search*), no conjunto de validação, nos seguintes espaços de busca: $\psi \in \{64, 128, 256, 512, 640, 768, 1024\}$ e $\gamma \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Assim sendo, esta arquitetura possui somente uma camada escondida, com um total de 256 neurônios com a função de ativação tangente hiperbólica. O tamanho *d* do *embedding* utilizado foi 300, ao passo que a taxa de *dropout* foi de 0.5 na última camada escondida. Para este modelo foi

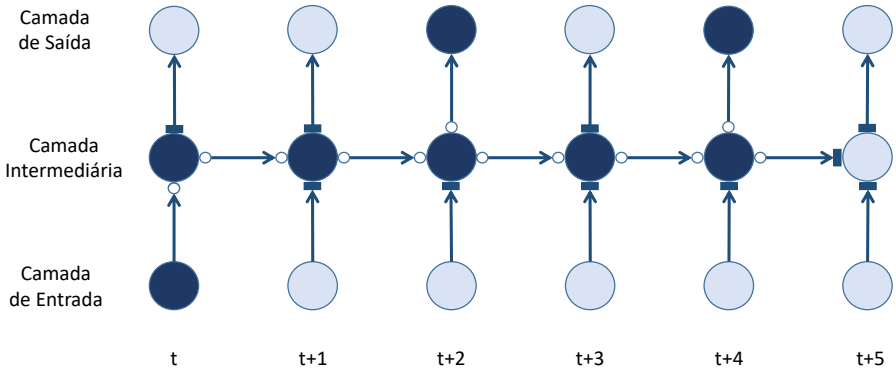


Figura 4.4 – Portões dos blocos de memória preservam a informação nas redes LSTM. Para os blocos da camada intermediária, os círculos representam portões abertos e os blocos portões fechados. O portão embaixo do nó é o portão de entrada, o superior é o portão de saída e o lateral é o portão de esquecimento.

utilizada uma taxa de aprendizado α de 1×10^{-3}. Neste modelo foi utilizada somente representação de informações *word-embeddings*. No restante deste trabalho, o modelo de LSTM aqui detalhado será denominado **LSTM-Emb**. Para este modelo também foi testada uma implementação derivada da LSTM-Emb, a qual possui as mesmas configurações, diferenciando-se somente pela não atualização dos vetores de representação durante a fase de treinamento. Este modelo sem a atualização dos *embeddings* será denominado **LSTM-Emb-Freeze** no decorrer deste trabalho.

4.4.2 Redes Neurais Convolucionais

Uma descrição detalhada desta arquitetura de RNA é apresentada na Seção 2.3.4.

- **Parâmetros utilizados nos experimentos:** Para a execução de CNN foram testadas as representações distribuída e local. Para a representação distribuída foram utilizados os mesmos parâmetros propostos no trabalho de Kim [79]. Neste modelo, o *embedding* foi definido em $d = 300$, onde os vetores de cada palavra foram inicializados com valores aleatórios e então otimizados durante o treinamento da RNA. Foram utilizados 128 filtros nas convoluções, uma taxa de *dropout* γ de 0.8 na última camada totalmente conectada e uma taxa de aprendizado α de 1×10^{-3}. Este modelo de CNN apresentado será denominado **Conv-Emb**. Também foi testada uma implementação derivada da Conv-Emb, com as mesmas configurações, diferenciando-se somente pela não atualização dos vetores de representação durante a fase de treinamento. Este modelo sem a atualização dos *embeddings* será denominado **Cov-Emb-Freeze**.

  Foi implementada, também, uma CNN com representação a nível de caractere (representação local). Esta é uma implementação da CNN (versão pequena) apresentada no trabalho de Zhang et al. [2]. Neste modelo foi considerado um tamanho de entrada $m = 140$, pois

este é o número máximo de caracteres que um *tweet* pode conter. O alfabeto de caracteres considerado possui um tamanho total de $\eta = 70$ e pode ser visualizado na Figura 3.2, a qual foi apresentada no Capítulo 3. Neste modelo foi utilizado uma taxa de *dropout* γ de 0.8 e 0.9 na penúltima e última camada da RNA, respectivamente, e uma taxa de aprendizado $\alpha = 10^{-3}$. A matriz de pesos dos neurônios ReLU foi inicializada conforme o procedimento apresentado em [102]. O modelo de CNN com representação local aqui apresentado será denominado **Conv-Char**.

4.4.3 SVM

O SVM (Máquina de Vetores de Suporte, do inglês *Support Vector Machine*), é um algoritmo de aprendizagem supervisionada utilizado para tarefas de classificação, proposto Vapnik [118]. Seu funcionamento apoia-se na construção de um hiperplano para separar as classes no espaço vetorial, mantendo maior distância possível entre estas. Os elementos, por sua vez, para serem classificados utilizando este modelo, devem ser representados em forma de vetores. Tratando-se de aplicações de PLN, comumente o *bags-of-words* é o tipo de representação escolhida.

Em um caso real de classificação de dados, certamente o algoritmo terá que lidar com atributos contendo dezenas, centenas ou até milhares de atributos (dimensões). Porém, para fins de exemplificação, suponhamos que no conjunto de treinamento existam instâncias com somente dois atributos, ou seja, instâncias com duas dimensões, e que estas divididas em duas classes (triângulos e quadrados). Neste caso, o classificador irá analisar os dados e então traçar uma reta que separe os dados das duas classes da melhor forma possível, tentando manter a maior distância entre as instâncias destas classes. Esta operação pode ser visualizada na Figura 4.5.

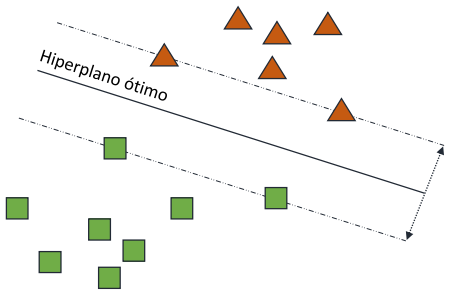


Figura 4.5 – Separação linear de dados no SVM.

Em alguns casos não é possível traçar uma reta para separar linearmente os dados. Assim, utiliza-se uma função ϕ que mapeia os vetores para uma dimensão de ordem maior, possibilitando assim a sua separação linear. Essa função ϕ é conhecida como *kernel*, e o seu funcionamento pode ser visualizado na Figura 4.6.

O SVM, muito conhecido pelo seu poder de generalização, é um dos algoritmos mais utilizados em tarefas de mineração de texto [37, 119, 16, 57, 120, 78, 121]. Nestes cenários muitas vezes

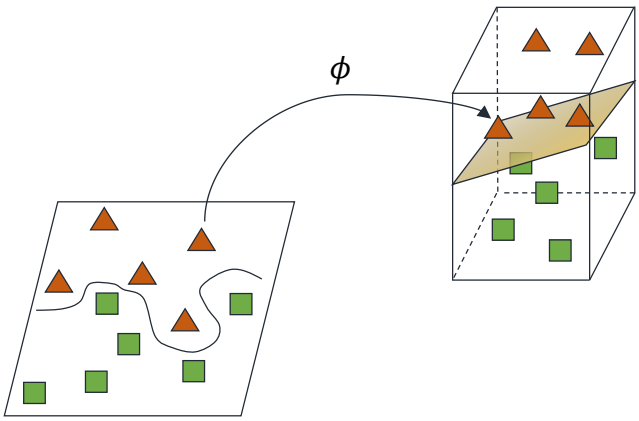


Figura 4.6 – Mapeamento da função ϕ para um hiperplano de ordem superior no SVM.

é necessário lidar com a alta dimensionalidade dos dados, e outros algoritmos falham por gerarem modelos muito ajustados (*overfitting*) ou pouco ajustados (*underfitting*). Ainda, dependendo do problema a ser resolvido, é comum encontrar bases de dados rotulados que não possuem um grande número de instâncias (onde poderia-se aplicar outro algoritmo como as RNAs), tornando-o assim uma boa alternativa.

- **Parâmetros utilizados nos experimentos:** Para a execução do SVM foram testadas diferentes tipos de representação de dados. Também, para a montagem das representações *n*-gramas foi realizada uma etapa de pré-processamento de dados, pois esta é uma etapa comumente aplicada na execução de SVM para tarefas de PLN. Assim sendo, foram realizados os seguintes procedimentos na etapa de pré-processamento: normalização do texto (todas as palavras foram convertidas em minúsculas), remoção de *stopwords*, remoção de ruídos (URLs, @user e #) e *stemming*.

  Desta forma, as implementações SVM testadas são assim denominadas:

  - **SVM-U:** implementação do SVM utilizando a representação unigramas;
  - **SVM-B:** implementação do SVM utilizando a representação bigramas;
  - **SVM-T:** implementação do SVM utilizando a representação trigramas;
  - **SVM-UB:** implementação do SVM utilizando a representação unigramas e bigramas;
  - **SVM-UBT:** implementação do SVM utilizando a representação unigramas, bigramas e trigramas;

  A documentação do modelo SVM que foi implementado pode ser visualizada neste link[2].

[2] "http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html"

4.5 Resultados

Após explicado o cenário e toda a metodologia experimental adotada, nesta seção serão apresentados os resultados dos experimentos. Na Seção 4.5.1 serão apresentados os resultados da primeira arquitetura desenvolvida, a **Conv-Char-R**, a qual inspirou o desenvolvimento de uma segunda arquitetura de CNN, a **Conv-Char-S**, que por sua vez será apresentada na Seção 4.5.2. Os resultados envolvendo a última arquitetura proposta, a NNLS, serão apresentados na Seção 4.5.3. Os resultados de cada arquitetura serão apresentados em separado, pois, como os modelos foram evoluindo gradativamente, os cenários analisados foram aqueles considerados quando cada arquitetura foi projetada.

4.5.1 Conv-Char-R

Para validação da arquitetura **Conv-Char-R**, a mesma foi comparada com os seguintes *baselines*: **LSTM**, **Conv-Emb[-Freeze]**, e **Conv-Char**. Estes experimentos, os quais também foram executados no *dataset* apresentado na Seção 4.3, resultaram nas métricas apresentadas na Tabela 4.2.

Tabela 4.2 – Acurácia e F-measure obtidas no *dataset* multi-lingual.

Modelo	Acurácia	F-measure
Conv-Emb-Freeze	0.688	0.728
Conv-Emb	**0.714**	0.749
LSTM-Emb	0.713	**0.753**
Conv-Char	0.696	0.721
Conv-Char-R	0.695	0.722

A melhor F-measure é obtida pela **LSTM-Emb**, 0.753, e este modelo também alcança a segunda melhor acurácia, 0.713, perdendo somente para o modelo **Conv-Emb**. Considerando o tipo de representação nos modelos convolucionais, os resultados apontam que arquiteturas que utilizam a representação *word-embeddings* levam uma ligeira vantagem frente aos modelos que utilizam representação a nível de caractere.

Os modelos também foram testados em execuções por língua. Neste cenário, cada modelo foi treinado no *dataset* multi-lingual, e então foi testado em conjuntos separados por cada língua. Os resultados desta análise individual das línguas são apresentados nas Tabelas 4.3 e 4.4.

Tabela 4.3 – F-measure por língua.

Modelo	Inglês	Alemão	Português	Espanhol
Conv-Emb-Freeze	0.736	0.752	0.583	0.795
Conv-Emb	0.763	0.782	0.608	0.804
LSTM-Emb	**0.779**	**0.783**	**0.619**	**0.806**
Conv-Char	0.729	0.758	0.589	0.776
Conv-Char-R	0.722	0.755	0.605	0.774

Considerando a arquitetura **Conv-Char-R**, a qual é uma versão reduzida e otimizada da **Conv-Char**, pode-se observar que ambas apresentam virtualmente o mesmo desempenho. Desta

Tabela 4.4 – Acurácia por língua.

Modelo	Inglês	Alemão	Português	Espanhol
Conv-Emb-Freeze	0.734	0.694	0.671	0.678
Conv-Emb	0.766	0.733	**0.695**	0.693
LSTM-Emb	**0.770**	**0.734**	0.686	**0.697**
Conv-Char	0.736	0.718	0.688	0.668
Conv-Char-R	0.727	0.721	0.694	0.660

forma, a **Conv-Char-R** apresenta-se como o melhor custo-benefício comparando o resultado obtido e o número de parâmetros. Ainda, a **Conv-Char-R** apresenta um desempenho de aproximadamente $\approx 2\%$ menor que o melhor modelo (**LSTM-Emb**) em termos de acurácia e $\approx 3\%$ em termos de F-measure. Logo, a **Conv-Char-R** alcança resultados competitivos se comparada a modelos considerados estado da arte em Análise de Sentimento, com a grande vantagem de conter $\approx 90\times$ menos parâmetros e consumir $\approx 4\times$ menos memória que modelos baseados em *embeddings*, conforme apresenta a Figura 4.7.

4.5.2 Conv-Char-S

Os experimentos das diferentes representações de informações para o SVM apontam para uma supremacia nos resultados envolvendo a representação unigramas, em todas as línguas testadas. Esta supremacia confirma-se tanto para a acurácia quanto F-measure, conforme apresenta a Tabela 4.5.

Tabela 4.5 – Resultados das execuções do SVM. Os melhores resultados em cada língua estão em destaque.

Modelo	Língua	Acurácia	F-measure
SVM-U	Multi-lingual	**0.694**	**0.751**
	Inglês	**0.728**	**0.776**
	Português	**0.715**	**0.684**
	Espanhol	**0.615**	**0.733**
	Alemão	**0.691**	**0.735**
SVM-B	Multi-lingual	0.562	0.717
	Inglês	0.687	0.694
	Português	0.630	0.594
	Espanhol	0.560	0.726
	Alemão	0.673	0.695
SVM-T	Multi-lingual	0.556	0.712
	Inglês	0.595	0.604
	Português	0.556	0.526
	Espanhol	0.549	0.702
	Alemão	0.616	0.651
SVM-UB	Multi-lingual	0.672	0.746
	Inglês	0.695	0.755
	Português	0.670	0.596
	Espanhol	0.600	0.732
	Alemão	0.688	0.706
SVM-UBT	Multi-lingual	0.660	0.743
	Inglês	0.610	0.684
	Português	0.670	0.610
	Espanhol	0.560	0.712
	Alemão	0.645	0.658

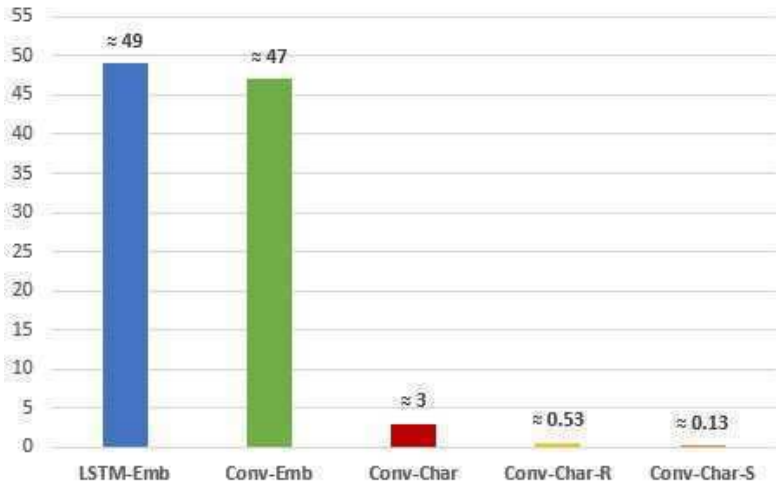
Para o aplicações em cenários multi-linguais, talvez não seja interessante utilizar representações que não unigramas. Isto se deve ao fato de que, utilizando bigramas ou outra combinação maior de palavras, ocorreriam duas línguas ou mais em uma mesma *feature*, fazendo com que o espaço dimensional seja maior e a assertividade da classificação seja reduzida. Conforme apresenta a Tabela 4.5, os resultados envolvendo bigramas e também trigramas foram bem piores se comparados a unigramas para dados multi-linguais. Nas demais línguas o desempenho também foi abaixo, porém, de modo geral, a diferença foi mais sutil. Quando utilizou-use unigramas+bigramas e unigramas+bigramas+trigramas, pode-se observar que o resultado foi superior se comparado ao uso de bigramas e trigramas separados, o que sugere que a representação unigramas pode ter elevado o resultado das combinações em que esta esteve presente.

O SVM mostrou-se um excelente *baseline*. Porém muito de seu desempenho deve-se às técnicas de pré-processamento aplicadas nos dados (e.g., remoção de *stopwords* e *stemming*), juntamente com a sua capacidade de lidar com a alta dimensionalidade de *features* mesmo em *datasets* de tamanho médio, como é o presente caso. Outro problema ao se adotar o SVM e realizar uma etapa de pré-processamento, é a necessidade deste em acessar uma lista de *stopwords* e *stemmers* para cada língua considerada na análise, tornando a sua execução custosa. Ainda, a representação unigrama tende a ser muito esparsa a medida que novas línguas são adicionadas, tornando a sua adoção particularmente problemática no que tange a *maldição da dimensionalidade*.

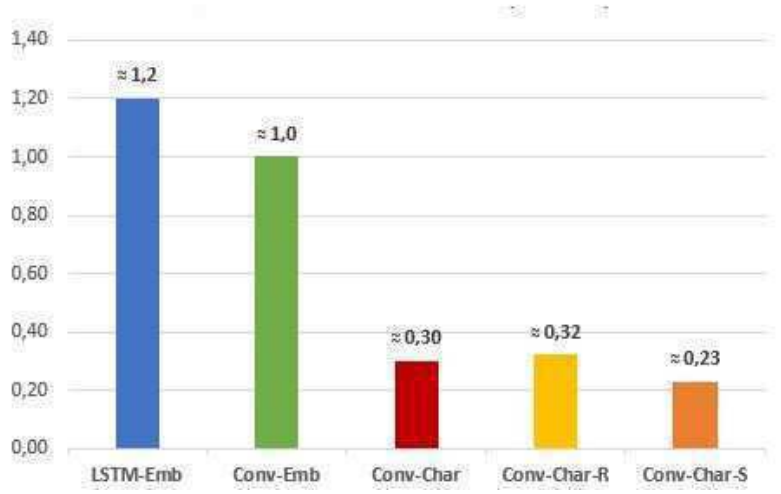
Dentre os *baselines* envolvendo arquiteturas de *deep learning*, o modelo proposto alcançou, de maneira geral, o melhor resultado em termos de acurácia e F-measure. No conjunto multi-lingual, arquitetura proposta (**Conv-Char-S**) alcança a melhor acurácia (0.721) e também a melhor F-measure (0.752), sendo seguida pela **Conv-Emb** e **LSTM-Emb**. Conforme apresenta a Tabela 4.6, normalmente o modelo que alcançou a melhor acurácia também obteve a melhor F-measure.

Analisando a **LSTM-Freeze** e **Conv-Emb-Freeze**, percebe-se que modelos com *word-embeddings* possuem um desempenho melhor quando atualiza-se a sua representação (*embedding*). Assim sendo, aprender o tipo de representação através do *backpropagation* é sempre melhor do que desconsiderar o seu treinamento (*freeze*) e somente aprender os demais parâmetros da rede. Conforme apresenta a Figura 4.7(a), os modelos que utilizam representações *word-embeddings* possuem uma enorme quantidade de parâmetros que precisam ser aprendidos durante a fase de treinamento, criando assim uma desvantagem deste tipo de abordagem frente a outros modelos.

As arquiteturas com representação a nível de caractere mostram-se muito menos custosa no que tange parâmetros e memória, conforme apresentado na Figura 4.7. Considerando a **Conv-Char**, a mesma não mostra-se capaz de superar o desempenho de modelos baseados em *word-embeddings*, entretanto, ainda assim, apresenta um desempenho competitivo em relação à capacidade preditiva e à complexidade do modelo. Uma desvantagem de representações *word-embeddings* é que, por menor que seja a diferença entre duas palavras, será necessário gerar uma nova *feature* para representação, o que pode demandar uma grande quantidade de memória para armazenamento dos vetores do vocabulário. O mesmo problema não é encontrado para abordagens a nível de caractere, a qual não apoia-se na utilização de um vocabulário pré-definido.



(a) Quantidade de parâmetros de cada modelo.



(b) Quantidade de memória necessária para treinamento do modelo.

Figura 4.7 – Quantidade de parâmetros (em milhões) e memória (em GB) requerida para treinamento de cada modelo.

Analisando o desempenho da arquitetura proposta, pode-se observar que a mesma apresenta resultados muito melhores que a **Conv-Char**, a qual também é uma Rede Neural Convolucional e utiliza caracteres como representação de entrada. Conforme apresenta a Tabela 4.6, a **Conv-Char-S** obteve o melhor desempenho preditivo, tanto no cenário multi-lingual quanto na análise individual das línguas.

Comparando o modelo proposto com arquiteturas que utilizam *embeddings*, o desempenho também foi superior, em todos os cenários analisados. A arquitetura **Conv-Emb** foi a que apresentou um desempenho bem parecido, porém ligeiramente inferior, tanto em termos de acurácia quanto de F-measure.

Especificamente para a língua espanhola, todos os modelos tiveram um desempenho semelhante. Provavelmente este comportamento deve-se ao fato das classes estarem muito desba-

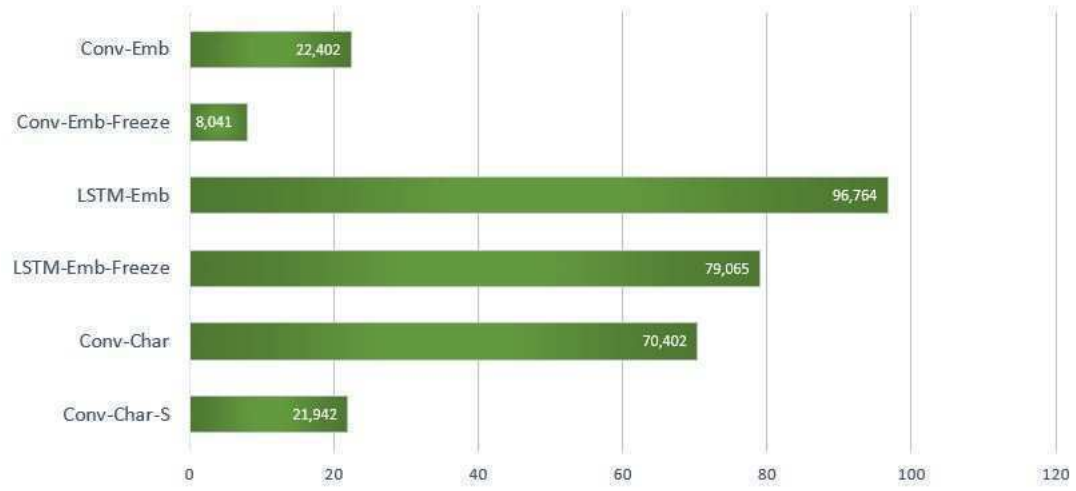
Tabela 4.6 – Resultados dos modelos avaliados nos diferentes *datasets*.

Modelo	Língua	Acurácia	F-measure
Conv-Emb	Multi-lingual	0.715	0.750
	Inglês	0.769	**0.777**
	Português	0.730	0.653
	Espanhol	0.693	0.811
	Alemão	0.731	0.779
Conv-Emb-Freeze	Multi-lingual	0.671	0.719
	Inglês	0.744	0.741
	Português	0.702	0.609
	Espanhol	0.689	0.811
	Alemão	0.690	0.746
LSTM-Emb	Multi-lingual	0.709	0.742
	Inglês	0.771	0.766
	Português	0.717	0.637
	Espanhol	0.693	0.808
	Alemão	0.729	0.773
LSTM-Emb-Freeze	Multi-lingual	0.640	0.699
	Inglês	0.656	0.642
	Português	0.657	0.538
	Espanhol	0.689	0.814
	Alemão	0.632	0.721
Conv-Char	Multi-lingual	0.696	0.726
	Inglês	0.746	0.737
	Português	0.743	0.649
	Espanhol	0.683	**0.815**
	Alemão	0.722	0.764
Conv-Char-S	**Multi-lingual**	**0.721**	**0.752**
	Inglês	**0.775**	0.765
	Português	**0.753**	**0.655**
	Espanhol	**0.695**	0.814
	Alemão	**0.749**	**0.793**

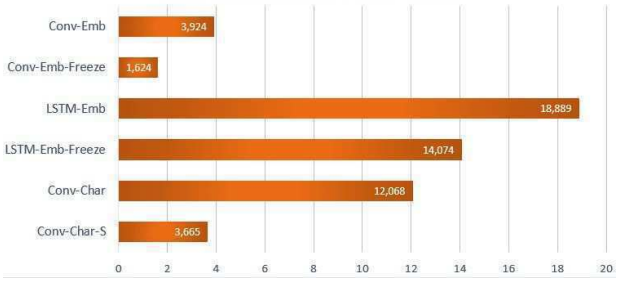
lanceadas, tanto no conjunto de treinamento quanto no de validação e teste. Nas demais línguas os dados também estão desbalanceados, porém esta diferença não é tão abrupta se comparada ao *dataset* espanhol.

Com base na Figura 4.7, nota-se que os modelos os quais utilizam *embeddings* como representação demandam muito mais parâmetros que serão aprendidos durante a fase de treinamento, e também necessitam mais espaço para carregar o modelo em memória, tanto para execuções em CPU quanto em GPU. A camada de *embedding* é a grande responsável por inflar o total de parâmetros destes modelos, pois, para cada vetor de palavras, existem 300 parâmetros (utilizando $d = 300$). Considerando o conjunto multi-lingual, o qual possui 157.000 palavras únicas, o total de parâmetros necessários somente para montar a representação dos dados é igual a $157.000 \times 300 = 47.100.000$.

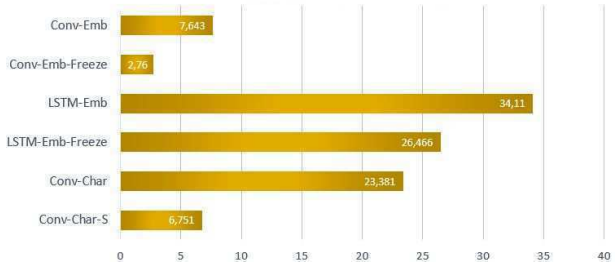
A arquitetura proposta, além do melhor resultado em todos os cenários analisados, também mostrou-se a melhor escolha no que tange tempo de processamento. Na Figura 4.8, pode-se visualizar o tempo necessário para o treinamento de cada modelo por uma época. Os modelos onde o *embedding* não é atualizado durante a fase de treinamento, devido ao número de parâmetros muito reduzidos, tiveram um tempo de execução muito menor que a sua versão que atualiza a representação. Entretanto, a acurácia e F-measure obtidas por estas versões sem a atualização do *embedding* mostram que sua adoção não compensa. Por outro lado, a **Conv-Emb** teve um bom desempenho nos cenários analisados, alcançando um tempo de execução ligeiramente acima do modelo proposto, mesmo realizando o treinamento dos *embeddings*.



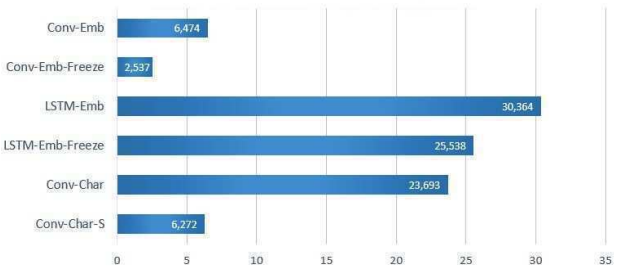
(a) Tempo de treinamento para o conjunto Multi-lingual.



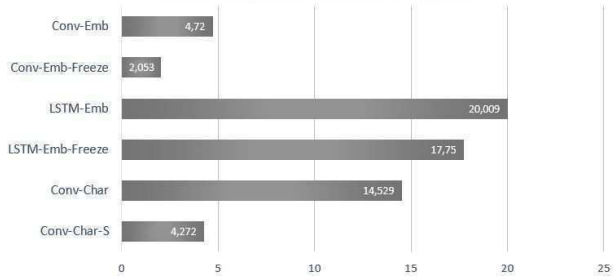
(b) Tempo de treinamento para o idioma Inglês.



(c) Tempo de treinamento para o idioma Português.



(d) Tempo de treinamento para o idioma Espanhol.



(e) Tempo de treinamento para o idioma Alemão.

Figura 4.8 – Tempo de treinamento (em segundos) de uma época nos modelos avaliados nos diferentes *datasets*.

Comparando a **Conv-Char-S** com o outro modelo que utiliza representação a nível de caractere, a **Conv-Char**, percebe-se que o tempo de treinamento da segunda é $\approx 2.5\times$ maior. Esta diferença deve-se ao fato de que a **Conv-Char** realiza mais operações de convolução e *pooling* em sua estrutura, fazendo com que o seu treinamento seja mais demorado. Entretanto, conforme os resultados apresentado anteriormente na Tabela 4.6, este maior tempo de treinamento e número de camadas de convolução e *pooling* não reflete em um resultado melhor, fazendo com que a **Conv-Char-S** seja também a melhor escolha entre estes dois modelos.

4.5.3 NNLS

A NNLS realiza uma classificação multitarefa, onde aprende-se a classificar o sentimento e também identificar a língua a qual cada documento foi escrito. Desta forma, os resultados são apresentados da seguinte maneira: (i) todas as arquiteturas de redes neurais (NNLS e *baselines*) foram treinados para realizar cada tarefa separadamente; (ii) todos os modelos foram treinados em uma classificação multitarefa, onde os *baselines* foram adaptados para poderem realizar tal classificação, similar à abordagem proposta. Diferente da metodologia apresentada na Seção 4.2, onde a acurácia considerada para os testes envolvendo a Conv-Char-R e Conv-Char-S foi o valor médio de cinco execuções de cada algoritmo, para a NNLS considerou-se apenas uma execução, tanto para o modelo proposto quanto *baselines*. Os resultados referentes à classificação multitarefa são apresentados na Tabela 4.7.

Tabela 4.7 – Resultados dos modelos na classificação multitarefa.

Modelo	Análise de Sentimento	Identificação de Língua
LSTM	71.33%	97.62%
ConvEmb	71.75%	97.74%
ConvChar	70.59%	97.55%
ConvChar-S	73.37%	98.37%
NNLS-v1	**73.63%**	**98.11%**
NNLS-v2	**74.43%**	**98.40%**

A arquitetura proposta supera o desempenho de todos os demais *baselines* no cenário multitarefa. Os demais modelos baseados em caracteres também obtiveram bons resultados, modelos os quais possuem diversas camadas convolucionais, fato que pode ter aumentado a não-linearidade dos modelos e melhorado o desempenho destes no cenário multi-lingual. Entretanto, o modelo proposto possui menos camadas de convolução e *pooling*, e mesmo assim alcança uma acurácia relativa de $\approx 5.15\%$ maior para AS e $\approx 0.87\%$ para a identificação da língua.

Excetuando o modelo proposto, a arquitetura que mais se destacou foi a Conv-Char-S, mostrando-se resultados bastante competitivos em ambas as tarefas. As abordagens que utilizam *word-embeddings*, mesmo possuindo um tempo de treinamento superior conforme apresentado anteriormente, não obtiveram um resultado melhor. LSTM e ConvEmb alcançaram um resultado similar, onde a LSTM obteve uma acurácia ligeiramente superior que ConvEmb.

Outro cenário analisado foi o treinamento dos modelos para realizarem somente a classificação da polaridade e somente a identificação da língua. Estes experimentos foram conduzidos buscando identificar se algum *baseline* alcançaria um resultado superior ao modelo proposto considerando uma classificação individual. A Tabela 4.8 apresenta os resultados da análise de sentimento e identificação da língua realizadas em separado, onde não houve o compartilhamento dos pesos da rede entre as tarefas.

O modelo proposto alcançou o melhor resultado na tarefa de Análise de Sentimento, ao passo que LSTM alcançou a maior acurácia na identificação da língua. Estes resultados consideram o treinamento de cada modelo para a classificação da polaridade e da língua individualmente.

Tabela 4.8 – Resultado dos modelos na classificação individual das tarefas. Os valores entre parênteses são o número de filtros convolucionais.

Modelo	Análise de Sentimento	Identificação de Língua
LSTM	71.57%	98.43%
ConvEmb	71.19%	98.38%
ConvChar	70.71%	96.91%
ConvChar-S	71.78%	97.61%
NNLS-v1(128)	**72.52%**	**97.72%**
NNLS-v1(200)	**72.87%**	**97.53%**
NNLS-v2(128)	**72.97%**	**97.59%**
NNLS-v2(200)	**73.55%**	**97.83%**

Como um experimento complementar, também foi investigado o desempenho de todos os modelos em uma análise de sentimento separando os conjuntos por língua. Neste cenário todos os modelos foram treinados em dados multi-linguais mas foram testados nos conjuntos separados por língua. O resultado deste experimento pode ser visualizado na Tabela 4.9.

Tabela 4.9 – Resultado da AS com treinamento em dados multi-linguais e teste com dados de cada língua.

Modelo	Inglês	Alemão	Português	Espanhol
LSTM	70.57%	76.58%	71.12%	69.32%
ConvEmb	72.30%	**77.58%**	69.28%	69.50%
FastText	70.04%	77.22%	71.01%	69.15%
ConvChar	70.24%	72.07%	71.24%	69.74%
ConvChar-S	74.97%	75.50%	74.36%	69.83%
NNLS-v1	**75.35%**	**76.31%**	**74.86%**	**70.01%**
NNLS-v2	**76.68%**	**76.72%**	**75.13%**	**70.02%**

5. CONCLUSÕES

Este trabalho apresentou uma abordagem que utiliza Redes Neurais Convolucionais para resolver a tarefa de Análise de Sentimento multi-lingual em dados extraídos do Twitter, introduzindo três arquiteturas de CNN distintas. O conjunto multi-lingual continha quatro línguas em sua composição: inglês, português, espanhol e alemão. Normalmente, as abordagens para AS multi-lingual apoiam-se na utilização de tradutores de texto, convertendo a a língua alvo para a língua com os dados de treinamento. Desta forma, pode-se aplicar normalmente estes dados em um mesmo algoritmo classificador.

As três arquiteturas propostas fazem com que esta etapa de pré-processamento dos dados multi-linguais seja desnecessária, tampouco o uso de ferramentas para tradução. Isto se deve ao fato dos modelos utilizarem a representação a nível de caractere, o que simplifica a forma como as informações são apresentadas ao classificador.

Primeiramente, foi apresentada a arquitetura Conv-Char-R, a qual alcança resultados competitivos se comparada a arquiteturas mais complexas como a LSTM, porém com a grande vantagem de possuir $\approx 90\times$ menos parâmetros e consumir $\approx 4\times$ menos memória para seu treinamento. Com base nos bons resultados obtidos pela Conv-Char-R, foi elaborada uma arquitetura com uma estrutura ainda mais reduzida. Desta forma, foi proposta a Conv-Char-S, uma arquitetura de Rede Neural Convolucional com somente uma camada convolucional, uma camada de *pooling* e uma camada de saída. Mesmo com uma estrutura mais limitada, o modelo proposto alcançou uma acurácia de 0.721 e F-score de 0.752 no conjunto multi-lingual, superando assim todos os *baselines*, incluindo modelos considerados estado da arte em tarefas de AS, como LSTM. Para validação do modelo proposto, também foram executados experimentos envolvendo SVM com diversas combinações de representações, CNN utilizando *word-embeddings* como representação, e também uma arquitetura mais complexa de CNN, a qual também utiliza representação a nível de caractere, além da LSTM, citada anteriormente. Nestes experimentos a Conv-Char-S obteve a melhor acurácia e F-measure em todas as línguas analisadas, e também no conjunto multi-lingual.

Também foram analisados outras características das diferentes arquiteturas que não o seu poder preditivo. Nestes experimentos buscou-se identificar os modelos que necessitam um maior recurso de máquina para realizar o seu treinamento, bem como aqueles que demandam mais tempo neste processo. Verificou-se que arquiteturas que utilizam *embeddings* como representação possuem uma quantidade expressivamente maior de parâmetros para treinamento e também necessitam mais memória para execução de seus modelos. Dentre os modelos analisados, a Conv-Char-S foi a arquitetura que menos gerou parâmetros treináveis e também a que menos memória exigiu, totalizando $1225\times$ menos parâmetros e utilizando $\approx 4\times$ menos memória que o segundo melhor *baseline*. No que tange tempo para treinamento, o modelo proposto também mostrou-se a melhor escolha dentre os *baselines*, demandando até $\approx 4.5\times$ menos tempo.

A última arquitetura de CNN proposta, a NNLS, apresenta o conceito de aprendizagem hierárquica. Neste cenário, cada fluxo foca em aprender diferentes composições dos caracteres,

como *emoticons*, n-gramas e combinações de palavras, características que podem ser cruciais para a definir o resultado da classificação. A estrutura deste modelo foi projetada para realizar uma classificação multi tarefa, podendo este identificar a polaridade das sentenças e também identificar a qual língua os documentos pertencem, com o aprendizado acontecendo simultaneamente.

A NNLS-v2 teve o melhor desempenho dentre todos os algoritmos analisados em uma classificação multitarefa, alcançando uma acurácia de 74.43% para AS e 98.40% para Identificação da Língua. Excetuando as duas versões da NNLS, o segundo melhor modelo foi a Conv-Char-S, com uma acurácia de 73.37% para AS e 98.35% para Identificação de Língua. A inclusão de mais uma tarefa para o algoritmo, que conceitualmente diminuiria o seu poder preditivo, gerou justamente o efeito inverso, aumentando a regularização do modelo e consequentemente o seu poder preditivo, mostrando-se a melhor escolha entre todos os *baselines* analisados.

5.1 Publicações e Trabalhos Futuros

A arquitetura **Conv-Char-R**, a qual foi apresentada na Seção 3.2.1, foi publicada na conferência *Thirtieth International Florida Artificial Intelligence Research Society Conference* (FLAIRS-30), conforme a referência [122]. A arquitetura **Conv-Char-S**, apresentada na Seção 3.2.2, foi publicada na conferência *2017 International Joint Conference on Neural Networks* (IJCNN-2017), conforme a referência [123]. A arquitetura **NNLS**, introduzida na Seção 3.2.3, será apresentada na conferência *The 33rd ACM/SIGAPP Symposium On Applied Computing* (SAC-2018).

Nos trabalhos futuros pretende-se explorar as demais línguas presentes no *dataset*, buscando verificar o comportamento dos modelos propostos em outras línguas que não as quatro exploradas no presente trabalho. Neste trabalho não foram considerados idiomas com uma estrutura sintática completamente diferente, como o japonês e o mandarim. Caso fossem consideradas linguagens assim na análise, faria-se necessária um trabalho para definir o ponto de separação dos caracteres, haja vista que em linguagens como o japonês, eventualmente um simples caractere representa um conjunto de palavras, fato que não se observa nas línguas analisadas no presente trabalho. Em outros trabalhos futuros, também deseja-se testar os modelos em outras tarefas de PLN que não sejam a Análise de Sentimento ou Identificação de Língua, verificando assim a sua aplicabilidade em outros cenários e com outras finalidades.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proceedings of the 2015 ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 959–962, ACM, 2015.

[2] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, pp. 649–657, 2015.

[3] C. F. da Silva, *Uso de Informações Lingüísticas na etapa de pré-processamento em Mineração de Textos*. PhD thesis, Universidade do Vale do Rio dos Sinos, 2004.

[4] J. S. Olsson, D. W. Oard, and J. Hajič, "Cross-language text classification," in *Proceedings of the 2005 ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 645–646, 2005.

[5] B. Fortuna and J. Shawe-Taylor, "The use of machine translation tools for cross-lingual text mining," in *Proceedings of the 2005 ICML Workshop on Learning with Multiple Views*, 2005.

[6] C. Banea, R. Mihalcea, J. Wiebe, and S. Hassan, "Multilingual subjectivity analysis using machine translation," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 127–135, 2008.

[7] J. G. Shanahan, G. Grefenstette, Y. Qu, and D. A. Evans, "Mining multilingual opinions through classification and translation," in *Proceedings of 2004 AAAI Spring Symposium on Exploring Attitude and Affect in Text*, 2004.

[8] G. Zhou, Z. Zeng, J. X. Huang, and T. He, "Transfer learning for cross-lingual sentiment classification with weakly shared deep neural networks," in *Proceedings of the 2016 ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 245–254, ACM, 2016.

[9] L. Shi, R. Mihalcea, and M. Tian, "Cross language text classification by model translation and semi-supervised learning," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1057–1067, Association for Computational Linguistics, 2010.

[10] B. Pang and L. Lee, *Opinion mining and sentiment analysis*, vol. 2. Now Publishers Inc., 2008.

[11] V. L. Sauter, *Decision support systems for business intelligence*, vol. 2. John Wiley & Sons, 2014.

[12] B. Liu, *Sentiment analysis and opinion mining*, vol. 1. Morgan & Claypool Publishers, 2012.

[13] M. Tsytsarau and T. Palpanas, "Survey on mining subjective data on the web," *Data Mining and Knowledge Discovery*, vol. 24, 2012.

[14] K. Becker and D. Tumitan, "Introdução à mineração de opiniões: Conceitos, aplicações e desafios," in *Simpósio Brasileiro de Banco de Dados*, 2013.

[15] S. Soderland, "Building a machine learning based text understanding system," in *Proceedings of the 2001 Workshop on Adaptive Text Extraction and Mining (IJCAI)*, pp. 64–70, 2001.

[16] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 79–86, Association for Computational Linguistics, 2002.

[17] T. T. Thet, J.-C. Na, and C. S. Khoo, "Aspect-based sentiment analysis of movie reviews on discussion boards," *Journal of Information Science*, 2010.

[18] M. Gamon, "Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis," in *Proceedings of the 2004 International Conference on Computational Linguistics*, p. 841, Association for Computational Linguistics, 2004.

[19] M. Ghiassi, J. Skinner, and D. Zimbra, "Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network," *Expert Systems with Applications*, vol. 40, no. 16, pp. 6266–6282, 2013.

[20] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan, "A system for real-time twitter sentiment analysis of 2012 us presidential election cycle," in *Proceedings of the 2012 ACL System Demonstrations*, pp. 115–120, Association for Computational Linguistics, 2012.

[21] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment in twitter events," *Journal of the American Society for Information Science and Technology*, 2011.

[22] P. Barnaghi, P. Ghaffari, and J. G. Breslin, "Text analysis and sentiment polarity on fifa world cup 2014 tweets," in *Proceedings of the 2015 ACM Conference of the Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD)*, 2015.

[23] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, 2011.

[24] X. Ji, S. A. Chun, and J. Geller, "Monitoring public health concerns using twitter sentiment classifications," in *Proceedings of the 2013 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 335–344, IEEE, 2013.

[25] A. D. Kramer, J. E. Guillory, and J. T. Hancock, "Experimental evidence of massive-scale emotional contagion through social networks," in *Proceedings of the National Academy of Sciences*, vol. 111, pp. 8788–8790, 2014.

[26] Y. R. Tausczik and J. W. Pennebaker, "The psychological meaning of words: Liwc and computerized text analysis methods," *Journal of Language and Social Psychology*, 2010.

[27] K. Denecke, "Using sentiwordnet for multilingual sentiment analysis," in *Proceedings of the 2008 IEEE International Conference on Data Engineering Workshop (ICDEW)*, pp. 507–512, IEEE, 2008.

[28] A.-L. Gînscă, E. Boroș, A. Iftene, D. TrandabĂț, M. Toader, M. Corîci, C.-A. Perez, and D. Cristea, "Sentimatrix: multilingual sentiment analysis service," in *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pp. 189–195, Association for Computational Linguistics, 2011.

[29] Z. Lin, S. Tan, and X. Cheng, "Language-independent sentiment classification using three common words," in *Proceedings of the 2011 ACM International Conference on Information and Knowledge Management*, pp. 1041–1046, ACM, 2011.

[30] M. Taboada, K. Voll, and J. Brooke, "Extracting sentiment as a function of discourse structure and topicality," *Simon Fraser Univeristy School of Computing Science Technical Report*, 2008.

[31] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, *et al.*, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 2007 ACL Annual Meeting on Interactive Poster and Demonstration Sessions*, pp. 177–180, Association for Computational Linguistics, 2007.

[32] C. C. Aggarwal and C. Zhai, *Mining text data*, vol. 1. Springer Science & Business Media, 2012.

[33] R. Feldman and J. Sanger, *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge University Press, 2007.

[34] N. Indurkhya and F. J. Damerau, *Handbook of natural language processing*, vol. 2. CRC Press, 2010.

[35] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne, "Finding high-quality content in social media," in *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pp. 183–194, ACM, 2008.

[36] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, vol. 1. McGraw-Hill, Inc., 1986.

[37] T. Joachims, *Text categorization with support vector machines: Learning with many relevant features*, vol. 1. Springer, 1998.

[38] C. D. Manning, P. Raghavan, H. Schütze, *et al.*, *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008.

[39] D. D. Lewis, "Evaluating text categorization i.," in *Proceedings of the 1991 Human Language Technology Conference (HLT)*, vol. 91, pp. 312–318, 1991.

[40] M. L. Neves, *PubsFinder-um Agente Inteligente para Busca e Classificação de Páginas de Publicações*. PhD thesis, Universidade Federal de Pernambuco, 2001.

[41] C. V. Gasperin and V. L. S. Lima, "Fundamentos do processamento estatístico da linguagem natural," *Trabalho Individual, PUCRS*, 2000.

[42] A. T. d. C. Junior, *Processamento de linguagem natural para indexação automática semântico-ontológica*. PhD thesis, Universidade de Brasília, 2013.

[43] D. D. Palmer and M. A. Hearst, "Adaptive multilingual sentence boundary disambiguation," *Computational Linguistics*, vol. 23, no. 2, pp. 241–267, 1997.

[44] J. A. Goldsmith, "Segmentation and morphology," *The Handbook of Computational Linguistics and Natural Language Processing*, vol. 57, p. 364, 2010.

[45] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[46] J. B. Lovins, *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory Cambridge, 1968.

[47] C. D. Paice, "An evaluation method for stemming algorithms," in *Proceedings of the 1994 ACM Annual International Conference on Research and Development in Information Retrieval*, pp. 42–50, Springer-Verlag New York, Inc., 1994.

[48] V. Orengo and C. Huyck, "A stemming algorithmm for the portuguese language," in *Proceedings of the 2001 Symposium on String Processing and Information Retrieval (SPIRE)*, p. 0186, IEEE, 2001.

[49] M. Lennon, D. S. Peirce, B. D. Tarry, and P. Willett, "An evaluation of some conflation algorithms for information retrieval," *Journal of Information Science*, 1981.

[50] D. Harman, "How effective is suffixing?," *Journal of the American Society for Information Science*, 1991.

[51] V. M. Bastos, *Ambiente de descoberta de conhecimento na web para a língua portuguesa*. PhD thesis, Universidade Federal do Rio de Janeiro, 2006.

[52] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*, vol. 1. O'Reilly Media, Inc., 2009.

[53] T. Erjavec and S. Džeroski, "Machine learning of morphosyntactic structure: Lemmatizing unknown slovene words," *Applied Artificial Intelligence*, vol. 18, no. 1, pp. 17–41, 2004.

[54] G. Grefenstette, "Evaluation techniques for automatic semantic extraction: comparing syntactic and window based approaches," in *Proceedings of the 1993 Workshop on Acquisition of Lexical Knowledge from Text (SIGLEX)*, Citeseer, 1993.

[55] M. Ciaramita, "Boosting automatic lexical acquisition with morphological information," in *Proceedings of the 2002 ACL workshop on Unsupervised Lexical Acquisition*, pp. 17–25, Association for Computational Linguistics, 2002.

[56] V. Pekar, "Linguistic preprocessing for distributional classification of words," in *Proceedings of the 2004 Workshop on Enhancing and Using Electronic Dictionaries*, pp. 15–21, Association for Computational Linguistics, 2004.

[57] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Stanford Project Report*, vol. 1, p. 12, 2009.

[58] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in *Proceedings of the 2010 International Conference on Language Resources and Evaluation(LREC)*, 2010.

[59] D. Tumitan and K. Becker, "Sentiment-based features for predicting election polls: a case study on the brazilian scenario," in *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 2, pp. 126–133, IEEE, 2014.

[60] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan, "A system for real-time twitter sentiment analysis of 2012 us presidential election cycle," in *Proceedings of the 2012 ACL System Demonstrations*, pp. 115–120, Association for Computational Linguistics, 2012.

[61] A. Bifet and E. Frank, "Sentiment knowledge discovery in twitter streaming data," in *Proceedings of the 2010 International Conference on Discovery Science*, pp. 1–15, Springer, 2010.

[62] A. Abbasi, H. Chen, and A. Salem, "Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums," *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 3, p. 12, 2008.

[63] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of the 2003 International Conference on World Wide Web*, pp. 519–528, ACM, 2003.

[64] J. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin, "Learning subjective language," *Computational linguistics*, vol. 30, no. 3, pp. 277–308, 2004.

[65] S. Mukherjee and P. Bhattacharyya, "Sentiment analysis: A literature survey," *arXiv preprint arXiv:1304.4520*, 2013.

[66] S.-M. Kim and E. Hovy, "Automatic detection of opinion bearing words and sentences," in *Proceedings of the 2005 International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 61–66, 2005.

[67] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: social honeypots+ machine learning," in *Proceedings of the 33rd ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 435–442, ACM, 2010.

[68] D. Tang, F. Wei, B. Qin, M. Zhou, and T. Liu, "Building large-scale twitter-specific sentiment lexicon: A representation learning approach," in *Proceedings of 2014 International Conference on Computational Linguistics (COLING)*, pp. 172–182, 2014.

[69] D. Rao, D. Yarowsky, A. Shreevats, and M. Gupta, "Classifying latent user attributes in twitter," in *Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents*, pp. 37–44, ACM, 2010.

[70] N. F. da Silva, E. R. Hruschka, and E. R. Hruschka, "Tweet sentiment analysis with classifier ensembles," *Decision Support Systems*, vol. 66, pp. 170–179, 2014.

[71] X. Zhang and Y. LeCun, "Text understanding from scratch," *arXiv preprint arXiv:1502.01710*, 2015.

[72] Y. Goldberg, "A primer on neural network models for natural language processing," *Journal of Artificial Intelligence Research*, 2015.

[73] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[74] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[75] Z. Zhao and T. Liu, "Cluster-driven model for improved word and text embedding," in *Proceedings of the 2016 European Conference on Artificial Intelligence (ECAI)*, vol. 285, p. 99, IOS Press, 2016.

[76] L. Wolf, Y. Hanani, K. Bar, and N. Dershowitz, "Joint word2vec networks for bilingual semantic representations," *International Journal of Computational Linguistics and Applications*, vol. 5, no. 1, pp. 27–44, 2014.

[77] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, vol. 14, pp. 1532–1543, 2014.

[78] J. Smailović, M. Grčar, N. Lavrač, and M. Žnidaršič, "Stream-based active learning for sentiment analysis in the financial domain," *Information Sciences*, vol. 285, pp. 181–203, 2014.

[79] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[80] S. Haykin, "Neural networks: A comprehensive foundation," *Neural Networks*, vol. 2, 2004.

[81] K. Faceli, A. Lorena, J. Gama, and A. Carvalho, *Inteligência artificial: uma abordagem de aprendizado de máquina*. Grupo Gen-LTC, 2011.

[82] A. d. P. Braga, A. Carvalho, and T. B. Ludermir, *Redes neurais artificiais: teoria e aplicações*. Livros Técnicos e Científicos, 2000.

[83] S. Haykin, *Redes Neurais: Princípios e Prática*, vol. 2. Bookman, 1999.

[84] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[85] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 2010 International Conference on Machine Learning (ICML)*, pp. 807–814, 2010.

[86] M. Minsky and S. Papert, *Perceptrons: An introduction to Computational Geometry*, vol. 1. MIT Press, 1969.

[87] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[88] J. L. McClelland, D. E. Rumelhart, P. R. Group, *et al.*, "Parallel distributed processing," *Explorations in the Microstructure of Cognition*, vol. 2, 1986.

[89] A. C. P. d. L. F. Carvalho, A. d. P. Braga, and T. B. Ludermir, *Fundamentos de redes neurais artificiais*. DCC/IM, COPPE/Sistemas, NCE-UFRJ, 1998.

[90] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," in *Proceedings of the 1988 Connectionist Models Summer School*, 1988.

[91] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[92] B. A. Pearlmutter, "Gradient descent: Second-order momentum and saturating error," in *Advances in Neural Information Processing Systems*, pp. 887–894, 1991.

[93] M. Riedmiller and I. Rprop, "Rprop-description and implementation details," 1994.

[94] J. Ba and B. Frey, "Adaptive dropout for training deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 3084–3092, 2013.

[95] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[96] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[97] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, 2014.

[98] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[99] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, IEEE, 1998.

[100] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[101] L. G. Hafemann, *An analysis of deep neural networks for texture classification*. PhD thesis, Universidade Federal do Paraná, 2014.

[102] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

[103] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[104] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the 2015 IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[105] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, 2015.

[106] S. Narr, M. Hulfenhaus, and S. Albayrak, "Language-independent twitter sentiment analysis," *Knowledge Discovery and Machine Learning (KDML)*, pp. 12–14, 2012.

[107] C. N. dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of the 2014 International Conference on Computational Linguistics (COLING)*, 2014.

[108] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *Proceedings of the 2013 ACL Conference*, Citeseer, 2013.

[109] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Proceedings of the 2011 International Joint Conference on Artificial Intelligence (IJCAI)*, p. 1237, Barcelona, Spain, 2011.

[110] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3642–3649, IEEE, 2012.

[111] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the 2014 IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1725–1732, 2014.

[112] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 2008 International Conference on Machine learning (ICML)*, pp. 160–167, ACM, 2008.

[113] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, 2016.

[114] F. Chollet *et al.*, "Keras." https://github.com/fchollet/keras, 2015.

[115] I. Mozetič, M. Grčar, and J. Smailović, "Multilingual twitter sentiment classification: The role of human annotators," *PloS One*, vol. 11, no. 5, 2016.

[116] Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *Proceedings of the 1993 IEEE International Conference on Neural Networks*, pp. 1183–1188, IEEE, 1993.

[117] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[118] V. Vapnik, *The nature of statistical learning theory*, vol. 2. Springer science & business media, 2013.

[119] T. Joachims, *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers, 2002.

[120] J. Silva, L. Coheur, A. C. Mendes, and A. Wichert, "From symbolic to sub-symbolic information in question classification," *Artificial Intelligence Review*, vol. 35, no. 2, pp. 137–154, 2011.

[121] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification.," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1422–1432, 2015.

[122] W. Becker, J. Wehrmann, H. E. Cagnini, and R. C. Barros, "An efficient deep neural architecture for multilingual sentiment analysis in twitter," in *Proceedings of the 2017 International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pp. 246–251, AAAI, 2017.

[123] J. Wehrmann, W. Becker, H. E. Cagnini, and R. C. Barros, "A character-based convolutional neural network for language-agnostic twitter sentiment analysis," in *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2384–2391, IEEE, 2017.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br