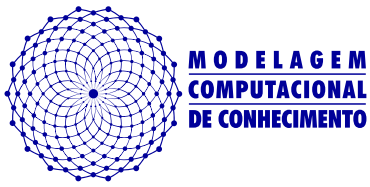


Dissertação de Mestrado

Categorização de Textos por Aprendizagem de Máquina

Keila Barbosa Costa dos Santos
keilabarbosa@laccan.ufal.br

Orientadores:

Dr. Alejandro Cesar Frery Orgambide
Dr. Heitor Soares Ramos Filho

Maceió, Julho de 2019

Keila Barbosa Costa dos Santos

Categorização de Textos por Aprendizagem de Máquina

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas.

Orientadores: Dr. Alejandro Cesar Frery Orgambide.
Dr. Heitor Soares Ramos Filho.

Maceió, Julho de 2019

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecário Responsável: Marcelino de Carvalho

S237c Santos, Keila Barbosa Costa dos.
Categorização de textos por aprendizagem de máquina / Keila Barbosa Costa dos Santos. – 2019.
85 f. : il.

Orientador: Alejandro Cesar Frery Orgambide.
Coorientador: Heitor Soares Ramos Filho.
Dissertação (mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2019.

Bibliografia: f. 72-85.

1. Aprendizado de máquina. 2. Processamento de linguagem natural (Computação). . 3. Redes neurais (Computação). 4. Inteligência artificial. I. Título.

CDU: 004.85

Folha de Aprovação

Keila Barbosa Costa

Categorização de Textos por Aprendizagem de Máquina

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Modelagem Computacional de Conhecimento da Universidade Federal de Alagoas e aprovada em 10 de julho de 2019.

Prof. Dr. Alejandro Cesar Frery Orgambide
Instituto de Computação - UFAL
Orientador

Prof. Dr. Heitor Soares Ramos Filho
Departamento de Ciência da Computação - UFMG
Co-orientador

Banca Examinadora:

Prof. Dr. André Luiz Lins de Aquino
Instituto de Computação - UFAL
Examinador Interno

Prof. Dr. Jorge Artur Peçanha de Miranda Coelho
Faculdade de Medicina - UFAL
Examinador Interno

Prof. Dr. Osvaldo Anibal Rosso
Instituto de Física - UFAL
Examinador Externo

RESUMO

A categorização automática de textos em classes pré-definidas tem testemunhado um interesse crescente nos últimos anos, devido à maior disponibilidade de documentos em formato digital e à necessidade subsequente de organizá-los. Na comunidade de pesquisa, a abordagem dominante para esses problemas é baseada em técnicas de aprendizado de máquina: um processo indutivo geral constrói automaticamente um classificador aprendendo a partir de um conjunto de documentos pré-classificados, as características das categorias. As vantagens dessa abordagem sobre a abordagem de engenharia do conhecimento (consistindo na definição manual de um classificador por especialistas de domínio) são uma eficácia muito boa, economias consideráveis em termos de força de trabalho especializada e portabilidade direta para diferentes domínios. Esta pesquisa discute as principais abordagens para classificação de texto que se enquadram no paradigma de aprendizado de máquina. Discutiremos em detalhes questões relativas a representação de documentos, construção de classificadores e avaliação de classificadores assim como os métodos Deep Learning. As Redes neurais profundas (DNNs) revolucionaram o campo do Processamento Natural de Linguagem (PNL). As Redes Neural Convolucional (CNN) e Redes Neural Recorrente (RNN), são os dois principais tipos de Arquiteturas DNN amplamente exploradas para lidar com várias tarefas de PNL. A CNN é supostamente boa em extrair recursos de posição e variáveis e RNN na modelagem de unidades em sequência. O estado da arte em muitas tarefas de PNL geralmente mudam devido à batalha de CNNs e RNNs. Este trabalho é uma comparação entre os modelos clássicos de aprendizado de máquina (Maximum Entropy Modeling, Support Vector Machine, Bootstrap Aggregating, Boosting, Redes neurais da NNET, Random Forest, Análise discriminante linear escalada, Decision Trees e Naïve Bayes) e desta nova abordagem que encontra-se no estado da arte utilizando redes CNN e RNN, com objetivo principal a construção do índice da revista *IEEE Geoscience and Remote Sensing Letters*, observando a performance e o desempenho de diferentes modelos, a classificação é realizada a partir de dois conjuntos de dados (Título e Abstract) dos artigos da IEEEGRSL. Em contrapartida aos métodos tradicionais, introduzimos uma rede neural convolucional recorrente para classificação de texto a partir do Abstract dos artigos da revista por observar que os modelos clássicos tendem a perder precisão quando elevamos a quantidade de dados. Os resultados experimentais mostram que o método proposto tiveram uma performance satisfatória, porém a rede RCNN superou os métodos clássicos em desempenho. No entanto, ao implementar essa técnica de classificação utilizando Deep Learning, os índices de acerto para o conjunto de dados Abstract superou os modelos clássicos implementado neste trabalho, chegando a uma *precisão* de 94 % com uma performance de 6 segundos.

Palavras-chave: *Aprendizado de Máquina.* Processamento Natural de Linguagem. Aprendizado Profundo. Visão Computacional. Rede Neural Artificial. Redes Neurais Recorrentes. Redes Neural Convolucional.

ABSTRACT

Automatic categorization of texts into predefined classes has witnessed a growing interest in recent years, due to the increased availability of documents in digital format and the subsequent need to organize them. In the research community, the dominant approach to these problems is based on machine learning techniques: a general inductive process automatically builds a classifier learning from a set of pre-classified documents, the characteristics of categories. The advantages of this approach over the knowledge engineering approach (consisting of the manual definition of a classifier by domain experts) are very good effectiveness, considerable savings in terms of skilled workforce and direct portability to different domains. This research discusses the main approaches to text classification that fit the machine learning paradigm. We will discuss in detail issues relating to document representation, classifier construction, and classifier evaluation as well as the Deep Learning methods. Deep neural networks (DNNs) have revolutionized the field of Natural Language Processing (NLP). Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are the two main types of widely exploited DNN architectures to handle various NLP tasks. CNN is supposed to be good at extracting position and variable resources and RNN in sequence unit modeling. The state of the art in many NLP tasks usually changes due to the battle of CNNs and RNNs. This work is a comparison between the classic machine learning models (Maximum Entropy Modeling, Support Vector Machine, Bootstrap Aggregating, Boosting, NNET Neural Networks, Random Forest, Scaled Discriminant Analysis, Decision Trees and Naïve Bayes) and this new approach which is state of the art using CNN and RNN networks, with the main objective of building the index of the journal *IEEE Geoscience and Remote Sensing Letters*, observing the performance and the performance of different models, the classification is performed from of two datasets (Title and Abstract) from IEEEGRSL articles. In contrast to the traditional methods, we introduced a recurring convolutional neural network for text classification from the Abstract of the journal's articles by noting that classic models tend to lose accuracy when we increase the amount of data. The experimental results show that the proposed method had a satisfactory performance, but the RCNN network surpassed the classical methods in performance. However, when implementing this classification technique using Deep Learning, the hit ratios for the Abstract dataset surpassed the classic models implemented in this paper, reaching a *precision* of 94 % with a performance of 6 seconds.

Keywords: *Machine Learning*. Natural Language Processing. Deep Learning. Computer vision. Artificial Neural Network. Recurrent Neural Networks. Convolutional Neural Networks.

AGRADECIMENTOS

Quero agradecer a Deus que permitiu e me conduziu a alcançar mais esse sonho.

Dedico essa dissertação à minha família que sempre me apoiou e me incentivou durante todo esse processo.

Aos meus orientadores Professores Dr. Alejandro Cesar Frery Orgambide e Dr. Heitor Soares Ramos Filho, também ao professor Dr. André Luiz Lins de Aquino pela oportunidade, confiança, atenção concedidas durante o desenvolvimento deste trabalho, meu respeito e apreço.

A minha mãe Berenice Barbosa Silva, pelo exemplo de vida, dedicação, motivação, companheirismo e conselhos durante toda minha vida.

Aos amigos que sempre estiveram ao meu lado, me dando força, apoio, incentivo e acreditando na minha capacidade, em especial Kelly Costa e Antônio Medeiros.

Em especial dedico este trabalho a minha filha Khyonara Barbosa dos Santos e meu filho Arthur Gabriel Barbosa dos Santos, que deu um sentido especial à minha existência e me tem proporcionado grandes momentos.

Keila Barbosa Costa dos Santos

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1

INTRODUÇÃO

As categorizações de texto, ou as atribuições de textos de linguagem natural para categorias predefinidas com base no seu conteúdo, é de crescente importância à medida que o volume de informações na Internet continua a nos sobrecarregar. O uso de classes predefinidas implica em uma abordagem de "aprendizagem supervisionada", onde o artigo já classificado que efetivamente define as categorias é usado como "dados de treinamento"para construir um modelo que pode ser usado para classificar novos artigos.

Isso contrasta com aprendizado "não supervisionado", onde não há dados de treinamento e grupos de documentos semelhantes são procurados entre os artigos de teste. Com aprendizado supervisionado, rótulos significativos (como *keyphrases* e *keywords*) são anexados aos documentos de treinamento, e rótulos apropriados podem ser usado de forma automática para testar documentos, dependendo de qual categoria eles se encaixam (Frank et al. 2000).

A categorização de texto é um tópico importante no aprendizado de máquina. As abordagens típicas extraem "características"dos artigos e usam os vetores de atributos como entrada para um esquema de aprendizado que aprende a classificar os artigos. Os atributos geralmente são palavras. Como há muitos deles, um processo de seleção é aplicado para determinar os mais importantes, e o restante é descartado. Este modelo de *bag of words* negligência a ordem das palavras e o efeito do contexto. Também levanta alguns problemas: como definir uma "palavra", o que fazer com números e outras sequências não-alfabéticas, e se aplicar *stemming*.

Tem sido frequentemente observado que a Rede Neural de Convolução (CNN) e a Rede Neural Recorrente (RNN) parecem fornecer abordagens alternativas muito promissoras à categorização (Wang et al. 2016).

Em muitos sistemas de armazenamento e recuperação de texto cuidadosamente organizados, os textos são classificados em uma ou mais categorias escolhidas de um sistema de classificação. Exemplos incluem o NTIS (*National Technical Information Service*) de documentos do governo dos Estados Unidos da América, serviços de notícias como UPI e Reuters,

publicações como o *ACM Computing Reviews* e muitos outros.

A classificação manual de documentos é demorada e cara. Trabalhos recentes mostraram que em certos ambientes, sistemas baseados em conhecimento podem fazer classificações com rapidez e precisão (Hayes et al. 1990, Hayes & Weinstein 1991). Modelos baseados em regras de engenharia humana para atribuição de categorização de texto, enquanto relativamente eficazes, são também muito dispendiosos em termos de tempo e esforço para o seu desenvolvimento e atualização. Os métodos de aprendizagem de máquina fornecem uma alternativa interessante para automatizar o processo de construção de regras.

Este trabalho apresenta resultados em experimentos para derivar as regras de atribuição automaticamente a partir do texto a ser classificado. Um exemplo bem conhecido de um sistema baseado em conhecimento para a tarefa de classificação é o CONSTRUE (Hayes et al. 1990) usado pelo serviço de notícias da Reuters. Este é um sistema especialista baseado em regras construídas manualmente para atribuir categorias de assunto a notícias, têm relatado precisão de mais de 90 % em 750 casos de teste (Hayes & Weinstein 1991). Embora estes sejam resultados excepcionalmente bons, o conjunto de teste parece ter sido relativamente escasso quando comparado ao número de tópicos possíveis. A exemplo de um sistema de aprendizado de máquina para a mesma tarefa é um sistema baseado no raciocínio em memória (Masand et al. 1992), que emprega classificação de vizinhos mais próximos (*nearest neighbors*) e tem uma precisão relatada em um intervalo de 70 % até 80 % nas notícias da Dow Jones.

Ao considerar o problema de categorizar documentos, a abordagem baseada em regras tem um apelo considerável. Embora soluções como os métodos probabilísticos lineares usados por Weiss & Indurkhya (1993) ou vizinhos mais próximos (*nearest neighbors*), também possam se mostrar razoáveis, os modelos que eles empregam não são explicitamente interpretáveis. Como os sistemas de engenharia humana foram construídos com sucesso usando soluções baseadas em regras, seria mais útil continuar com um modelo compatível com o conhecimento expresso pelo homem. Dada a natureza parcimoniosa e interpretável das regras de decisão, podemos prontamente aumentar nosso conhecimento ou verificar as regras examinando documentos relacionados categoricamente.

Relatamos aqui os resultados obtidos com o uso de várias abordagens clássicas de Machine Learning (Maximum Entropy Modeling, Support Vector Machine, Boosting, Random Forest, Decision Trees e Naïve Bayes) e Deep Learning (Convolutional Neural Network e Recurrent Neural Networks). As coleções utilizadas neste trabalho são essencialmente Títulos e Abstract de artigos coletados entre os anos de 2004 até 2018 do periódico *IEEE Geoscience and Remote Sensing Letters*, contendo 17 categorias e 2830 artigos.

1.1 Contribuições

O *IEEE Geoscience and Remote Sensing Letters* (GRSL) [1] é uma publicação mensal de artigos curtos (máximo de 5 páginas) que aborda novas ideias e conceitos formativos em sensoriamento remoto, além de resultados de impacto para a comunidade de sensoriamento remoto. Os artigos devem relacionar-se com a teoria, os conceitos e as técnicas da ciência e da engenharia aplicados à percepção da Terra, dos oceanos, da atmosfera e do espaço, e ao processamento, interpretação e disseminação dessas informações. O conteúdo técnico dos artigos deve ser novo e significativo. Os dados experimentais devem ser completos e incluir descrição suficiente do aparato experimental, métodos e condições experimentais relevantes.

O seu índice é realizado de forma manual, mensalmente pelo Editor-Chefe. Ele analisa cada artigo, e o associa a uma categoria dentre as predeterminadas pelo Comitê Assessor da Sociedade Científica responsável pelo periódico. As categorias são muito mais estáveis do que os conteúdos. Neste contexto o presente trabalho tem como contribuição a categorização automática deste índice de forma a substituição de tarefas manuais, minimizar erros humanos, otimizar o tempo, reduzir os custos e focar na qualidade e precisão dos resultados.

Do ponto de vista científico, a pesquisa de classificadores de documentos com grande número de classes constitui uma área ativa e relevante. Embora muitas abordagens foram propostas, classificadores de texto e documentos ainda é uma área importante da pesquisa principalmente porque a eficácia dos classificadores automatizados não é impecável e ainda precisa de melhorias.

Realizando o levantamento na base Web of Science [2] foi encontrados 197 artigos que possuíam relevância em seu título nos anos de 2016 até 2018 referindo-se ao tema "*document classification*"e "*document categorization*". Além disso, existem inúmeros trabalhos que se referem ao tema em anos anteriores a 2016. Dessa forma, verifica-se que a área em questão possui relevância acadêmica.

1.2 Objetivo

1.2.1 Geral

Desenvolver de uma abordagem computacional para classificar automaticamente os documentos de texto em uma categoria predefinida usando aprendizado de máquina com ênfase em aprendizagem profunda (*Deep Learning*) e abrir esse modelo para que seja interpretado usando o *Local Interpretable Model-agnostic Explanations* (LIME) .

Ao construir modelos complexos, muitas vezes é difícil explicar por que o modelo deve ser confiável. Embora medidas globais como a precisão sejam úteis, elas não podem ser usadas

[1] Disponível em: https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=8859
[2] Disponível em: http://isiknowledge.com/

para explicar por que um modelo fez uma previsão específica. O LIME é um método para explicar o resultado de modelos de caixa preta ajustando um modelo local em torno do ponto em questão. A abordagem é descrita com mais detalhes no artigo de Ribeiro et al. (2016).

1.2.2 Específico

Classificar documentos de texto (conjunto de resumos de artigos) do periódico *IEEE Geoscience and Remote Sensing Letters* em suas 17 categorias. Observando o desempenho de diferentes modelos de aprendizado de máquina para classificação, como fruto será possível a construção de forma automática do índice da revista a partir do modelo com melhor desempenho, avaliando a capacidade dos algoritmos de aprendizado de máquina para categorização, fazendo uma comparativa do desempenho das técnicas clássicas de aprendizado de máquina e das técnicas de aprendizagem profunda e, por fim, a exploração do modelo de redes profundas com o LIME.

Este estudo tenta responder às seguintes questões de pesquisa (QP):

- QP1: Quão preciso oferece os algoritmos de aprendizado de máquina para classificação de documentos como os tratados neste trabalho?
- QP2: Como os algoritmos de aprendizado de máquina lidam com conjuntos de dados desbalanceados, que é o caso dos aqui tratado?

A modelagem é um processo da metodologia *Cross Industry Standard Process for Data Mining (CRISP-DM)* onde se definem as técnicas que auxiliam o processo de mineração de dados.

A modelagem consiste na aplicação de técnicas que objetivam encontrar padrões e descoberta do conhecimento. Aqui são utilizados algoritmos de aprendizagem de máquina, sendo que estes terão como entrada os dados pré-processados.

Bases de dados textuais fornecem dados semânticos e estatísticos em seu conteúdo. Diversas formas de abordagem aos dados textuais podem ser empregadas. As duas abordagens mais utilizadas são: a análise estatística, que é baseada na frequência de ocorrência dos termos nos textos, e a análise semântica, que é baseada na funcionalidade de cada termo no texto. As abordagens podem ser utilizadas sozinhas ou em conjunto. A modelagem proposta para a resolução da problemática deste trabalho aborda a aplicação de modelos estatísticos e técnicas de inteligência artificial.

A avaliação consiste em testar a efetividade do modelo aplicado. Usualmente essa análise é baseada em indicadores e métricas comparativas. É uma validação da adequação dos tratamentos aplicados aos dados e da modelagem escolhida.

1.3 Organização da Dissertação

Esta dissertação está organizada da seguinte forma:

No Capítulo 1 apresentamos a definição do problema, os objetivos e alguns conceitos básicos sobre classificação de documentos.

No Capítulo 2 vemos detalhes sobre os últimos trabalhos, o estado da arte no que diz respeito ao Processamento Natural de Linguagem, Inteligência Artificial, Aprendizagem de Máquina e também Aprendizagem Profunda para classificação de documentos.

No Capítulo 3 são apresentadas as metodologias abordadas na pesquisa.

No Capítulo 4 inicialmente são discutidos os testes e realizado uma avaliação e comparativa dos resultados obtidos.

No capítulo 5 descrevemos as conclusões deste trabalho, comentamos as nossas contribuições, e sugerimos direções futuras de pesquisa.

O capítulo apresentou a abordagem do problema foco nesta dissertação. Em seguida, foram apresentadas as contribuições e objetivos desta pesquisa. Dando sequência, o segundo capítulo apresentará a fundamentação base e aspectos cruciais ao desenvolvimento desta dissertação.

2

FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresentar os principais conceitos de mineração de textos e inteligência artificial abrangendo o processamento e preparação dos textos, os algoritmos de classificação utilizados e os métodos de avaliação dos resultados gerados. O levantamento do estado da arte leva em consideração artigos envolvendo classificação automática de textos e classificação com grande número de classes.

2.1 Mineração de Textos

Mineração de textos é o processo de descoberta de conhecimento que utiliza técnicas de análise e extração de dados a partir de textos, frases ou palavras. É o processo de extrair padrões interessantes e não triviais ou conhecimento a partir de documentos em textos não estruturados (Sebastiani 2002). Esta descoberta de conhecimento envolve diversas aplicações tais como análise de textos, extração de informações, sumarização, classificação, agrupamentos, linguística computacional, dentre outras.

Segundo Jurafsky & Martin (2014), a classificação de textos é uma das aplicações da mineração de textos e pode ser usada em alguns contextos como detecção de Spam [1], identificação dos autores, identificação de gênero (usando pronomes ou outros determinantes), análise de sentimentos, definição de categorias e identificação da linguagem em que foi escrito o texto.

No processo de classificação de textos tem-se como entrada um documento d e um conjunto fixo de classes $\mathbb{C} = (c1, c2, \ldots, cj)$. A saída será determinar a classe sobre à qual o documento d está semanticamente relacionado, ou seja, dado um documento, será assinalada a classe à que o mesmo pertence (Jurafsky & Martin 2014).

Os modelos de classificação podem ser divididos em *single-label* ou *multi-label* (Aly 2005). No *single-label*, cada documento pode pertencer a apenas uma classe. Nos modelos de

[1] Spam é um termo de origem inglesa cujo significado designa uma mensagem eletrônica recebida mas não solicitada pelo usuário.

classificação *multi-label*, um documento pode ser associado a uma ou mais categorias (Cerri et al. 2011). Esse tipo de classificação pode ser uma solução para a classificação de textos em larga escala.

A classificação de textos é constituída pelas seguintes etapas: coleta de dados, definição da abordagem, pré-processamento, um mecanismo de indexação, a aplicação do algoritmo de aprendizagem de máquina e a análise dos resultados (Korde & Mahender 2012).

Baseado no que se pretende analisar, define-se qual será abordagem a ser utilizada: semântica ou estatística (Wagner 2010).

2.2 Inteligência Artificial

Uma linha comum nas definições de "inteligência artificial"é que um agente autônomo executa ou recomenda ações (Russell & Norvig 2016, Poole et al. 1998).

O primeiro conjunto de esforços para estudar inteligência artificial (IA) foi o encontro no Darthmouth College [2], em 1956. No livro "Automata Studies"(Shannon & McCarthy 2016), o primeiro artigo tratava de redes neurais como um paradigma da arquitetura computacional. Pode-se dizer que aí nasceram simultaneamente os dois paradigmas da inteligência artificial: a simbólica e a conexionista.

Um breve histórico sobre Redes Neurais Artificiais deve começar por três das mais importantes publicações iniciais, desenvolvidas por: McCulloch & Pitts (1943), Hebb et al. (1949) e Rosenblatt (1958). Estas publicações introduziram o primeiro modelo de redes neurais simulando "máquinas", o modelo básico de rede de auto-organização, e o modelo perceptron de aprendizado supervisionado, respectivamente.

Alguns históricos sobre a área costumam omitir os anos 60 e 70, e apontar um reinício da área com a publicação dos trabalhos de Hopfield (1982) relatando a utilização de redes simétricas para otimização, e de Rumelhart et al. (1985) que introduziram o poderoso método Backpropagation.

Podemos observar em Amari et al. (1996), Anderson (1983), Reilly & Cooper (1995), Grossberg (1988), Fukushima (1988), Grossberg (1988), Miller et al. (1995) alguns trabalhos também bastante relevantes sobre modelos de redes neurais em visão, memória, controle e auto-organização.

2.3 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) é a subárea da inteligência artificial que estuda a capacidade e as limitações de uma máquina em entender a linguagem dos seres humanos. O objetivo do PLN é fornecer aos computadores a capacidade de entender e

[2] Disponível em: https://home.dartmouth.edu/

compor textos. "Entender"um texto significa reconhecer o contexto, fazer análise sintática, semântica, léxica e morfológica, criar resumos, extrair informação, interpretar os sentidos, analisar sentimentos e até aprender conceitos com os textos processados.

De acordo com Jackson & Moulinier (2007), o termo Processamento de Linguagem Natural é usado para descrever a função de *softwares* ou de componentes de *hardware* em um sistema computacional que analisam ou sintetizam linguagem falada ou escrita.

Segundo Jackson & Moulinier (2007), são diversas as aplicações de PNL. Algumas aplicações são listadas abaixo:

- Recuperação de documentos: descoberta de documentos que são considerados relevantes para uma consulta do usuário. Segundo o autor, esta é a aplicação principal na Internet atualmente. Cabe observar que usuários fazendo buscas na Internet estão efetivamente executando recuperação de documentos. A tendência tem sido em direção à crescente sofisticação na indexação, identificação e apresentação de textos relevantes.

- Roteamento de documentos: processo automático segundo o qual documentos que se enquadram em determinado critério são diretamente enviados para o usuário interessado.

- Classificação de documentos: estratégia na qual documentos são associados a categorias baseadas no seu conteúdo, sendo que comumente um mesmo documento pode fazer parte de diversas categorias.

- Indexação de documentos: processo de geração de índices de palavras ou frases vinculadas aos documentos nas quais elas ocorrem.

- Extração de informações: não está relacionada com a localização de um documento, mas com a obtenção de informação específica contida em um ou mais documentos.

- Sumarização de documentos: é um caso particular de extração de informações, na qual é pretendido extrair sentenças que resumem um documento.

Para que as atividades citadas anteriormente sejam possíveis, é imprescindível compreender a estrutura da linguagem utilizada. Segundo Gonzalez & Lima (2003), o Processamento de Linguagem Natural trata computacionalmente os diversos aspectos da comunicação humana, como sons, palavras, sentenças e discursos, considerando formatos e referências, estruturas e significados, contextos e usos.

Em sentido amplo, pode-se dizer que o PNL visa fazer com que os sistemas computacionais se comuniquem em linguagem humana, nem sempre em todos os níveis de entendimento e de geração de sons, palavras, sentenças e discursos. De acordo com Gonzalez & Lima (2003), tais níveis seriam os seguintes:

- Fonético e fonológico: que trabalham o relacionamento das palavras com os seus sons;

- Morfológico: abrange a construção das palavras a partir das unidades de significado primitivas e sua classificação em categorias morfológicas;
- Sintático: aborda o relacionamento das palavras entre si e como as frases podem ser partes de outras, construindo sentenças;
- Semântico: estuda o relacionamento das palavras com seus significados e como eles são combinados para formar os significados das sentenças;
- Pragmático: abrange o uso de frases e sentenças em diferentes contextos, afetando o significado.

Nas próximas duas seções, faremos um breve levantamento das técnicas de PNL relevantes tanto na área de conhecimento linguístico quanto de métodos estatísticos.

2.4 Aplicação do conhecimento linguístico

O conhecimento linguístico pode ser explorado através de técnicas de etiquetagem de texto, normalização de variações linguísticas e eliminação de *stopwords*.

Quando um conhecimento linguístico é considerado, a etiquetagem gramatical do texto é um dos passos iniciais. Conforme Gonzalez & Lima (2003), um etiquetador gramatical (*part-ofspeech tagger*) é um sistema que identifica, através de uma etiqueta (*tag*), a categoria gramatical de cada item lexical do texto analisado. Segundo Bick (1998), enquanto um etiquetador morfológico inclui informações sobre categorias morfológicas, como substantivos e adjetivos, um etiquetador sintático acrescenta etiquetas indicando as funções sintáticas das palavras, como sujeito e objeto direto.

Além da etiquetagem ou marcação gramatical, existe a etiquetagem semântica (Vieira 2000, Gonzalez & Lima 2003), que anexa informações relacionadas ao significado, podendo indicar os papéis dos itens lexicais na sentença, como agente, processo e estado. De forma genérica, o termo léxico significa uma relação de palavras com suas categorias gramaticais e seus significados. Em relação a um determinado idioma, um léxico é o universo de todos os seus itens lexicais, que seus falantes utilizam, já utilizaram ou poderão vir a utilizar (Scapini 1995, Gonzalez & Lima 2003).

Segundo Arampatzis et al. (2000) a normalização linguística pode ser subdividida em três casos distintos: morfológica, sintática e léxico-semântica. Cabe lembrar que utilizaremos neste trabalho somente a normalização morfológica.

As *stopwords* não fornecem nenhuma contribuição na identificação do conteúdo do texto. A remoção das *stopwords* tem como objetivo eliminar palavras que não são representativas ao documento e isso consequentemente diminui o número de palavras a serem analisadas, e também o número de palavras a serem armazenadas em uma base de busca de informações.

Stopwords são palavras frequentes em um texto e que não representam nenhuma informação de maior relevância para a extração de palavras-chave. Por exemplo: advérbios, artigos, conjunções, preposições e pronomes.

2.5 Aplicação de métodos estatísticos

Os métodos estatísticos têm dado grande contribuição ao Processamento de Linguagem Natural, como são os casos da lei de Zipf (1949) e do gráfico de Luhn como mostra a Figura 2.1.

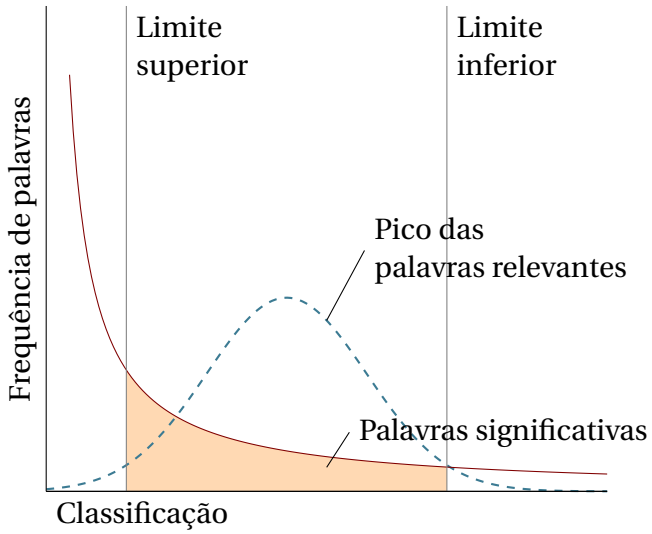


Figura 2.1: Lei de Zipf e cortes de Luhn

A lei, formulada por George Kingsley Zipf professor de linguística de Harvard, postula que a frequência de ocorrência de algum evento está relacionada a uma função de ordenação. Zipf (1949) mostrou que uma das características das linguagens humanas, populações das cidades e muitos outros fenômenos humanos e naturais, seguem uma distribuição similar, a qual denominou de "*Principle of Least Effort*".

Zipf, em 1949, estabeleceu o que ficou conhecida como *constant rank-frequency law of Zipf* (Moens 2006, Gonzalez & Lima 2003).

A Lei de Zipf em documentos de linguagem natural pode ser aplicada não apenas aos termos mas, também, a frases e sentenças da linguagem.

Esta lei define que, tomando um determinado texto, o produto $k_t \log(f_t)$ é aproximadamente constante, em que f_t é o número de vezes que o termo $_t$ ocorre no texto e k_t é a posição deste termo em uma relação de todos os termos daquele texto, ordenados pela frequência de ocorrência.

Por outro lado, Luhn sugeriu, em 1958, que a frequência de ocorrência das palavras em um texto pode fornecer uma medida útil sobre a expressividade das mesmas (Frants et al.

1997, Moens 2001, Gonzalez & Lima 2003), pois o "autor normalmente repete determinadas palavras ao desenvolver ou variar seus argumentos e ao elaborar diferentes enfoques sobre do que se trata o assunto". As palavras com maior frequência de ocorrência deveriam ser consideradas pouco expressivas porque este conjunto de palavras é composto normalmente por artigos, preposições e conjunções. Também as palavras que muito raramente ocorrem deveriam ser consideradas pouco expressivas justamente em razão da baixa frequência. Restam como expressivas as palavras com maior frequência de ocorrência intermediária.

A utilização da teoria de probabilidade e das abordagens estatísticas em geral, no Processamento Natural de Linguagem, apontam caminhos promissores para o entendimento do significado (Bod 1995). A teoria da probabilidade é relevante por gerar modelos matemáticos para frequências de ocorrência, e as abordagens estatísticas, por permitirem inferências valiosas sob incerteza.

Os métodos estatísticos podem ser utilizados para auxiliar o PNL em diversas situações. Eles têm sido utilizados na etiquetagem gramatical, na resolução de ambiguidade e na aquisição de conhecimento lexical (Krenn & Samuelson 1994), entre outras aplicações.

2.6 Conflação

Conflação é o ato de fusão ou combinação para igualar variantes morfológicas de palavras. A conflação pode ser manual, usando algum tipo de expressão regular, ou automática, via programas chamados radicalizadores (*stemmers*).

Para reduzir as variações de uma palavra para uma forma única utilizam-se técnicas de conflação. Segundo Jones (1997), Gonzalez & Lima (2003), existem dois métodos principais de conflação para se obter tal redução: radicalização e redução à forma canônica (tratada por alguns autores como **lematização**).

Radicalização (*stemming*) é o processo de combinar as formas diferentes de uma palavra em uma representação comum, o radical (*stem*) (Orengo et al. 2006). O radical é o conjunto de caracteres resultante de um processo de radicalização. Este não é necessariamente igual à raiz linguística, mas permite tratar variações de uma palavra da mesma forma. Por exemplo, "conector"e "conectores"são essencialmente iguais, mas sem sofrerem a redução por radicalização serão tratadas como palavras distintas.

Na língua inglesa, *stem* é o mesmo que radical ou tema. Raiz é *root*. Assim, segundo Dictionary (1986), a raiz ou parte principal de um substantivo, verbo, etc. a que inflexões são adicionadas; a parte que parece inalterada ao longo dos casos e derivados de um substantivo, pessoas de um tempo, etc.

Conforme Soukhanov et al. (1992): A parte principal de uma palavra na qual os afixos são adicionados.

De acordo com Jones (1997), Gonzalez & Lima (2003), a **Radicalização** consiste em reduzir

todas as palavras ao mesmo radical, por meio da retirada dos afixos (sufixos e prefixos) da palavra. Segundo Porter (1980), a **Radicalização** é o processo de remoção das terminações morfológicas e flexionais das palavras. O algoritmo foi desenvolvido especificamente para a língua inglesa e tem a característica de remover apenas os sufixos das palavras.

Redução à forma canônica é o ato de representar as palavras através do infinitivo dos verbos e masculino singular dos substantivos e adjetivos. Redução à forma canônica é tratada por alguns autores como **Lematização**. A palavra reduzida desta forma recebe a denominação de lema ou forma canônica.

De acordo com De Lucca & Nunes (2002), a redução à forma canônica difere fundamentalmente de **Radicalização**. Enquanto a redução à forma canônica existe puramente no contexto lexicográfico, enquanto a radicalização não. Assim, as estruturas são diferentes, embora eventualmente seus resultados possam ser semelhantes.

Em mineração de textos, técnicas de PNL são utilizadas principalmente na fase de pré-processamento. Tarefas como identificação de classes gramaticais de termos, reconhecimento de entidades e até mesmo redução da dimensionalidade de representação de documentos são auxiliadas por PNL.

Podemos ver alguns trabalhos também bastante relevantes sobre PNL em categorização de documentos Joachims (2002), Bekkerman et al. (2003), Lewis (1991, 1992), Stamatatos et al. (2000), Jacobs (1992), Dumais et al. (1998), Joachims (2002).

2.7 Aprendizado de Máquina

Aprendizado de Máquina (AM) é uma subárea da pesquisa muito importante em inteligência artificial, pois a capacidade de aprender é essencial para um comportamento inteligente. Aprendizado de Máquina estuda métodos computacionais para adquirir novos conhecimentos, novas habilidades e novos meios de organizar o conhecimento já existente (Mitchell et al. 1997).

O campo de aprendizado de máquina emprega algoritmos computacionais que melhoram automaticamente através de sua utilização, adquirindo experiência a partir de uma série de exemplos. Tipicamente isso ocorre em duas etapas. A primeira, onde é feito o treinamento, e a segunda, onde ocorre a classificação (Scavuzzo et al. 2018).

Após a seleção de transformação e extração de atributos, os documentos podem ser representados de uma forma que pode ser usada pelos algoritmos de *Machine Learning* (ML). Muitos classificadores de textos têm sido propostos na literatura usando técnicas de aprendizado de máquina, modelos probabilístico, etc. Eles frequentemente diferem na abordagem adotada: árvores de decisão (*decision trees*), naïve-Bayes, indução de regras (*rule induction*), redes neurais (*neural networks*), vizinhos mais próximos (*nearest neighbors*), e, mais recentemente, máquina de vetores de suporte (*support vector machines*).

Trabalhos mais recentes empregaram métodos de mineração de dados e aprendizado de máquina. Entre os mais precisos destas técnicas é a máquina de vetores de suporte (SVM) (Eskin et al. 2003, Joachims 1999). As SVMs usam funções kernel para encontrar hiperplanos separadores em espaços de alta dimensão. Outros métodos de kernel usados para a recuperação de informação incluem kernels de string como o *spectrum Kernel* (Leslie et al. 2001) e o *Mismatch Kernel* (Eskin et al. 2003), que são amplamente utilizados com dados de sequência de DNA e RNA.

SVM e métodos relacionados são difíceis de interpretar. Por essa razão, muitos sistemas de recuperação de informações usam os métodos de árvores de decisão (French et al. 1997) e naïve Bayes (McCallum et al. 1998, Kim et al. 2006). Esses métodos são mais fáceis de entender e, como tal, podem dar suporte à reformulação de consultas, mas frequentemente não atingem a precisão de outras abordagens.

Alguns trabalhos recentes de modelagem de tópicos são investigados para fornecer interpretações semelhantes como os métodos naïve Bayes, mas com maior precisão (Van Linh et al. 2017).

Naïve Bayes é frequentemente usado em classificação de texto, aplicações e experimentos por causa de sua simplicidade e eficácia (Vapnik 2013). No entanto, o desempenho é frequentemente degradado. Schneider (2005) abordou os problemas e mostrar que eles podem ser resolvidos por algumas simples correções (Schneider 2005).

Johnson et al. (2002) descreveu uma árvore de decisão rápida com algoritmo de construção que aproveita a escassez de dados de texto é um método de simplificação de regras que converte a árvore de decisão em um conjunto de regras logicamente equivalentes.

Lim (2004) propôs um método que melhora o desempenho da classificação do texto baseado em vizinhos mais próximos (kNN) usando parâmetros estimados.

No contexto da combinação de múltiplos classificadores para categorização de texto, um número de pesquisadores mostraram que a combinação de diferentes classificadores pode melhorar a precisão da classificação (Bao & Ishii 2002, Cho & Lee 2003, Bi et al. 2004, Nardiello et al. 2003).

O uso de técnicas de aprendizado de máquina tem sido amplamente investigado para classificação de texto (Lewis & Ringuette 1994, Koller & Sahami 1997, Yang & Liu 1999, Pang et al. 2002, Leopold & Kindermann 2002, Madsen et al. 2004, Kotsiantis et al. 2007, Lu & Tanne 2017, Arras et al. 2017, Kowsari et al. 2017, Lu & Tanne 2017, Ding & Salem 2018).

2.8 Deep Learning

Aprendizagem profunda é uma versão eficiente de redes neurais (Hinton & Salakhutdinov 2006) que pode executar aprendizagem não supervisionada, supervisionada e semi-supervisionada (Johnson & Zhang 2014). Aprendizagem profunda tem sido amplamente

usada para processamento de imagens, mas muitos estudos recentes têm aplicado a aprendizagem profunda em outros domínios como texto e mineração de dados. A arquitetura básica em uma rede neural é uma rede não-linear totalmente conectada de nós de processamento organizados em camadas. A primeira camada é a camada de entrada, a camada final é a camada de saída e todas as outras camadas estão escondidas.

Neste documento, vamos nos referir a estas redes totalmente conectadas como *Deep Neural Networks* (DNN).

Redes Neurais Convolucional (CNNs) são modeladas com inspiração na arquitetura do córtex visual, em que os neurônios não estão totalmente conectados mas são espacialmente distintos (LeCun et al. 1998). CNNs fornecem excelentes resultados na generalização da classificação de objetos em imagens (Oquab et al. 2014). Trabalhos mais recentes usaram CNNs para mineração de texto (Lee & Dernoncourt 2016). Pesquisas relacionadas com este trabalho (Zhang et al. 2015) usaram CNNs para classificação de texto em nível de caracteres fornecidos por um DNN totalmente conectado, independentemente de as CNNs exigirem grandes conjuntos de treinamento.

Outra arquitetura fundamental de aprendizagem profunda usada neste documento é a Rede Neural Recorrente (RNN) mostrada na Figura 2.2. RNNs conectam a saída de uma camada de volta à sua entrada. Essa arquitetura é particularmente importante para a aprendizagem de estruturas dependentes do tempo para incluir palavras ou caracteres no texto (Medsker & Jain 1999).

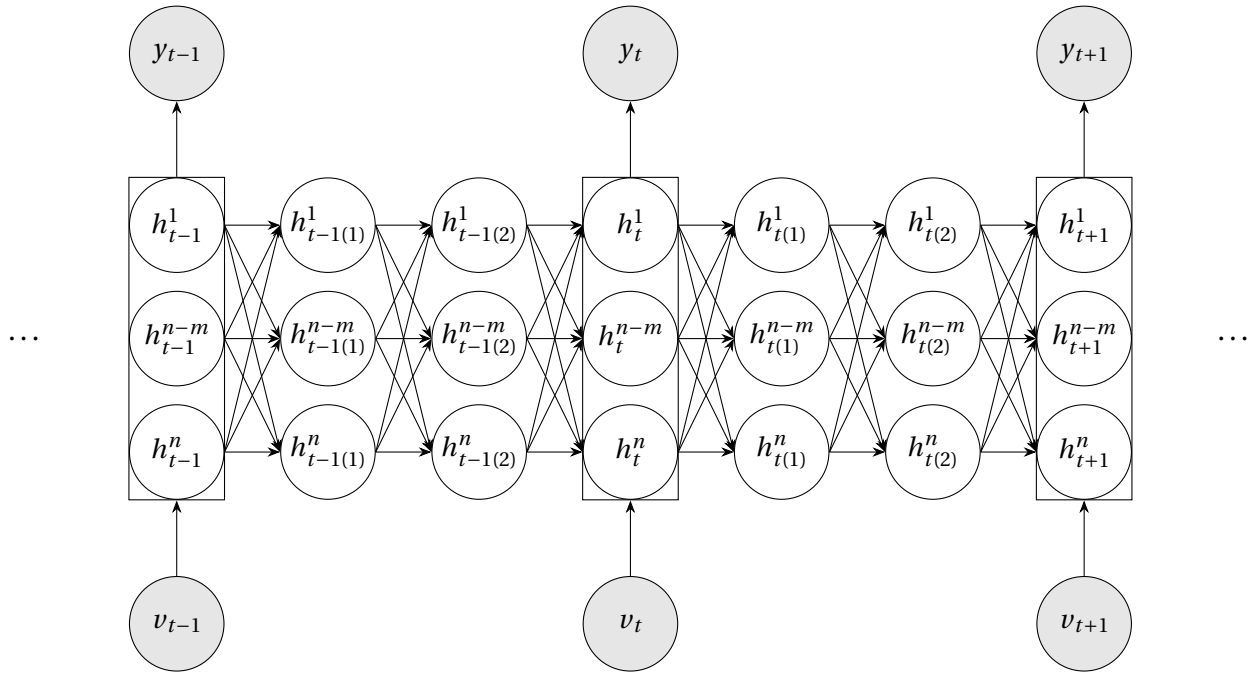


Figura 2.2: Redes Neurais Recorrentes (RNN)

O aprendizado profundo para classificação não é novo, embora as arquiteturas específicas, as análises comparativas e a aplicação à classificação de documentos sejam novas. Salakhutdi-

nov et al. (2013), Huang et al. (2012) usaram o aprendizado profundo para categorizar imagens hierarquicamente.

Podemos denotar alguns trabalhos de aprendizagem profunda para classificação de texto (Iyyer et al. 2015), análise de sentimento a críticas de filmes (Johnson & Zhang 2014), análise de opinião de resenhas em filmes, classificando sentenças como subjetivas ou objetivas, classificando tipos de perguntas, sentimento de resenhas de produtos, dentre outras (Kim 2014).

2.8.1 Redes Neurais Recorrentes

As *Redes Neurais Recorrentes* também conhecida como *Recurrent Neural Networks* (RNNs) geraram recentemente uma quantidade significativa de novidades no campo da aprendizagem profunda. Similarmente às redes convolucionais, elas existem há décadas, mas seu total potencial só começou a ser amplamente reconhecido recentemente, em grande parte devido aos crescentes recursos computacionais.

Podemos visualizar a arquitetura do modelo RNN na Figura 2.2.

No domínio de Processamento Natural de Linguagem as Redes Neurais Recorrentes transcrevem a fala para o texto (Graves & Jaitly 2014), executam tradução automática (Sutskever et al. 2014), geram texto manuscrito [3], e têm sido usados como poderosos de linguagem tanto no nível de caracteres quanto de palavras (Sutskever et al. 2011, Graves 2013, Le & Mikolov 2014). Atualmente, parece que os modelos em nível de palavras funcionam melhor que os modelos em nível de caractere.

As Redes Neurais Recorrentes também estão se tornando rapidamente difundidas na visão computacional. Podemos observar RNNs em classificação de vídeo em nível de quadro (Donahue et al. 2015), legenda de imagens (Vinyals et al. 2015), legendas de vídeo (Venugopalan et al. 2015), respostas visuais a perguntas (Ren et al. 2015) e modelos recorrentes de atenção visual (Mnih et al. 2014), tanto pela sua direção de alto nível (processamento sequencial de imagens) quanto pela modelagem de baixo nível (regra de aprendizado *reinforce* que é um caso especial de métodos de gradiente em aprendizado por reforço, que permite treinar modelos que realizam computação diferencial, isto é, que olha ao redor da imagem).

Podemos observar algumas limitações das redes recorrentes. Um problema é que as RNNs não são indutivas: elas memorizam sequências extremamente bem, mas elas não necessariamente mostram sinais convincentes de generalização da maneira correta. Um segundo problema é que elas associam desnecessariamente seu tamanho de representação à quantidade de computação por etapa.

O primeiro exemplo convincente foi desenvolvido no artigo de Graves et al. (2014) da *DeepMind's*. Este artigo esboçou um caminho para modelos que podem realizar operações de leitura e gravação entre grandes *arrays* de memória externa e um conjunto menor de

[3] Disponível em: http://www.cs.toronto.edu/ graves/handwriting.html

memória de registros. Crucialmente, o artigo também apresentou mecanismos de endereçamento de memória muito interessantes. O conceito de *soft attention* acabou se tornando um poderoso recurso de modelagem e também foi destaque em *Neural Machine Translation* por aprendizado em conjunto para alinhar e traduzir (Bahdanau et al. 2014, Sukhbaatar et al. 2015).

As redes neurais recursivas tiveram sucessos significativos em várias tarefas de PNL. Por exemplo, Socher et al. (2013) usaram uma rede neural recursiva para prever o sentimento da sentença.

Essencial para esse sucesso é o uso de *Long Short-Term Memory* (LSTMs), um tipo muito especial de rede neural recorrente que funciona, para muitas tarefas de PNL muito melhor do que a versão padrão no qual podemos observar na Figura 2.3 onde uma sequência de texto pode ser alimentada com caractere de cada vez (Elman 1990).

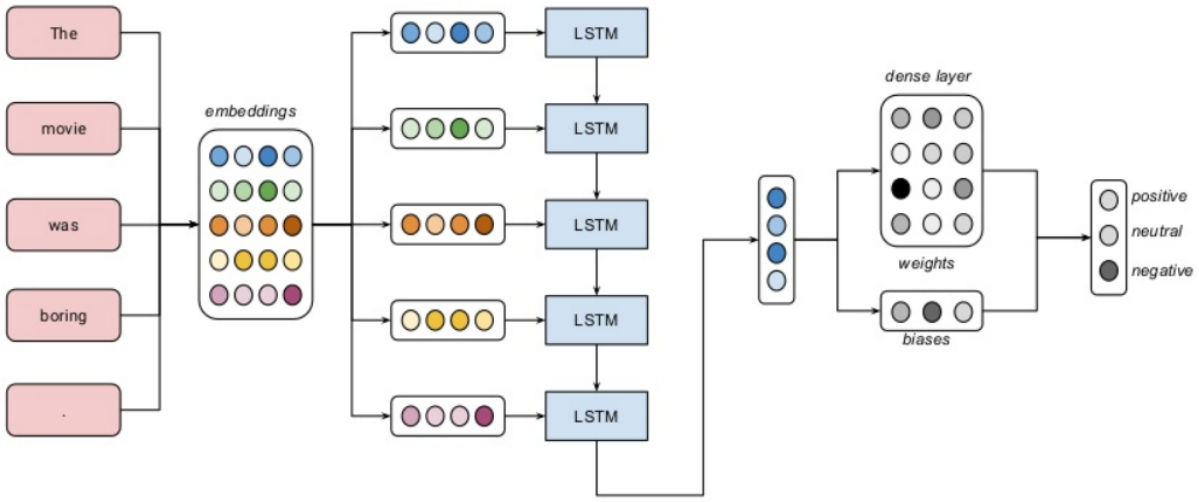


Figura 2.3: Arquitetura do Modelo Long Short-Term Memory (LSTM's) aplicado a modelagem de linguagem para classificação de texto.

2.8.2 Rede Neural Convolucional

Redes Neurais de Convolução também conhecida como *Convolutional Neural Networks* (CNN), utilizam camadas com filtros de convolução que são aplicados a características locais (LeCun et al. 1998). Originalmente inventados para visão computacional, os modelos CNN foram subsequentemente mostrados como eficazes para a PNL e alcançaram excelentes resultados na análise semântica (Yih et al. 2014), classificação de sentenças (Kim 2014), recuperação de consulta de pesquisa (Shen et al. 2014), modelagem de frases (Kalchbrenner et al. 2014) e outras tarefas tradicionais da PNL (Collobert et al. 2011).

A arquitetura original da rede neural convolucional, como introduzida por LeCun et al. (1989), alterna entre camadas convolucionais incluindo não-linearidades e camadas de subamostragem. Na Figura 2.4, as camadas convolucionais já incluem não-linearidades

e, assim, uma camada convolucional na verdade representa duas camadas. Os mapas de características da camada final de subamostragem são então inseridos no classificador real que consiste em um número arbitrário de camadas totalmente conectadas. A camada de saída geralmente usa as funções de ativação softmax.

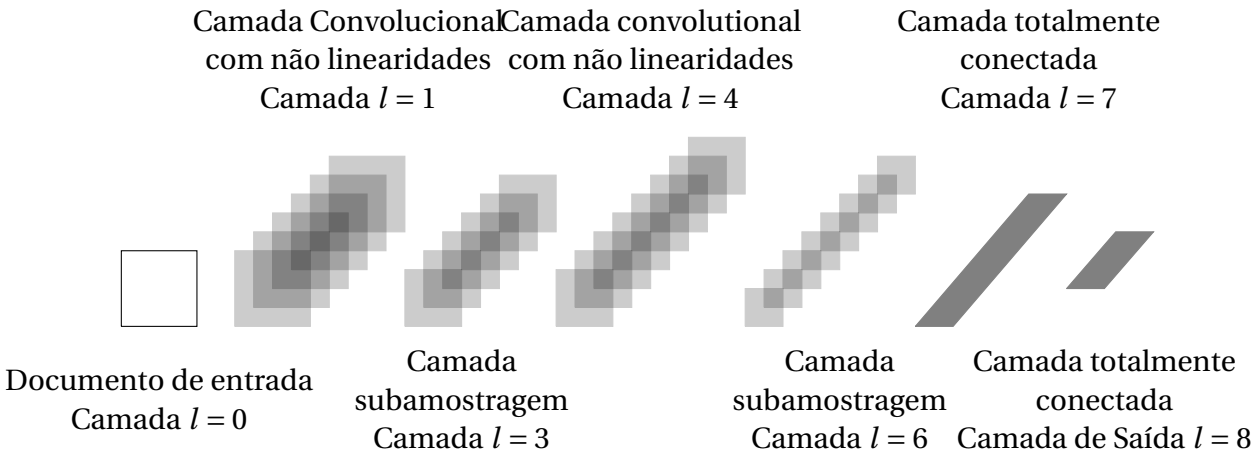


Figura 2.4: Arquitetura tradicional de uma Rede Neural Convolucional.

Podemos notar uma representação na Figura 2.5 de como a entrada é propagada para avaliar redes neurais profundas discretizadas com uma profundidade arbitrária de camadas ocultas para chegar a uma classificação.

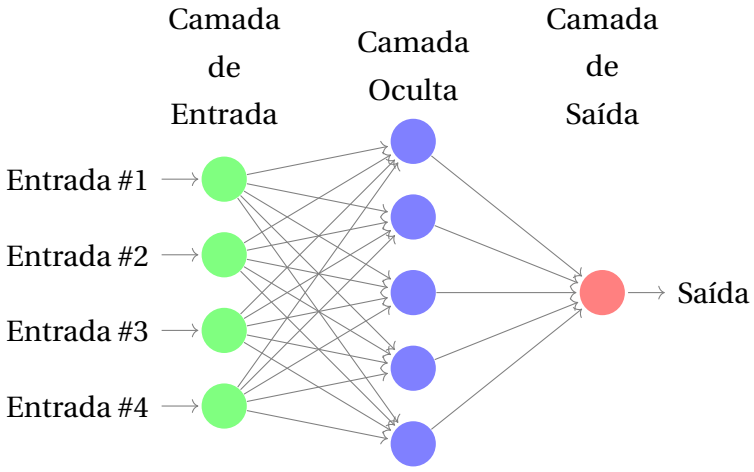


Figura 2.5: Modelo de uma Rede Neural Convolucional.

O modelo de neurônio representado pela Figura 2.6 recebe múltiplos sinais de outros neurônios através de seus dentritos, e cada um destes sinais é multiplicado pelo próprio peso da conexão. Estes sinais são adicionados no corpo celular ou função somatória, e quando este sinal composto alcança um valor umbral, um sinal potencial é enviado pelo axônio, o qual é a saída do neurônio Haykin (1994). Cada neurônio realiza operações, uma função linearmente dependendo dos valores dos pesos de entrada seguidos por operações não lineares.

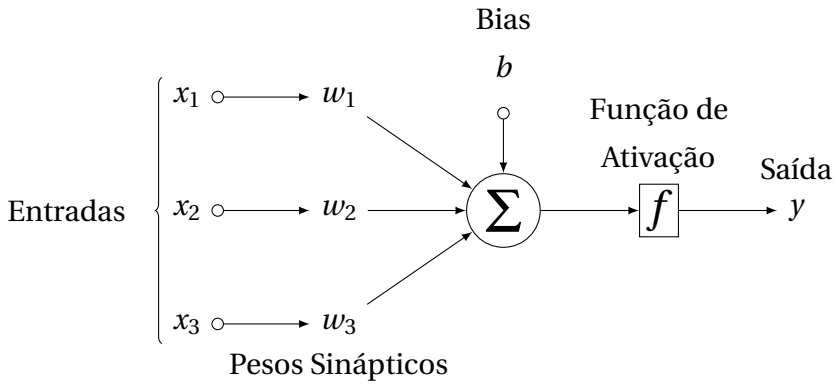


Figura 2.6: Modelo Básico de um Neurônio Artificial "Perceptron".

2.8.3 Word Embeddings

Os *embeddings* de palavras foram originalmente desenvolvidos por Bengio et al. (2003), alguns anos antes da renovação do aprendizado profundo de 2006, em um momento em que as redes neurais estavam fora de moda. A ideia de representações distribuídas para símbolos é ainda mais antiga (Hinton 1986).

Palavras semelhantes estão juntas. Outra maneira de chegar a isso é observar quais palavras estão mais próximas da incorporação de uma determinada palavra, podemos observar esta técnica no artigo de Turian et al. (2010), Mikolov et al. (2013).

O uso de representações de palavras tornou-se um "segredo secreto"para o sucesso de muitos sistemas de PNL nos últimos anos, em tarefas que incluem reconhecimento de entidades nomeadas, marcação de partituras, análise e rotulação de papéis semânticos (Luong et al. 2013). Aprender uma boa representação em uma tarefa e usá-la em outra tarefa é um dos principais truques da caixa de ferramentas *deep learning*. Um dos pontos fortes dessa abordagem é que ela permite que a representação aprenda com mais de um tipo de dados.

Há uma contrapartida. Em vez de aprender uma maneira de representar um tipo de dados e usá-lo para executar vários tipos de tarefas, podemos aprender uma maneira de mapear vários tipos de dados em uma única representação. Um bom exemplo disso é uma incorporação de palavras bilíngues, como apresentado por Zou et al. (2013). Podemos aprender a incorporar palavras de duas linguagens diferentes em um único espaço compartilhado.

Recentemente, o aprendizado profundo começou a explorar modelos que incorporam imagens e palavras em uma única representação. A ideia básica é que se classifiquem imagens produzindo um vetor em uma incorporação de palavras, podemos notar esta técnica nos trabalhos de Zou et al. (2013), Krizhevsky et al. (2012), Norouzi et al. (2013), Frome et al. (2013).

O presente capítulo elencou alguns fatores históricos e científicos sobre a mineração de texto, Inteligência Artificial, PNL, categorização de documentos, Aprendizado de Máquina, *Aprendizado Profundo* e suas principais abordagens. Além disso, apresentou os principais trabalhos relacionados sobre os temas. No próximo capítulo, apresentaremos alguns modelos propostos de visão computacional abordado em *Deep Learning* para suporte a *Classificação de Texto* que darão base a este trabalho.

3

MODELOS E METODOLOGIA

ESTE capítulo descreve em profundidade os métodos e técnicas, assim como os dados e ferramentas que foram usados para conduzir o estudo de categorização de texto e construção automática do índice da revista científica *IEEE Geoscience and Remote Sensing Letters*, abordando os modelos clássicos de aprendizado de máquina e *deep learning* que encontra-se no estado da arte.

3.1 Metodologia

O modelo de referência desta pesquisa é o *Cross Industry Standard Process for Data Mining* (CRISP-DM), que foi utilizado na etapa de mineração de textos (Wirth & Hipp 2000) e implementação dos modelos.

O Processo Padrão da Indústria Transversal para Mineração de Dados ou modelo (CRISP-DM), como é conhecido, é uma estrutura de processo para projetar, criar, construir, testar e implantar soluções de aprendizado de máquina.

Conforme mostra a Figura 3.1, adaptada de (Azevedo & Santos 2008), o CRISP-DM possui 6 fases a qual podemos ver com detalhes abaixo.

1. Compreensão empresarial (entender o negócio): entender o objetivo do projeto a partir de uma perspectiva de negócios, definindo um plano preliminar para atingir os objetivos. Como insumo dessa fase, se deve entregar três coisas o (*background*, objetivo do projeto e os critérios de sucesso).

2. Entender os dados: coletar, descrever usando estatísticas, explorar e verificar a qualidade do dado.

3. Preparação dos dados: *Data Selection, Data Cleaning, Construct Data e Integrating Data.* Normalmente ocorre várias vezes no processo.

4. Modelagem: Várias técnicas de modelagem são aplicadas, e seus parâmetros calibrados para otimização. Assim, é comum retornar à preparação dos dados durante essa fase. É nesse momento que se realiza a construção do modelo. Essa fase consiste na escolha do algoritmo, criar o modelo e tunar seus parâmetros.

5. Avaliar os resultados do modelo: Os critérios de sucesso definido na primeira fase precisam ser verificados se foram atingidos caso contrário é necessário voltar a primeira fase.

6. Implantação: O conhecimento adquirido pelo modelo é organizado e apresentado de uma maneira que o cliente/usuário possa utilizar.

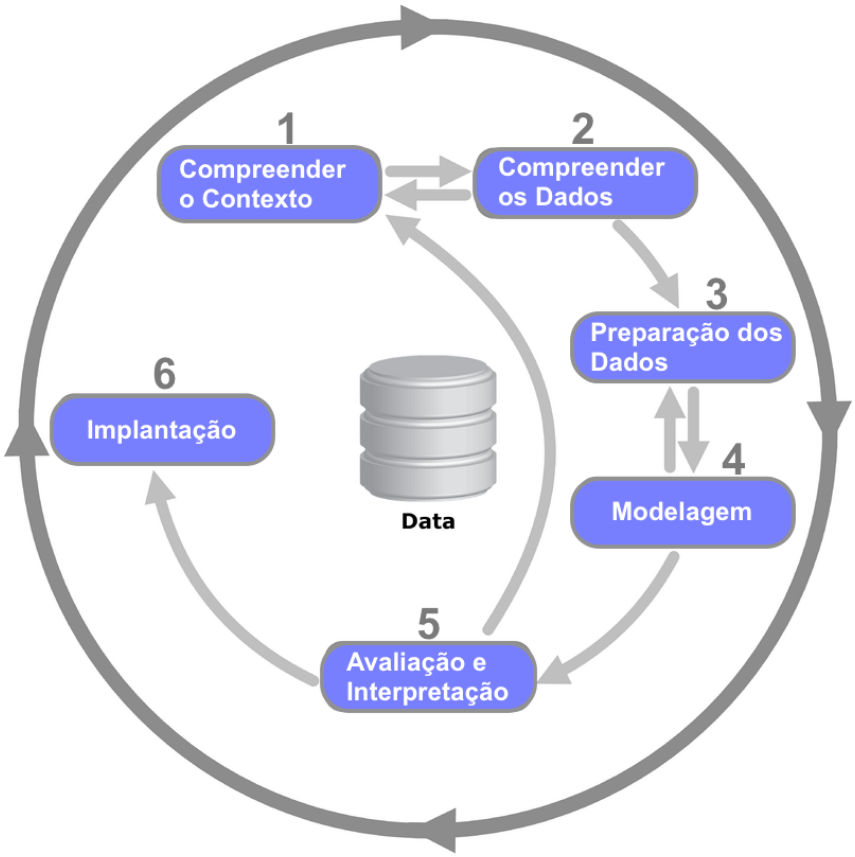


Figura 3.1: Diagrama CRISP-DM e a relação entre as diferentes fases do processo.

Para responder as questões proposta pela pesquisa conforme subseção 1.2.2, definimos uma abordagem composta por seis etapas baseada na metodologia (CRISP-DM):

1. Revisão bibliográfica e o delineamento de como alcançar os objetivos;

2. Entendimento, coleta, exploração e verificação da qualidade do dado;

3. Pré-processamento dos dados;

4. Proposição e implementação de modelos de Aprendizagem de Máquina;

5. Análise de resultados e testes;

6. Validação com a *IEEE Geoscience and Remote Sensing Letters*.

Em primeiro lugar, foi realizado o levantamento do estado da arte em mineração de textos e classificação objetivando encontrar possíveis soluções para o problema aqui abordado, este processo diz respeito a primeira fase da metodologia (CRISP-DM) Figura 3.1.

Para o treinamento dos Algoritmos para classificação de texto foi utilizada a seguinte plataforma:

- Ubuntu 16.4.4 LTS (GNU/Linux 4.13.0-41-generic x86-64)
- Model: 158
- Model name: Intel(R) Core(TM) i7-7700K CPU 4.20GHz

A análise, treinamento, testes, gráficos dos dados, foi realizada na plataforma *R* version 3.1.3 e 3.6.0 GUI 1.69 [1] que foi escolhida por ser do tipo *FLOSS – Free/Libre Open Source Software (Softwares licenciados que disponibilizam o código fonte de forma livre e permitem utilização, estudo e modificação)*, bem como pela sua excelente qualidade numérica quando comparada a outras (Almiron et al. 2010). Foram utilizadas as Ferramentas de *Deep Learning Keras, TensorFlow, TensorBoard* com *Keras* e *Local Interpretable Model-Agnostic Explanations* (LIME).

3.1.1 Pré-processamento dos Dados

O levantamento dos dados foi realizado de forma direta no site da revista. Foram coletados dados dos anos de 2004 à Agosto de 2018 através de *downloads* em formato BIBTEX[2] de cada revista mensal. A partir desta etapa foi construído o banco de dados. As variáveis explicativas escolhida para o processo formam 17 grupos e o número total de observações dentro de cada grupo de artigos pode ser vista na Figura 3.2. Estes critérios do processo diz respeito a segunda fase da metodologia (CRISP-DM) empregada neste trabalho conforme Figura 3.1.

O pré-processamento é um dos principais componentes em muitos algoritmos de mineração de texto. Por exemplo, uma estrutura tradicional de categorização de texto compreende pré-processamento, extração de atributos, seleção de atributos e etapas de classificação. Embora seja confirmado que a extração de características (Günal et al. 2006), seleção de atributos (Feng et al. 2012) e o algoritmo de classificação (Tan et al. 2011) têm impacto significativo no processo de classificação, a fase de pré-processamento pode ter influência perceptível sobre

[1] R. Disponível em https://www.r-project.org
[2] Disponível em http://www.bibtex.org/

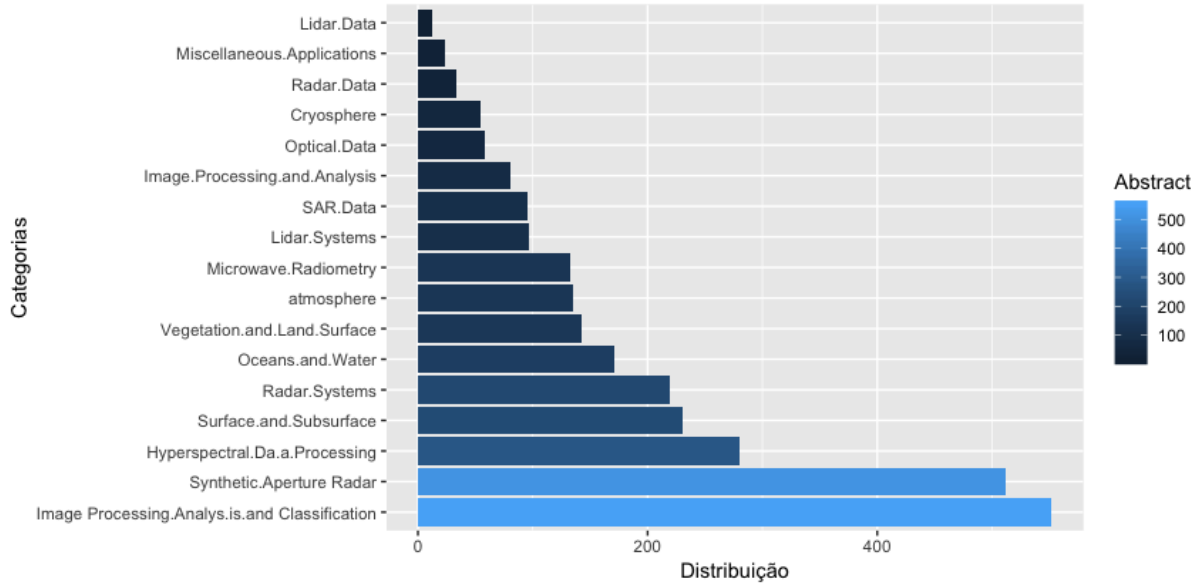


Figura 3.2: Distribuição da Base de Dados por Categoria.

este sucesso. Uysal & Gunal (2014) investigaram o impacto de tarefas de pré-processamento, particularmente na área de classificação de texto.

A etapa de pré-processamento realizada neste trabalho diz respeito a terceira fase da metodologia (CRISP-DM) conforme Figura 3.1 e consiste na tarefa de tokenização, transformação do texto em *DocumentTermMatrix*, *WordStemming*, remoção de números, remoção de *SparseTerms*, remoção de *Stopwords*, filtragem, lematização e *tolower*. A seguir, faremos uma breve descrição de cada uma delas.

Os textos passaram por um processamento conforme a Figura 3.3 usando os pacotes do *R* chamado *tm* (Feinerer 2013) e *RTextTools* (Collingwood et al. 2013). Estes pacotes possuem várias funções pré-definidas para tratamento e classificação de textos, dentre as quais foram utilizadas:

- *removeNumbers* para remoção de números;
- *removestripWhitespace* para remoção de espaços em branco;
- *removePunctuation* para remoção de pontuação;
- *contenttransformer(tolower)* para transformação das letras maiúsculas em minúsculas;
- *removeWords, Stopwords* ("en") para remoção de *stopwords*;
- *stemDocument* para redução das palavras aos seus radicais.

O Processo de classificação de documentos descrito no fluxo de trabalho na Figura 3.3 consistiu em ter uma base de dados como entrada, possuindo documentos definidos e

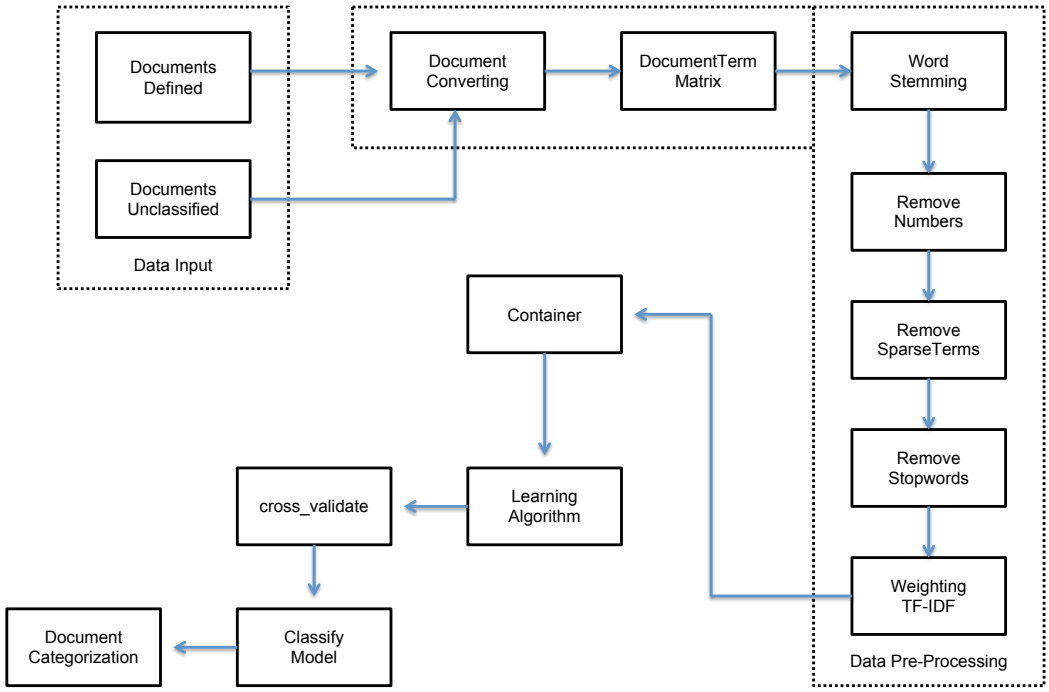


Figura 3.3: Processo de classificação de documentos.

documentos não classificados convertida e transformada em matriz de termos do documento (DTM).

A matriz de termos do documento (DTM) é um dos formatos mais comuns para representar um corpus de texto (ou seja, uma coleção de textos) em um formato de saco de palavras. Um DTM é uma matriz na qual as linhas são documentos, colunas são termos e células indicam com que frequência cada termo ocorreu em cada documento.

Após a conversão (DTM) os dados receberam a etapa de pré-processamento e o método de indexação TF-IDF descrito com mais detalhes na subseção 3.1.2. A etapa seguinte visa a implementação dos modelos (algoritmos de aprendizado de máquina) e treinamento onde foi construído o classificador propriamente dito. Tendo o conjunto de dados já pré-processado, divide-se esse conjunto em dois subconjuntos, tipicamente denominados conjunto de treinamento e conjunto de teste. O conjunto de teste foi usado para avaliar a performance do classificador, processo que será detalhado na seção 3.1.13, validando o modelo e resultando na classificação dos documentos, ou seja a saída dos documentos categorizados.

Também foi incorporada a este trabalho a tokenização que é a tarefa de quebrar sequência em pedaços (palavras/ frases) chamados tokens, e talvez ao mesmo tempo jogar fora certas sentenças como sinais de pontuação. A lista de tokens é então usada para processamento adicional (Webster & Kit 1992), esta etapa consiste em separar cada palavra no documento

inteiro em forma de token de acordo com a Figura 3.4. O Parâmetro *num_words* define o número máximo de palavras usada e classificadas pela maior sequência de frequência, palavras que raramente aparecem foram removidas, palavras únicas encontradas nos dados de texto foram usadas para criar o modelo.

```
num_words <- 2024
# prepare tokenizers
tokenizer <- text_tokenizer(num_words = num_words,
                            lower = TRUE) %>%
  fit_text_tokenizer(data$text)
paste("this letter presents a new algorithm that allows the retrieval of the aerosol optical depth aod at a high
 m spatial resolution from landsat operational land imager oli data over urban areas because of the complex struc
ture over urban surfaces the bidirectional reflectance characteristic is obvious however most of the current aero
sol retrieval algorithms over land do not account for the anisotropic effect of the surface this letter improves
 the quality of aod retrieval by providing the surface reflectance based on the multiyear modis bidirectional ref
lectance distribution function brdfalbedo model parameters product mcd laughing a and the rossthicklisparse recip
rocal kerneldriven brdf model the groundbased aerosol robotic network aeronet aod measurements from five sites lo
cated in urban and suburban areas are used to validate the aod retrievals and the modis terra collection c dark t
argetdeep blue aod products mod at km spatial resolution are obtained for comparison the validation results show
 that the aod retrievals from the oli images are well correlated with the aeronet aod measurements r = with a low
rootmeansquare error of a mean absolute error of laughing and a relative mean bias of approximately laughing of t
he collocations fall within the excuse me tongue sticking out ected error the analysis indicates that the brdf is
essential in ensuring the accuracy of aod retrieval compared with the mod aod retrievals the oli aod retrievals h
ave better spatial continuity and higher accuracy the new algorithm can provide continuous and detailed spatial d
istributions of the aod over complex urban surfaces", length(tokenizer$word_counts))
```

```
## [1] "this letter presents a new algorithm that allows the retrieval of the aerosol optical depth aod at a high
m spatial resolution from landsat operational land imager oli data over urban areas because of the complex struct
ure over urban surfaces the bidirectional reflectance characteristic is obvious however most of the current aeros
ol retrieval algorithms over land do not account for the anisotropic effect of the surface this letter improves t
he quality of aod retrieval by providing the surface reflectance based on the multiyear modis bidirectional refle
ctance distribution function brdfalbedo model parameters product mcd laughing a and the rossthicklisparse recipro
cal kerneldriven brdf model the groundbased aerosol robotic network aeronet aod measurements from five sites loca
ted in urban and suburban areas are used to validate the aod retrievals and the modis terra collection c dark tar
getdeep blue aod products mod at km spatial resolution are obtained for comparison the validation results show th
at the aod retrievals from the oli images are well correlated with the aeronet aod measurements r = with a low ro
otmeansquare error of a mean absolute error of laughing and a relative mean bias of approximately laughing of the
collocations fall within the excuse me tongue sticking out ected error the analysis indicates that the brdf is es
sential in ensuring the accuracy of aod retrieval compared with the mod aod retrievals the oli aod retrievals hav
e better spatial continuity and higher accuracy the new algorithm can provide continuous and detailed spatial dis
tributions of the aod over complex urban surfaces 16242"
```

Figura 3.4: Processo de Tokenização de documentos da IEEEGRSL.

Existem muitas listas de tais palavras que são removidas como uma tarefa de pré-processamento. Essas palavras inúteis (por exemplo, preposições, conjunções, etc) formam uma lista chamada de *stopwords*. Nem todas as palavras apresentadas em um documento podem ser usada para treinar o classificador neste caso é feito uma filtragem removendo algumas dessas palavras (Madsen et al. 2004). Da mesma forma, palavras que ocorrem com bastante frequência no texto dizem ter pouca informação para distinguir diferentes documentos e também palavras que ocorrem muito raramente também não têm relevância significativa e podem ser removidas dos documentos (Silva & Ribeiro 2003, Saif et al. 2014).

A *lematização* é a tarefa que considera a análise morfológica das palavras, isto é, agrupando as várias formas flexionadas de uma palavra para que possam ser analisadas como um único item. Em outras palavras, os métodos de *lematização* tentam mapear as formas verbais para o tempo infinito e substantivos para uma única forma. Para lematizar os documentos, primeiro precisamos especificar o *POS-tagging* (etiquetagem de classe gramatical) de cada

palavra dos documentos e, como o *POS-tagging* é propenso a erros, na prática, os métodos de stemming são preferidos.

Aplicou-se também a função *removeSparseTerms* com o objetivo de reduzir o número de dimensões. A redução da dimensionalidade, desde que não impacte na eficácia do classificador, gera ganhos na medida em que melhora o desempenho computacional.

O *stemming* é outro passo comum de pré-processamento aplicado em nossos dados. Para reduzir o tamanho do conjunto inicial de atributos e remover erros ortográficos ou palavras com o mesmo haste. Um *stemmer* (um algoritmo que executa stemming), remove palavras com o mesmo radical e mantém a raiz ou o mais comum deles como característica. Embora o *stemming* seja considerado pela comunidade de texto e classificação para amplificar o desempenho de classificadores, há algumas dúvidas sobre a real importância do *stemming*, como a realizada pelo Porter Stemmer (Madsen et al. 2004).

O primeiro algoritmo *stemming* foi introduzido por Lovins (1968), mas o *stemmer* publicado por Porter (1980) é o método mais amplamente utilizado em Inglês (Hull 1996).

Text Representation se refere à codificação de dados textuais em variáveis numéricas. Neste trabalho foi utilizado o modelo *Vector Space Model* (VSM), também referenciado como Bag of Words, uma maneira de representar numericamente elementos textuais. Este modelo em sua forma mais simples representa um documento como um vetor binário, com cada posição desse vetor representando uma palavra no vocabulário, descrito com mais detalhes na próxima sessão.

3.1.2 Espaço Vetorial (VSM)

Um documento é uma sequência de palavras (Leopold & Kindermann 2002). Cada documento é geralmente representado por uma matriz de palavras. O conjunto de todas as palavras de um conjunto de treinamento é chamado vocabulário ou conjunto de atributos. Então um documento pode ser apresentado por um vetor binário, atribuindo o valor 1 se o documento contiver a palavra-atributo, ou 0 se a palavra não aparecer no documento.

Para permitir descrições mais formais dos algoritmos, primeiro definimos alguns termos e variáveis que serão usados com frequência. Dada uma coleção de documentos $\mathbb{D} = \{d_1, d_2, \ldots, d_D\}$, seja $\mathbb{V} = \{w_1, w_2, \ldots, w_v\}$ o conjunto de palavras/termos distintos na coleção. Então $\mathbb{V}$ é chamado de vocabulário. A frequência do termo $w \in \mathbb{V}$ no documento $d \in \mathbb{D}$ é denotada $f_d(w)$, e o número de documentos com a palavra é representado por $f_D(w)$. O termo vetor para o documento d é denotado por $\vec{t_d} = (f_d(w1), f_d(w2), \ldots, f_d(wv))$.

A maneira mais comum de representar documentos é convertê-los em vetores numéricos. Essa representação é chamada de "Modelo de Espaço Vetorial"(VSM). Embora sua estrutura seja simples e originalmente introduzida para indexação e recuperação de informações (Salton et al. 1975), o VSM é amplamente usado em vários algoritmos de mineração de texto; plataformas como *R* permitem uma análise eficiente da grande coleção de documentos

(Hotho et al. 2005).

No VSM, cada palavra é representada por uma variável com um valor numérico que indica o peso (importância) da palavra no documento. Existem dois modelos principais de pesos a termo:

- Neste modelo, um peso $w_{ij} > 0$ é atribuído a cada termo $w_i \in d_j$. Para qualquer termo que não apareça em d_j, $w_{ij} = 0$.
- Frequência de documentos com frequência inversa (TF-IDF).

Seja q o termo de esquema de ponderação, então o peso de cada palavra é calculado da seguinte forma:

$$q(w) = f_d(w) \cdot \log \frac{|D|}{f_D(w)}, \tag{3.1}$$

em que $|D|$ é o número de documentos na coleção D.

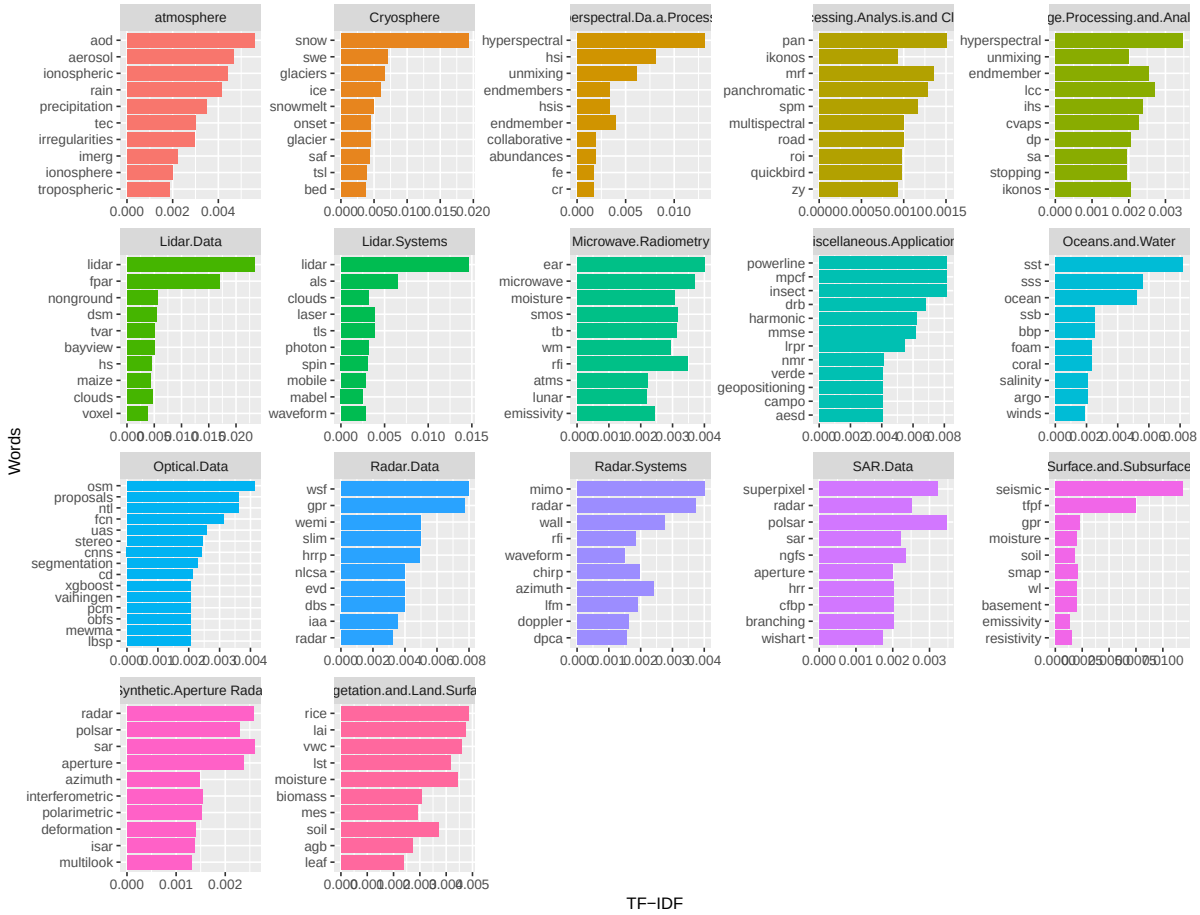


Figura 3.5: TF-IDF da variável Resumo para cada categoria.

No TF-IDF (abreviação do inglês *term frequency–inverse document frequency*) a frequência do termo é normalizado pela frequência inversa do documento, IDF. Essa normalização diminui o peso dos termos que ocorrem com mais frequência na coleção de documentos,

garantindo que a correspondência de documentos seja mais afetada por palavras distintas que têm frequências relativamente baixas na coleção. Com base no esquema do termo de ponderação, cada documento é representado por um vetor de pesos $w(d) = (w(d,w_1), w(d,w_2),\ldots,w(d,w_v))$.

Podemos, assim, calcular a similaridade entre dois documentos d_1 e d_2. Uma das medidas de similaridade mais utilizadas é a semelhança de cosseno e é calculada da seguinte forma:

$$S(d_1,d_2) = \cos\theta = \frac{d_1 \cdot d_2}{\sqrt{\sum_{i=1}^{v} w_{1i}^2} \cdot \sqrt{\sum_{i=1}^{v} w_{2i}^2}}. \tag{3.2}$$

Usamos TF-IDF para identificar palavras que são importantes para cada categoria da revista IEEEGRSL, conforme Figura 3.5.

3.1.3 Modelagem

A classificação de texto foi amplamente estudada em diferentes comunidades, como mineração de dados, banco de dados, aprendizado de máquina e recuperação de informações, e usada em um grande número de aplicações em vários domínios, como processamento de imagens, diagnóstico médico, organização de documentos etc. (Mitchell et al. 1997).

O problema da classificação é definido da seguinte forma. Dado um conjunto de treinamento $\mathbb{D} = \{d_1, d_2, \ldots, d_n\}$ de documentos, de modo que cada documento seja rotulado com um rótulo ℓ_i do conjunto $\mathbb{L} = \{\ell_1, \ell_2, \ldots, \ell_k\}$. A tarefa é encontrar um modelo de classificação (classificador) f em que:

$$f\colon D \to L, \qquad\qquad f(d) = \ell, \tag{3.3}$$

isto é, f cumpre a tarefa de atribuir um rótulo de classe ao novo documento d (instância de teste). A classificação é chamada de *hard* se um rótulo for explicitamente atribuído à instância de teste, e *soft* se um valor de probabilidade de pertinência for atribuído à instância de teste.

Existem outros tipos de classificação que permitem a atribuição de múltiplos rótulos (Gopal & Yang 2010) a uma instância de teste. Para uma visão abrangente de vários métodos de classificação, ver Duda et al. (2012), Gopal & Yang (2010). Yang & Liu (1999) avalia vários tipos de algoritmos de classificação de texto. Muitos dos algoritmos de classificação foram implementados em diferentes sistemas de software e estão disponíveis publicamente, como *BOW toolkit* (McCallum 1996), *Mallet* (McCallum 2002) e *WEKA* [3].

Algoritmos de aprendizado de máquina (AM) têm sido amplamente utilizados em diversas tarefas, que podem ser organizadas de acordo com diferentes critérios. Um deles diz respeito ao paradigma de aprendizado a ser adotado para lidar com a tarefa. De acordo com esses paradigmas foi escolhido o aprendizado supervisionado (classificação e regressão) para este

[3] Disponível em http://www.cs.waikato.ac.nz/ml/weka/

trabalho por se tratar de dados rotulados.

Em tarefas de aprendizado supervisionado, a meta é encontrar uma função a partir dos dados de treinamento que possa ser utilizada para prever um rótulo ou valor que caracterize um novo exemplo, com base nos valores de seus atributos de entrada. Para isso, cada objeto do conjunto de treinamento deve possuir atributos de entrada e saída.

O Treinamento é a etapa em que o sistema é alimentado com um conjunto de testes previamente classificados e organizados. O resultado desta etapa é um modelo de dados que permite executar a segunda etapa a classificação.

A Classificação é a etapa em que o modelo construído é utilizado para avaliar automaticamente novos documentos, classificando-os sem intervenção humana. Os métodos de classificação são os procedimentos que efetivamente classificam o documento em determinada classe.

Os algoritmos ou métodos de AM utilizados nessa tarefa induzem modelos preditivos. Esses algoritmos seguem o paradigma de aprendizado supervisionado. O termo supervisionado vem da simulação da presença de um "supervisor externo", que conhece a saída (rótulo) desejada para cada exemplo. Com isso o supervisor externo pode avaliar a capacidade de hipótese induzida de predizer o valor de saída para novos exemplos.

Neste trabalho, para a classificação automática dos textos baseado no *Título* dos artigos foi aplicado o método de indexação TF-IDF e também os algoritmos: Máxima Entropia (Maximum Entropy Modeling - MaxEnt) (Jurka 2012); Máquina de vetores de suporte (Support Vector Machine - SVM) (Dimitriadou et al. 2008); Agregação por Bootstrap (Bootstrap Aggregating - Bagging) (Peters et al. 2002); Boosting (Tuszynski 2012); Redes neurais da NNET (Venables & Ripley 2002); Floresta Aleatória (Random Forest - RF) (Liaw et al. 2002); Análise discriminante linear escalada (SLDA) (Peters et al. 2002); Árvore de Decisão (Decision Trees - TREE) (Venables & Ripley 2002); Naïve Bayes - NB (Bayes 1763).

Para a classificação automática dos textos baseado no *Abstract* dos artigos foi aplicado o método de indexação TF-IDF e também os algoritmos: *Maximum Entropy Modeling, Support Vector Machine, Boosting, Random Forest, Decision Trees e Naïve Bayes*.

Esta fase consistiu na construção dos modelos baseados nos algoritmos de classificação e também na construção da Rede Neural de Convolução (Convolutional Neural Network - CNN ou ConvNet) e da Rede Neural Recorrente (Recurrent Neural Networks - RNN). Esta etapa de construção dos modelos dá ênfase a quarta fase da metodologia (CRISP-DM) descrita na Figura 3.1.

Dessa forma, após efetuados os devidos tratamentos dos textos, os dados foram divididos em conjuntos de treinamento 70 % e teste 30 %. Os modelos foram testados utilizando os algoritmos citados nas próximas subseções.

3.1.4 Maximum Entropy Modeling

A classificação de *Maximum Entropy Modeling* (abreviadamente, MaxEnt, ou ME) é uma técnica alternativa que provou ser eficaz em várias aplicações de processamento de linguagem natural (Berger et al. 1996). No trabalho de Nigam et al. os autores mostram que, às vezes mas não sempre, o MaxEnt supera o naïve Bayes na classificação de texto padrão. Sua estimativa de $P(c \mid d)$ assume a seguinte forma exponencial:

$$P_{ME}(c \mid d) = \frac{1}{Z(d)} \exp\left(\sum_i \lambda_{i,c} F_{i,c}(d,c)\right), \tag{3.4}$$

em que $Z(d)$ é uma função de normalização, F_{ic} é uma função de atributo/classe para o atributo f_i e classe c, definida da seguinte forma:

$$F_{i,c'}(d,c0) = \begin{cases} 1, & n_i(d) > 0 \quad \text{e} \quad c' = c \\ 0 & \text{caso contrário.} \end{cases} \tag{3.5}$$

Os escalares λ_{ic} são parâmetros de atributos de peso; A inspeção da definição de P_{ME} mostra que um grande λ_{ic} significa que f_i é considerado um indicador forte para a classe c. Os valores dos parâmetros são definidos de modo a maximizar a entropia da distribuição induzida (daí o nome de classificador) sujeito à restrição de que os valores esperados das funções classe em relação ao modelo são iguais aos seus valores esperados em relação ao dados de treinamento: a filosofia subjacente é que devemos escolher o modelo que faz o menor número de suposições sobre os dados ainda consistente.

É importante ressaltar que, diferentemente de naïve Bayes visto na subseção 3.1.11, o MaxEnt não faz suposições sobre os relacionamentos entre os atributos e, portanto, poderia ter melhor desempenho quando a independência condicional não é verificada.

3.1.5 Máquina de Vetores de Suporte

Os algoritmos de máquina de vetores de suporte também conhecido como *Support Vector Machine* (SVM) foram propostos como um sistema de aprendizagem automática baseado na teoria estatística da aprendizagem (Vapnik 2013). Este método tem chamado a atenção de pesquisadores devido a seu desempenho bem-sucedido em diferentes áreas, como reconhecimento facial, categorização textual, previsões, recuperação de imagens e reconhecimento de caligrafia. Uma das excelentes características é a sua excelente capacidade de generalização, mesmo em espaços de alta dimensão e com pequenos conjuntos de treinamento (Palacio et al. 2018). São algoritmos de classificação de aprendizado supervisionados que foram utilizados extensivamente em problemas de classificação de texto.

SVM é uma forma de classificadores lineares. Classificadores lineares de documentos no contexto de texto são modelos que fazem uma decisão de classificação com base no valor

das combinações lineares dos recursos dos documentos. Assim, a saída de um preditor linear é definida como sendo $y = \vec{a} \cdot \vec{x} + b$, em que $\vec{x} = (x_1, x_2, \ldots, X_n)$ é o vetor de frequência de palavras do documento normalizado, $\vec{a} = (a_1, a_2, \ldots, a_n)$ é vetor de coeficientes e b é um escalar.

Podemos interpretar o preditor $y = \vec{a} \cdot \vec{x} + b$ nos rótulos das classes categóricas como um hiperplano de separação entre diferentes classes.

O SVM foi inicialmente introduzido por Cortes & Vapnik (1995), Vapnik (2006). As máquinas de vetores de suporte tentam encontrar um bom separador linear entre as várias classes. Um único SVM só pode separar duas classes, uma classe positiva e uma negativa (Hotho et al. 2005). O algoritmo SVM tenta encontrar um hiperplano com a distância máxima ξ (também chamada margem) dos exemplos positivos e negativos. Os documentos com distância ξ do hiperplano são chamados de vetores de suporte e especificam a localização real do hiperplano. Se os vetores do documento das duas classes não forem linearmente separáveis, um hiperplano é determinado de tal forma que o menor número de vetores de documentos estão localizados no lado errado.

Uma vantagem do método SVM é que ele é bastante robusto para alta dimensionalidade, isto é, a aprendizagem é quase independente da dimensionalidade do espaço de características. Ele raramente precisa de seleção de recursos, pois seleciona pontos de dados (vetores de suporte) necessários para a classificação (Hotho et al. 2005). Joachims (1998) descreveu que os dados de texto são uma escolha ideal para classificação de SVM devido à natureza esparsa de alta dimensionalidade do texto com poucos recursos irrelevantes.

Os métodos SVM têm sido amplamente utilizados em muitos domínios de aplicação, como reconhecimento de padrões, detecção de faces e filtragem de spam (Drucker et al. 1999, Osuna et al. 1997).

3.1.6 Boosting

O *Boosting* é uma das abordagens de aprendizado de máquina que combina muitos classificadores fracos e moderadamente precisos em um conjunto altamente preciso (Tuszynski & Orphaned 2013).

Começamos definindo um classificador (em geral, um preditor) fraco e um classificador forte (*weak leamer* e *strong leaner*, respectivamente). Suponha que existe um espaço X e uma função desconhecida $D_o(x) \in \{-1,1\}$ que designa uma classe para cada vetor de entrada. O problema é estimar a função (em aprendizado de máquina) D_o, que mapeia $X\, a \{-1, 1\}$.

O Boosting chama repetidamente um determinado classificador fraco para finalmente produzir a hipótese f, que é uma combinação linear de hipóteses K produzidas pelos classificadores fracos anteriores,

$$f(x) = sgn(\sum_{k=1}^{K} \alpha_k h_{(t_k,y_k)}(x)) \tag{3.6}$$

Um classificador fraco é criado a cada iteração k com diferentes distribuições ou pesos:

$$d^{(k)} = (d_i^{(k)}, \ldots d_L^{(k)}) \, (onde, \sum_{i=1}^{N} d_i^{(k)} = 1, d_i^{(k)} \geq 0) \tag{3.7}$$

Os pesos são calculados de maneira que exemplos concretos sejam focados em exemplos mais que fáceis.

De acordo com Kudo & Matsumoto para usar decisão de stumps o classificador fraco de Boosting, aplica a função de ganho da seguinte maneira:

$$gain(\langle t,y \rangle) =^{def} \sum_{i=1}^{L} y_i d_i h_{\langle t,y \rangle}(x_i)) \tag{3.8}$$

3.1.7 Redes neurais da NNET

As redes neurais da NNET também mostraram desempenho competitivo nas avaliações de categorização de texto (Yang & Liu 1999, Yang 1999). O pacote NNET criado por Ripley & Venables fornece métodos para o uso de redes neurais *feed-forward* com uma única camada oculta e para modelos *log-linear* multinomiais.

Uma rede neural *feedforward* é uma rede neural artificial em que as conexões entre as unidades não formam um ciclo dirigido. A rede neural *feedforward* foi o primeiro e provavelmente mais simples tipo de rede neural artificial criada. Nesta rede, a informação move-se em apenas uma direção, para frente, dos nós de entrada, através dos nós ocultos (se houver) e para os nós de saída. Não há ciclos ou loops na rede como as RNN.

As Redes Neurais são compostas de camadas de muitas funções preditivas simples que são conectadas via pesos. Esses pesos são determinados comparando repetidamente a saída da rede com o conjunto de dados de treinamento e o ajuste. Essa coleção de funções preditivas é frequentemente comparada à maneira como os neurônios do cérebro estão conectados para tomar decisões complexas. Usamos o pacote *nnet* (Ripley & Venables 2011) para construir o modelo de redes neurais.

3.1.8 Random Forest

Neste trabalho, usamos também o classificador *Random Forest* (RF) (Breiman 2001) para classificação de documentos. O RF é um algoritmo de aprendizado de máquina que constrói um conjunto baseados em árvore e, em seguida, classifica novos pontos de dados, obtendo um voto das previsões de cada classificador.

Existem vários motivos para selecionar o RF em relação a outros classificadores para esse problema. O RF foi mostrado para funcionar bem quando muitos atributos (na ordem de

milhares) estão disponíveis. Ele não se encaixa com o aumento no número de atributos e aumenta a diversidade entre os classificadores, amostrando os dados e alterando os conjuntos de atributos sobre os diferentes classificadores (árvores). A seleção aleatória dos atributos para dividir cada nó torna-o mais robusto para dados com ruído.

Os gráficos da variável de importância obtidos no estágio de treinamento podem ser usados para analisar quais regiões e características são importantes para a classificação. A importância da variável é estimada observando-se como a exatidão da predição diminui quando os dados *out-of-bag (OOB)* para essa variável são permutados, enquanto todos os outros permanecem inalterados (Breiman 2001).

3.1.9 Alocação de Dirichlet latente supervisionada

Introduzimos a Alocação de Dirichlet latente supervisionada também conhecida como *supervised latent Dirichlet allocation* (sLDA), um modelo estatístico de documentos rotulados que usa o mesmo mecanismo probabilístico que um modelo linear generalizado para acomodar vários tipos de resposta: valores reais irrestritos, valores reais restritos para serem positivos (por exemplo, tempo de falha), rótulos de classe ordenados ou não ordenados, inteiros não negativos (por exemplo, dados de contagem), e outros tipos (Mcauliffe & Blei 2008).

O sLDA é um modelo estatístico de documentos rotulados no qual cada documento é emparelhado com uma resposta e dado um documento não marcado, o objetivo é inferir sua estrutura de tópico usando um modelo ajustado e então formar sua previsão.

Onde $\alpha = [\alpha 1, ..., \alpha K]$, com K = dimensionalidade da distribuição de Dirichlet (número de tópicos) assumida como sendo conhecida, $\overline{z_d} = \frac{1}{N}\sum_{n=1}^{N} z_{d,n}, \beta = [\beta 1, ..., \beta K]$ com cada $\{\beta_k\}_{k=1}^{K}$ sendo uma distribuição sobre o vocabulário, α, β, η e σ^2 são tratados como constantes desconhecidas a serem estimadas.

3.1.10 Decision Trees

A árvore de decisão também conhecida como *Decision Trees* (TREE) é basicamente uma árvore hierárquica das instâncias de treinamento, na qual uma condição no valor do atributo é usada para dividir os dados hierarquicamente. Em outras palavras, a árvore de decisão (Friedl & Brodley 1997) particiona recursivamente o conjunto de dados de treinamento em subdivisões menores com base em um conjunto de testes definidos em cada nó ou ramificação. Cada nó da árvore é um teste de algum atributo da instância de treinamento, e cada ramo descendente do nó corresponde a um valor desse atributo. Uma instância é classificada iniciando no nó raiz, testando o atributo por esse nó e descendo a ramificação da árvore correspondente ao valor do atributo na instância especificada. E este processo é recursivamente repetido (Pereira et al. 2009).

No caso de dados de texto, as condições dos nós da árvore de decisão são comumente definidos em termos nos documentos de texto. Por exemplo, um nó pode ser subdividido em seus filhos, dependendo da presença ou ausência de um termo específico no documento.

Para uma discussão detalhada das árvores de decisão, ver Breiman (2017), Duda et al. (2012), Quinlan (1986).

Árvores de decisão têm sido usadas em combinação com outras técnicas. Freund & Schapire (1997), Schapire & Singer (2000) discutem técnicas de reforço para melhorar a precisão da classificação da árvore de decisão.

3.1.11 Naïve Bayes

Os classificadores probabilísticos ganharam também muita popularidade recentemente e mostraram um desempenho notável (Sebastiani 2002, Joachims 1996, Koller & Sahami 1997, Larkey & Croft 1996, Sahami et al. 1998). Essas abordagens probabilísticas fazem suposições sobre como os dados (palavras em documentos) são gerados e propõem um modelo probabilístico com base nessas suposições. Em seguida, usamos o conjunto de treinamento para estimar os parâmetros do modelo. A regra de Bayes é usada para classificar novos exemplos e selecionar a classe que provavelmente gerou o exemplo (McCallum et al. 1998).

O classificador naïve Bayes é talvez o classificador mais simples e mais amplamente utilizado. Modela a distribuição dos documentos em cada classe usando um modelo probabilístico, assumindo que a distribuição de diferentes termos é independente um do outro. Mesmo que essa suposição assim chamada "Bayes ingênuo"seja claramente falsa em muitas aplicações do mundo, Bayes ingênuo funciona surpreendentemente bem com muitos parâmetros.

Existem dois modelos principais comumente usados para classificações Bayes (McCallum et al. 1998). Ambos os modelos visam encontrar a probabilidade posterior de uma classe, com base na distribuição das palavras no documento. A diferença entre esses dois modelos é que um modelo leva em conta a frequência das palavras, enquanto o outro não:

1. Modelo de Bernoulli multivariado: neste modelo um documento é representado por um vetor de recursos binários que denota presença ou ausência das palavras no documento. Então, a frequência das palavras é ignorada. O trabalho original pode ser encontrado em Lewis (1998).

2. Modelo Multinomial: Capturamos as frequências das palavras (termos) em um documento representando o documento como de palavras. Muitas variações diferentes do modelo multinomial foram introduzidos em Kalt & Croft (1996), Sebastiani (2002), Nigam et al. (1998). McCallum et al. (1998) fizeram uma extensa comparação entre Bernoulli e modelos multinomiais e concluíram que:

- Se o tamanho do vocabulário é pequeno, o modelo de Bernoulli pode superar o modelo multinomial.
- O modelo multinomial sempre supera o modelo Bernoulli para grandes tamanhos de vocabulário, e quase sempre funciona melhor do que Bernoulli se o tamanho do vocabulário escolhido for ótimo para ambos os modelos.

Ambos os modelos supõem que os documentos são gerados por um modelo de mistura parametrizado por θ. Nós usamos a estrutura proposta por McCallum et al. (1998), definida a seguir.

O modelo de mistura compreende componentes de mistura $cj \in \mathbb{C} = \{c_1, c_2, \ldots, c_k\}$. Cada documento $d_i = \{w_1, w_2, \ldots, w_{ni}\}$ é gerado pela primeira seleção de um componente de acordo com as prioridades, $P(c_j \mid \theta)$ e depois usar o componente para criar o documento de acordo com seus próprios parâmetros, $P(d_i \mid c_j;\theta)$. Portanto, podemos calcular a probabilidade de um documento usando a soma das probabilidades sobre todos os componentes da mistura:

$$P(d_i \mid \theta) = \sum_{j=1}^{k} P(c_j \mid \theta) P(d_i \mid c_j;\theta). \tag{3.9}$$

Assumimos uma correspondência de um para um entre as classes $\mathbb{L} = \{\ell_1, \ell_2, \ldots, \ell_k\}$ e componentes de mistura e, portanto, c_j indica o componente de mistura j e a classe j. Consequentemente, dado um conjunto de exemplos de treinamento rotulados, $\mathbb{D} = \{d_1, d_2, \ldots d_{|D|}\}$, primeiro aprendemos (estimamos) os parâmetros do modelo de classificação probabilística, $\hat{\theta}$, e depois usando as estimativas desses parâmetros, realizamos a classificação dos documentos de teste calculando as probabilidades posteriores de cada classe c_j, dado o documento de teste, e selecionamos a classe mais provável (classe com a maior probabilidade):

$$P(c_j \mid d_i;\hat{\theta}) = \frac{P(c_j \mid \hat{\theta}) P(d_i \mid c_j;\hat{\theta}_j)}{P(d_i \mid \hat{\theta})} = \frac{P(c_j \mid \hat{\theta}) P(w_1, w_2, \ldots, w_{ni} \mid c_j;\hat{\theta}_j)}{\sum_{c \in C} P(w_1, w_2, \ldots, w_{ni} \mid c;\hat{\theta}_c) P(c \mid \hat{\theta})}, \tag{3.10}$$

em que, baseado na suposição de Bayes, as palavras em um documento são independentes umas das outras, portanto:

$$P(w_1, w_2, \ldots, w_{ni} \mid c_j;\hat{\theta}_j) = \prod_{i=1}^{n_i} P(w_i \mid c_j;\hat{\theta}_j). \tag{3.11}$$

3.1.12 Bootstrap Aggregating

Bootstrap Aggregating (Bagging) é um método para melhorar os resultados de algoritmos de classificação de aprendizado de máquina. Este método foi formulado por Breiman (1996) e seu nome foi deduzido da frase "*bootstrap aggregating*".

Em caso de classificação em duas classes possíveis, um algoritmo de classificação cria um classificador $H\colon D \rightarrow \{-1,1\}$ na base de treinamento de um conjunto de exemplos de

descrições (no nosso caso, desempenhado por uma coleção de documentos) D. O método de bagging cria uma sequência de classificadores $H_m, m = 1,\ldots,M$ em relação às modificações do conjunto de treinamento. Esses classificadores são combinados em um classificador composto. A previsão do classificador composto é dada como uma combinação ponderada de previsões classificadoras individuais:

$$H(d_i) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m H_m(d_i)\right). \tag{3.12}$$

O significado da fórmula acima pode ser interpretado como um procedimento de votação. Um exemplo d_i é classificado para a classe a qual a maioria dos classificadores particulares votam. Os artigos de Breiman (1997), Schapire et al. (1998) descrevem a teoria do voto classificador. Os parâmetros α_m, $m = 1,\ldots,M$ são determinados de forma que os classificadores mais precisos tenham maior influência que os classificadores menos precisos. A precisão dos classificadores de base H_m pode ser apenas um pouco maior que a precisão de uma classificação aleatória. É por isso que esses classificadores H_m são chamados de classificadores fracos.

Se for possível influenciar o procedimento de aprendizagem realizado diretamente pelo classificador H_m, o erro de classificação pode ser minimizado também por H_m, mantendo os parâmetros α_m constante.

O algoritmo descrito acima representa uma abordagem chamada versão base do bagging. Existem algumas outras estratégias chamadas de estratégias de bagging que funcionam com tamanho menor de conjunto de treinamento. Essas estratégias usam uma combinação do método de bagging e o método de validação cruzada.

A validação cruzada descrita com mais detalhes na próxima subseção representa a divisão do conjunto de treinamento em N subconjuntos de tamanho $[D/N]$. Um desses subconjuntos é usado como o conjunto de treinamento e os outros subconjuntos desempenham o papel de conjuntos de teste.

Em "estratégias de bagging", o conjunto de treinamento original é dividido em N subconjuntos do mesmo tamanho. Cada subconjunto é usado para criar um classificador. Um classificador específico é aprendido usando esse subconjunto. Um classificador composto é criado como a agregação de classificadores específicos.

3.1.13 Métricas de Avaliação

Em nossos experimentos, usamos o método de validação cruzada (*Cross-Validation*) que é uma técnica na qual visa entender como o modelo generaliza. A validação cruzada é usada principalmente no aprendizado de máquina aplicado para estimar a habilidade de um modelo de aprendizado de máquina em dados não vistos. Nesse caso, ao invés de usarmos apenas um conjunto de teste para validar nosso modelo, usamos N outros a partir dos mesmos dados.

Em sua versão básica, a chamada validação cruzada *k- fold*, envolve dividir aleatoriamente o conjunto de observações em k grupos, ou dobras, de tamanho aproximadamente igual. A primeira dobra é tratada como um conjunto de validação e o método é ajustado nas dobras restantes de $k-1$ (James et al. 2013).

O modelo foi ajustado usando todas as amostras, exceto o primeiro subconjunto. Em seguida, o erro de previsão do modelo ajustado é calculado usando as primeiras amostras retidas. A mesma operação é repetida para cada dobra e o desempenho do modelo é calculado pela média dos erros nos diferentes conjuntos de teste. Geralmente o k é fixado em 5 ou 10, mas não há regra formal. À medida que k aumenta, a diferença de tamanho entre o conjunto de treinamento e os subconjuntos de reamostragem fica menor. À medida que essa diferença diminui, o viés da técnica se torna menor (Kuhn & Johnson 2013). Aplica-se o modelo seguido pela técnica K-Fold e compara o resultado do erro médio de cada divisão do grupo. Para nosso modelo ele foi fixado em 4 por obter a menor taxa de erro médio de cada divisão do grupo. Isso significa que foi embaralhado os dados e dividido em quatro grupos. Os modelos são descartados depois de serem avaliados, uma vez que cumpriram sua finalidade.

A validação cruzada fornece uma estimativa do erro de teste para cada modelo sendo um dos métodos mais utilizados para seleção de modelos e para a escolha de valores de parâmetros de ajuste.

Para avaliar o desempenho do modelo de classificação, definimos uma fração aleatória dos documentos rotulados (conjunto de teste). Após o treinamento do classificador com o conjunto de treinamento, classificamos o conjunto de testes e comparamos os rótulos estimados com os rótulos verdadeiros e medimos o desempenho.

		Condição verdadeira				
	População total	Condição positiva	Condição negativa	Prevalência = $\frac{\Sigma \text{ Condição positiva}}{\Sigma \text{ População total}}$	Precisão (ACC) = $\frac{\Sigma \text{ Verdadeiro positivo} + \Sigma \text{ Verdadeiro negativo}}{\Sigma \text{ População total}}$	
Condição prevista	Condição previsível positiva	**Verdadeiro positivo**	**Falso positivo**, erro tipo I	Valor preditivo positivo (VPP), Precisão = $\frac{\Sigma \text{ Verdadeiro positivo}}{\Sigma \text{ Condição previsível positiva}}$	Taxa de descoberta falsa (FDR) = $\frac{\Sigma \text{ Falso positivo}}{\Sigma \text{ Condição previsível positiva}}$	
	Previsão previsível negativa	Erro **falso negativo**, tipo II	**Verdadeiro negativo**	Taxa de falsa omissão (FOR) = $\frac{\Sigma \text{ Falso negativo}}{\Sigma \text{ Condição negativa prevista}}$	Valor preditivo negativo (NPV) = $\frac{\Sigma \text{ Verdadeiro negativo}}{\Sigma \text{ Condição previsível negativa}}$	
		Taxa positiva verdadeira (TPR), Recall, Sensibilidade, probabilidade de detecção, Potência = $\frac{\Sigma \text{ Verdadeiro positivo}}{\Sigma \text{ Condição positiva}}$	Taxa Falsa Positiva (FPR), Fall-out, probabilidade de falso alarme = $\frac{\Sigma \text{ Falso positivo}}{\Sigma \text{ Condição Negativa}}$	Razão de verossimilhança positiva (LR +) = $\frac{\text{TPR}}{\text{FPR}}$	Odds ratio de diagnóstico (DOR) = $\frac{\text{LR +}}{\text{LR–}}$	F_1 pontuação = $2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		Taxa negativa falsa (FNR), taxa de faltas = $\frac{\Sigma \text{ Falso negativo}}{\Sigma \text{ Condição positiva}}$	Especificidade (SPC), Selectividade, Taxa negativa real (TNR) = $\frac{\Sigma \text{ Verdadeiro negativo}}{\Sigma \text{ Condição negativa}}$	Razão de verossimilhança negativa (LR–) = $\frac{\text{FNR}}{\text{TNR}}$		

Figura 3.6: Tabela de Indicadores de avaliação.

Para tarefas de classificação, os termos verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos comparam os resultados do classificador em teste com julgamen-

tos externos confiáveis. Os termos positivo e negativo referem-se à predição do classificador, e os termos verdadeiro e falso referem-se a se essa previsão corresponde ao julgamento externo como mostrado na Figura 3.6.

A parte dos documentos classificados corretamente para o número total de documentos é chamada de precisão (Hotho et al. 2005).

Aggarwal & Zhai (2012) definem as métricas da seguinte forma: "precisão é a fração das instâncias corretas entre as instâncias positivas identificadas. Recall (recordação) é a porcentagem de instâncias corretas entre todas as instâncias positivas. E o F-Score é a média geométrica da precisão e Recall"como mostrado nas equações 3.13, 3.14 e 3.15.

Neste trabalho foi usado os indicadores:

- PRECISÃO - é definida como o número de previsões que são verdadeiras positivas dividido pelo número de todas as amostras que são previstas como positivas. No campo de recuperação de informações, precisão é a fração de documentos recuperados que são relevantes para a consulta.

$$precision = \frac{|\{\text{Documentos relevantes}\} \cap \{\text{Documentos recuperados}\}|}{|\{\text{Documentos recuperados}\}|} \tag{3.13}$$

- RECOVERY/ RECALL - a taxa de recuperação é o número de casos positivos reais dividido pelo número de todas as amostras positivas verdadeiras. Na recuperação de informações, recall é a fração dos documentos relevantes que são recuperados com sucesso.

$$recall = \frac{|\{\text{Documentos relevantes}\} \cap \{\text{Documentos recuperados}\}|}{|\{\text{Documentos relevantes}\}|} \tag{3.14}$$

- F-SCORE - É a medida que combina precisão e recall é a média harmônica de precisão e recall, a tradicional medida F ou o F-score balanceado. O valor F leva em conta tanto a precisão quanto o recall, o que é um compromisso entre os dois indicadores. $F1$ Score pode ser a melhor medida para usar se precisarmos buscar um equilíbrio entre Precisão e Recall onde existe uma distribuição de classe desigual.

$$F1 = 2 \cdot \frac{\text{precisão} \cdot \text{recall}}{\text{precisão} + \text{recall}} \tag{3.15}$$

Ou seja, Precisão é uma medida da capacidade de um modelo de classificação para identificar apenas os pontos de dados relevantes, enquanto Recall é uma medida da capacidade de um modelo para encontrar todos os casos relevantes dentro de um conjunto de dados. As pontuações mais altas para Precisão e Recall mostram que o classificador está retornando resultados precisos, bem como retornando a maioria de todos os resultados positivos. Um

sistema ideal com alta Precisão e alto Recall retornará muitos resultados, com todos os resultados rotulados corretamente.

Estes três indicadores são indicadores de avaliação comumente usados na classificação do texto.

3.2 Modelos Deep Learning

Deep learning é um conjunto relativamente novo de métodos que está mudando o aprendizado de máquina de forma fundamental. Aprendizagem Profunda não é um algoritmo propriamente dito, mas uma família de algoritmos que implementam redes profundas com aprendizado com ou sem supervisão. Essas redes são tão profundas que novos métodos de cálculo, como GPUs, são necessários para construí-las (além de clusters de nós de cálculo).

Este trabalho usa métodos mais recentes de aprendizado de máquina para classificação de documentos em aprendizagem profunda.

Adiante iremos descreve o uso da abordagem de aprendizagem profunda para classificação de documentos. Esses métodos de aprendizagem profunda prometem fornecer maior precisão do que os métodos clássicos descritos anteriormente. Os métodos de aprendizagem profunda também fornecem arquiteturas flexíveis que usamos para produzir classificações. A classificação que nossos métodos produzem não é apenas altamente precisa, mas também permite uma maior compreensão da classificação resultante, mostrando onde o documento está dentro de um campo ou área de estudo.

O modelo *Long Short Term Memory networks* "LSTM"e o Modelo *Recurrent Convolutional Networks - RCN* descritos nas subseções 3.2.2 e 3.2.3 foram implementados considerando 2830 observações e 17 classes, usamos a função *"keras_model_sequential()"*, isso significa que a construção do modelo é de forma sequencial, ou seja cada camada é adicionada uma após a outra de forma sequencial.

Para os modelos de redes profundas os dados foram divididos em conjuntos de treinamento 60 % e os 40 % restantes são particionados para validação e testes dos dados.

A camada de incorporação foi inserida na primeira camada da arquitetura Deep Learning. Em vários tipos de estruturas de aprendizagem profunda, como Keras, as camadas de incorporação funcionam para o treinamento de dados de texto em vetores numéricos que representam a proximidade do significado de cada palavra. O processo de incorporação de palavras tem duas opções, isso pode ser feito durante o modelo de treinamento, ou também pode ser pré-treinado para produzir um modelo de incorporação de palavras (modelo word2vec/glove), depois usamos o modelo para incorporar a camada. Neste trabalho, nós usamos a segunda opção.

3.2.1 Word Embeddings

O *Word Embeddings* é uma técnica para identificar semelhanças entre palavras em um corpus usando algum tipo de modelo para prever a co-ocorrência de palavras dentro de um pequeno bloco de texto. Os *Word Embeddings* ganharam fama no mundo da análise automática de textos quando foi demonstrado que eles poderiam ser usados para identificar analogias.

Usamos a *Word Embeddings (Word2Vec/GloVe)* (Pennington et al. 2014) que é um método para mapear palavras de um vocabulário para vetores densos de números reais, em que palavras semanticamente semelhantes são mapeadas para pontos próximos. Os Word Embeddings fornecem uma representação densa de palavras e seus significados relativos.

Definimos a camada de incorporação como parte do modelo de rede neural. O modelo *skip-gram* é uma amostra do *word2vec*, uma classe de modelos preditivos computacionalmente eficientes para aprender a incorporação de palavras a partir de texto bruto.

A implementação deste modelo *skip-gram* criado por Mikolov et al. (2013) em *R* usando o pacote *Keras* foi realizado em nossos dados.

A maneira mais simples de associar um vetor denso a uma palavra seria escolher o vetor aleatoriamente. O problema com essa abordagem é que o espaço de incorporação resultante não teria estrutura: por exemplo, as palavras "preciso"e "exato"podem acabar com integrações completamente diferentes, embora sejam intercambiáveis na maioria das sentenças. Seria muito difícil para uma rede neural profunda dar sentido a um espaço de incorporação tão ruidoso e não estruturado.

Portanto, é razoável aprender um novo espaço de incorporação a cada nova tarefa. Felizmente, a retro-propagação torna isso fácil, e o *Keras* torna isso ainda mais fácil.

3.2.2 Rede Neural Recorrente

Em geral, as Redes Neurais Recorrentes são uma boa opção para avançar ou completar informações como preenchimento automático e modelar dados de sequência para previsões, mas sofrem de memória de curto prazo, ou seja se uma sequência for longa o suficiente, elas terão dificuldade em carregar informações de etapas de tempo anteriores para as posteriores.

Durante a propagação reversa, as redes neurais recorrentes sofrem do problema de gradiente de fuga. Os gradientes são valores usados para atualizar os pesos de redes neurais. O problema do gradiente de fuga é quando o gradiente diminui à medida que ele se propaga no tempo. Se um valor de gradiente se torna extremamente pequeno, isso não contribui muito para o aprendizado.

LSTM e GRU foram criados como a solução para a memória de curto prazo. Eles têm mecanismos internos chamados portas que podem regular o fluxo de informações. Essas portas podem aprender quais dados em uma sequência são importantes para manter ou jogar fora. Ao fazer isso, ele pode passar informações relevantes pela longa cadeia de sequências

para fazer previsões. LSTMs e GRUs podem ser encontrados em reconhecimento de fala, síntese de fala, geração de texto e também são usadas em problemas de Processamento de Linguagem Natural. Treinar LSTMs na entrada (usando aprendizagem supervisionada para modificar as células com base no erro da saída) com o algoritmo *backpropagation-through-time*, um tipo de retro-propagação aplicável a redes recorrentes.

O poder dos LSTMs se dá não em pequenas redes, mas em redes verticalmente profundas, que aumentam sua memória e suas redes, como consequência sua capacidade de representação. A complexidade desses algoritmos para o NLP pode ser significativa (Elman 1990).

Estudos envolvendo a utilização dessas ferramentas são recentes, tais como os de Mena-Chalco & Junior (2009), Mikolov et al. (2013), Pennington et al. (2014), Graves et al. (2013), Hochreiter & Schmidhuber (1997).

Em nosso modelo Figura 3.7 a camada de entrada é composta por palavras de *word2vec* concatenada. Adicionamos a camada dropout ao modelo LSTM, sendo está uma técnica de regularização para modelos de redes neurais proposta por (Srivastava et al. 2014) em que neurônios selecionados aleatoriamente são ignorados durante o treinamento. O *Dropout* é implementado com uma camada oculta, selecionando aleatoriamente os nós a serem descartados com uma determinada probabilidade de 20 % de cada ciclo de atualização de peso. O *Dropout* é implementado em Keras e usado apenas durante o treinamento do modelo e não é usado ao avaliar a habilidade do nosso modelo.

Adicionamos uma camada LSTM com 256 filtros de tamanho e finalmente um classificador *softmax*.

```
## Model: "sequential"
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## input (Embedding)            (None, 326, 32)           64768
## _________________________________________________________________
## embedding_dropout (Dropout)  (None, 326, 32)           0
## _________________________________________________________________
## lstm (LSTM)                  (None, 256)               295936
## _________________________________________________________________
## output (Dense)               (None, 17)                4369
## =================================================================
## Total params: 365,073
## Trainable params: 365,073
## Non-trainable params: 0
## _________________________________________________________________
```

Figura 3.7: Arquitetura do Modelo Recurrent Neural Networks para a classificação de sentença.

A vantagem da RNN é a capacidade para capturar melhor as informações contextuais. Isto

pode ser benéfico para capturar semântica de textos longos. No entanto, a RNN é um modelo tendencioso, onde as palavras posteriores são mais dominante do que palavras anteriores. Assim, poderia reduzir a eficácia quando é usado para capturar a semântica de todo um documento, porque os principais componentes podem aparecer em qualquer documento e não no final.

Para enfrentar o problema de viés, a Rede Neural Convolucional (CNN), um modelo imparcial é introduzido nas tarefas de PNL, determinando de maneira justa frases discriminativas em um texto com uma camada de pool máximo. Assim, a CNN pode capturar melhor a semântica de textos em comparação com redes neurais recorrentes.

Tensorboard é um aplicativo da web para visualizar informações sobre o aplicativo do Tensorflow. Os dados são escritos em Tensorflow e lidos por Tensorboard.

Podemos vizualizar o modelo RNN (*Long Short-Term Memory* - LSTM), gerado com o Tensorboard na Figura 3.8. Por padrão, o TensorBoard exibe o gráfico no nível operacional. À esquerda, podemos ver a tag "Padrão" selecionada. Observe que o gráfico está invertido; os dados fluem de baixo para cima, de modo que são invertidos em comparação com o código Figura 3.7. No entanto, você pode ver que o gráfico corresponde à definição do modelo Keras, com arestas extras para outros nós de computação.

Temos a operação *placeholder* de nome *input* onde o tensor que entra por esse node vai para o *layer* de *input* do modelo ele é utilizado para gerar os *outputs* do modelo.

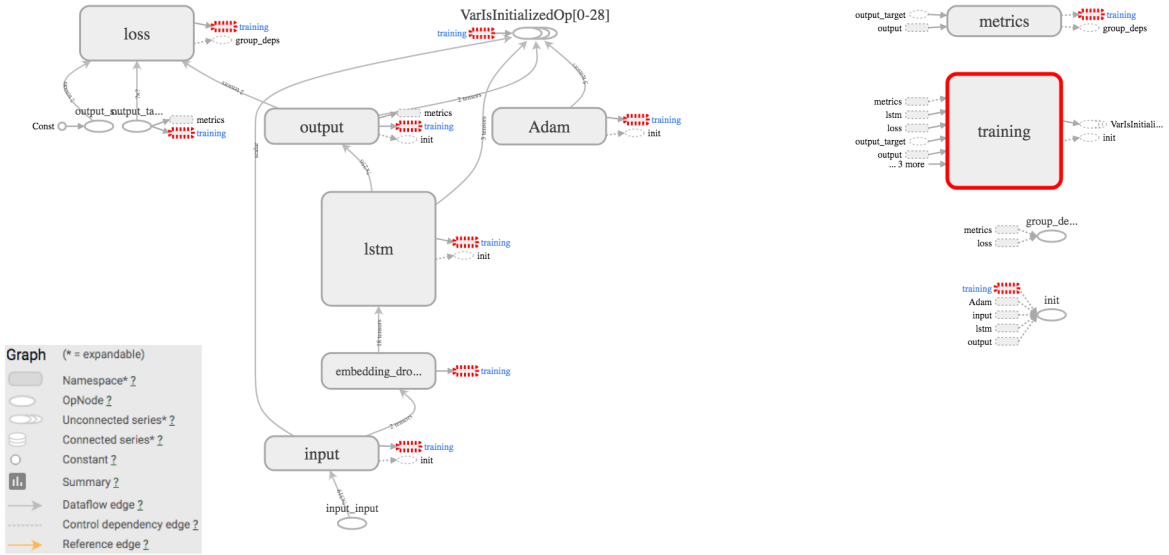


Figura 3.8: Gráfico computacional do Modelo Recurrent Neural Networks - LSTM.

3.2.3 Recurrent Convolutional Neural Network

Embora originalmente construído para processamento de imagem, as CNNs também foram efetivamente usadas para classificação de texto (Johnson & Zhang 2014). A camada

convolucional básica em uma CNN conecta-se a um pequeno subconjunto das entradas geralmente de tamanho 3×3. Similarmente, a próxima camada convolucional conecta-se apenas a um subconjunto de sua camada anterior. Dessa maneira, essas camadas de convolução, chamadas de mapas de recursos, podem ser empilhadas para fornecer vários filtros na entrada.

Para reduzir a complexidade computacional, as CNNs usam o *pooling* para reduzir o tamanho da saída de uma pilha de camadas. Diferentes técnicas de agrupamento são usadas para reduzir as saídas e preservar recursos importantes (Scherer et al. 2010). O método de agrupamento mais comum é o conjunto máximo, em que o elemento máximo é selecionado na janela de agrupamento.

Para alimentar a saída agrupada de mapas em destaque empilhados para a próxima camada, os mapas são achatados em uma coluna.

Em geral, durante a etapa de propagação reversa de uma CNN, não apenas os pesos são ajustados, mas também os filtros do detector de recursos. Um problema em potencial de CNNs usado para texto é o número de canais ou o tamanho do espaço de recurso. Isso pode ser muito grande (por exemplo, 50 mil palavras) para texto, mas para imagens isso é um problema menor (por exemplo, apenas três canais de RGB) (Johnson & Zhang 2014).

Construir uma arquitetura CNN significa que existem muitos hiper-parâmetros para escolher, alguns dos quais englobam: Representações de entrada (word2vec e GloVe), número e tamanhos de filtros de convolução, estratégias de agrupamento (máximo, média) e funções de ativação (ReLU, tanh). Zhang & Wallace (2015) realizaram avaliações empíricas do efeito de hiper-parâmetros variáveis nas arquiteturas da CNN, investigando seu impacto no desempenho e na variância em várias execuções.

Estudos anteriores sobre CNNs tendem a usar kernels convolucionais simples, como uma janela fixa (Collobert et al. 2011). Ao usar esses kernels, é difícil determinar o tamanho da janela: tamanhos pequenos de janelas podem resultar na perda de algumas informações críticas, enquanto janelas grandes resultam em um enorme espaço de parâmetros (que pode ser difícil de treinar). Para abordar a limitação do modelo acima, implementamos uma *Recurrent Convolutional Neural Network* (RCNN) Figura 3.9 e aplicamos à tarefa de classificação aos textos da IEEEGRSL.

Adicionamos uma camada de desistência de 10 % diretamente após a camada incorporação. Em seguida, adicionamos uma camada convolucional que passa um filtro sobre o texto para aprender blocos ou janelas específicas. Depois disso, temos uma camada MaxPooling, que combina todas as diferentes representações em partes em um único bloco.

Para as RCNN em vez de usamos um processo de vetorização (TF-IDF) convertemos cada palavra em um único número e aprendemos uma camada de Incorporação de palavras.

```
## Model: "sequential"
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## embedding (Embedding)        (None, 326, 128)          2560000
## _________________________________________________________________
## dropout (Dropout)            (None, 326, 128)          0
## _________________________________________________________________
## conv1d (Conv1D)              (None, 19, 64)            41024
## _________________________________________________________________
## max_pooling1d (MaxPooling1D) (None, 4, 64)             0
## _________________________________________________________________
## lstm (LSTM)                  (None, 70)                37800
## _________________________________________________________________
## dense (Dense)                (None, 17)                1207
## _________________________________________________________________
## activation (Activation)      (None, 17)                0
## =================================================================
## Total params: 2,640,031
## Trainable params: 2,640,031
## Non-trainable params: 0
## _________________________________________________________________
```

Figura 3.9: Arquitetura do Modelo Recurrent Convolutional Neural Networks Proposto.

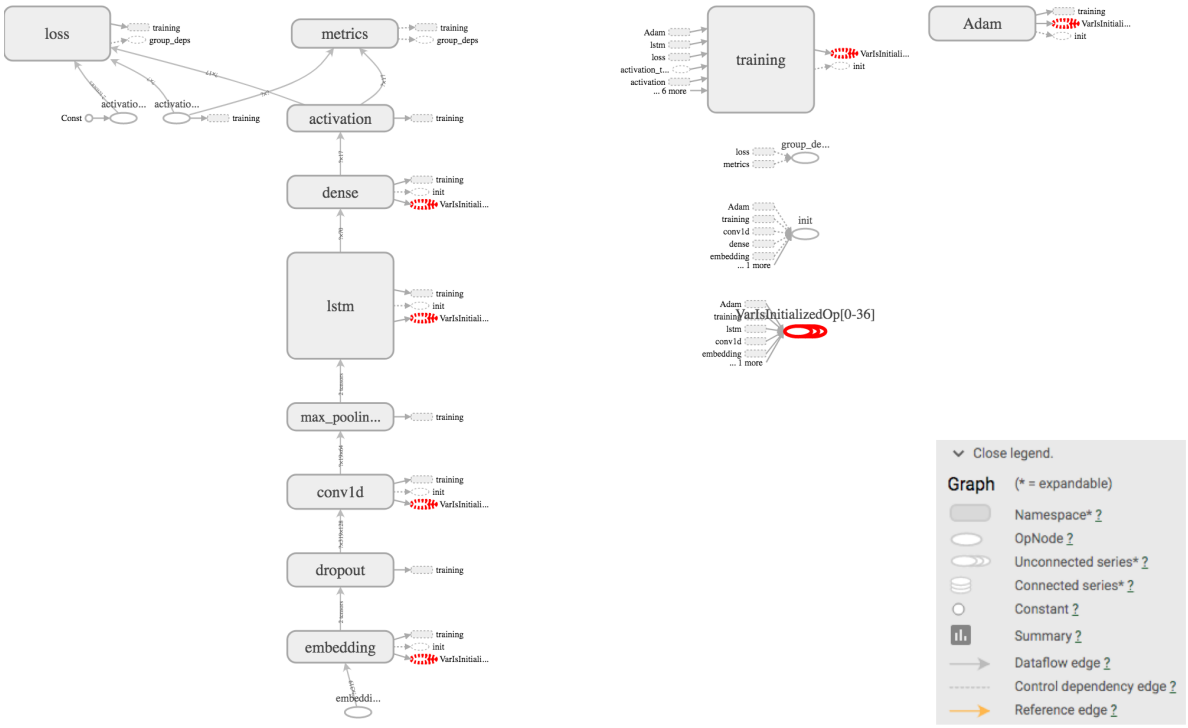


Figura 3.10: Gráfico computacional do Modelo Recurrent Convolutional Neural Networks - RCNN.

A camada de entrada é composta por palavras de *Glove* concatenada. Logo em seguida a rede é composta por uma camada convolucional de tamanho 19 com 64 filtros, uma função de ativação *Relu*, uma camada de *pool máximo* de tamanho 4 com 64 filtros, uma camada LSTM e finalmente um classificador *Sigmoid*. Uma arquitetura semelhante, mas um pouco mais complexa, foi previamente proposta em Kalchbrenner et al. (2014), Wang et al. (2015).

O modelo RNN com a implementação de uma camada CNN/Convnets é denominado como *Recurrent Convolutional Neural Networks* - CNN/RNN podemos observar na Figura 3.10 o modelo conceitual gerado no TensorBoard.

3.2.4 Local Interpretable Model-agnostic Explanations

As redes neurais costumavam ser vistas com suspeita por causa da sua natureza da "caixa preta", o que significa que esses modelos sofisticados são difíceis de explicar usando métodos tradicionais.

Nós introduzimos o *Local Interpretable Model-agnostic Explanations (LIME)* em Keras para Rede Neural (RCNN) API & LIME (Explicações Agnósticas do Modelo Interpretável Local) para abrir e explicar o modelo RCN.

O LIME é um algoritmo que fornece explicações baseadas em instâncias para previsões de qualquer classificador (Mishra et al. 2017, Ribeiro et al. 2016). Essas explicações são localmente fiéis à instância, independente do tipo de modelo do classificador, e são aprendidos sobre interpretável representações da instância. Por exemplo, para um sistema de classificação de e-mail, o LIME gera uma lista de palavras de um e-mail como uma explicação para sua classificação em alguma categoria. Para produzir a explicação, a LIME aborda o classificador localmente com um modelo interpretável (por exemplo, modelos lineares esparsos, árvores de decisão).

O presente capítulo detalhou o processo de *Classificação de documentos* usando redes neurais profunda e outras abordagens. A seguir, no quarto capítulo, apresentaremos os resultados e análises com base nos modelos utilizado.

4

RESULTADOS E ANÁLISES

As seções deste capítulo mostram, respectivamente, os resultados experimentais da execução de diferentes algoritmos de classificação que se diferenciam pelos algoritmos-base e pela arquitetura de classificação. Posteriormente mostraremos com maior profundidade os resultados obtidos na execução das redes RNN e RCNN.

Após a aplicação das funções para a classificação, foi criada uma matriz de confusão para estudar e obter os resultados da análise. Uma matriz de confusão, ou tabela de contingência, é uma representação da matriz da precisão de classificação de um algoritmo formado pela elaboração dos resultados reais ou esperados em relação aos resultados classificados ou previstos. Ao olhar para essa matriz e examiná-la, é possível discernir se um documento ou afirmação foi ou não classificado corretamente, e se foi classificado de alguma forma. A partir dessa matriz, calculamos valores como Verdadeiros Positivos, Falsos Positivos, Verdadeiros Negativos e Falsos Negativos; seguido por medidas de desempenho como *Precision, Recall, F-measure* e *Accuracy*. Esses parâmetros foram discutidos em detalhes na subseção 3.1.13.

Este capítulo está fundamentada na quinta fase da metodologia (CRISP-DM) Figura 3.1.

O desempenho dos classificadores para os modelos baseado na variável *Título* é apresentado na Tabela 4.1 e para variável *Abstract* na Tabela 4.2.

Podemos notar na Tabela 4.1 que o modelo com maior fração de documentos recuperados que são relevantes para a consulta é o *Boosting* com 98 %, porém quando se leva em consideração as outras medidas como a fração dos documentos relevantes que são recuperados com sucesso, assim como a precisão e a recuperação, o modelo *Agregação por Bootstrap* se destaca com 96 %.

Na Tabela 4.2 é possível observar que o *Boosting* supera todos os outros modelos com maior fração de documentos recuperados que são relevantes para a consulta assim como a fração dos documentos relevantes que são recuperados com sucesso, a precisão e a recuperação.

Algoritmo	PRECISION	RECALL	F-SCORE	ACC
SVM	0.90	0.92	0.91	0.87
SLDA	0.89	0.87	0.87	0.95
LOGITBOOST	**0.98**	0.95	**0.96**	0.98
BAGGING	0.96	**0.96**	**0.96**	0.97
FORESTS	0.93	0.88	0.89	0.96
TREE	0.80	0.83	0.81	0.94
NNETWORK	0.16	0.18	0.15	0.39
MAXENTROPY	0.97	0.91	0.93	**0.99**
NB	0.38	0.39	0.37	0.52

Tabela 4.1: Desempenho dos Algoritmos, Precisão, Recall, F-scores e Accuracy para variável Título.

Algoritmo	PRECISION	RECALL	F-SCORE	ACC
SVM	0.55	0.54	0.53	0.19
LOGITBOOST	**0.91**	**0.86**	**0.87**	**0.94**
FORESTS	0.76	0.63	0.64	0.84
TREE	0.67	0.64	0.64	0.81
MAXENTROPY	0.66	0.58	0.59	0.92
NB	0.33	0.34	0.32	0.50

Tabela 4.2: Desempenho dos Algoritmos, Precisão, Recall, F-scores e Accuracy para variável Abstract.

A partir dos resultados experimentais, fica claro que o modelo com melhor desempenho para os dois conjuntos de dados é o Boosting, porém é possível notar uma queda no seu desempenho quando o modelo recebe um volume de dados maior, isto implica dizer que, quanto maior o volume de dados o desempenho deste modelo tende a cair, não sendo uma boa opção para a era do Big Data onde o volume de dados tende ao crescimento.

Os algoritmos aplicados no método de aprendizado de máquina foram avaliados usando ferramentas poderosas disponíveis no pacote *RTextTools* de *R*. A função *create_analytics()* tornou todas as medidas de desempenho relevantes para os dados de teste classificados por um conjunto de algoritmos - (*BAGGING, BOOSTING, GLMNET, MAXENT, NNET, RF, SLDA, SVM* e *TREE*). A função retorna um contêiner com quatro resumos diferentes: por rótulo, por algoritmo, por documento e um resumo de *Ensemble*. Nesse caso, como todos os dados nos

conjuntos de treinamento e teste têm rótulos correspondentes, *create_analytics()* verificará os resultados dos algoritmos de aprendizado em relação aos valores reais para determinar a precisão do processo.

O resumo do rótulo fornece estatísticas sobre cada uma das categorias de requisitos nos dados classificados. Inclui o número de documentos classificados em cada etiqueta pelo método *Ensemble*, bem como suas probabilidades.

O resumo do algoritmo fornece valores de desempenho: Precisão e Recall, fornecidos por cada um dos algoritmos usados no *Ensemble*.

O resumo do documento fornece estatísticas sobre cada documento classificado e inclui a previsão de cada algoritmo, a pontuação de probabilidade do algoritmo, o número de algoritmos que concordaram com o mesmo rótulo, qual algoritmo teve a maior pontuação de probabilidade para sua previsão e o rótulo original dessa declaração. Por fim, o *Ensemble Summary* fornece detalhes sobre o classificador do conjunto como um todo descrito nas Tabelas 4.3 e 4.4 incluindo também a *n-Ensemble Agreement*.

Para este modelo, $n = 8$, ou seja 8 algoritmos foram empregados para os modelos envolvendo a variável Título e 5 algoritmos foram empregados para os modelos envolvendo a variável Resumo.

Coverage refere-se simplesmente à porcentagem de documentos que atendem ao limite de precisão de recall. O *Ensemble Agreement* simplesmente se refere a vários algoritmos fazendo a mesma previsão em relação à classe de um evento (ou seja, o Boosting e a Máxima Entropia rotularam o mesmo texto?).

Matematicamente, se k representa o percentual de casos que atendem ao limite do *Ensemble* e n representa o total de casos, a *Coverage* é calculada da seguinte maneira:

$$\text{Coverage} = \frac{k}{n} \tag{4.1}$$

Usando uma abordagem de concordância de 4 conjuntos, Collingwood & Wilkerson descobriram que quando 4 de seus algoritmos concordam com o rótulo de um documento textual, o rótulo da máquina é corresponde ao rótulo humano em 90 %. A taxa é de apenas 45 % quando apenas 2 algoritmos concordam com o rótulo do texto.

A tabela 4.3 e 4.4 reporta a *Coverage* e *Recall Accuracy* para diferentes níveis de *Ensemble agreement*. A tendência geral é que a *Coverage* diminua enquanto o Recall aumente.

Podemos observar na Tabela 4.3 que 96 % tem 6 algoritmos dos 8 (SVM, SLDA, Boosting, Bagging, RF, TREE, NNET, MaxEnt) que concordam. No entanto, a precisão de recall é de 100 % quando os 6 algoritmos concordam. Considerando que 90 % é frequentemente um padrão de confiabilidade inter-codificadores, podemos está usando um acordo de 6 *Ensemble Agreement* com esses dados.

Na tabela 4.4 onde 60 % tem 5 algoritmos dos 5 (SVM, Boosting, RF, TREE, MaxEnt) que concordam. No entanto, a precisão de recall é de 97 % quando os 5 algoritmos concordam.

	COVERAGE	RECALL
n >= 1	1.00	0.98
n >= 2	1.00	0.98
n >= 3	1.00	0.98
n >= 4	1.00	0.98
n >= 5	0.98	0.99
n >= 6	0.96	1.00

Tabela 4.3: Ensemble Agreement Coverage e Recall para variável Título.

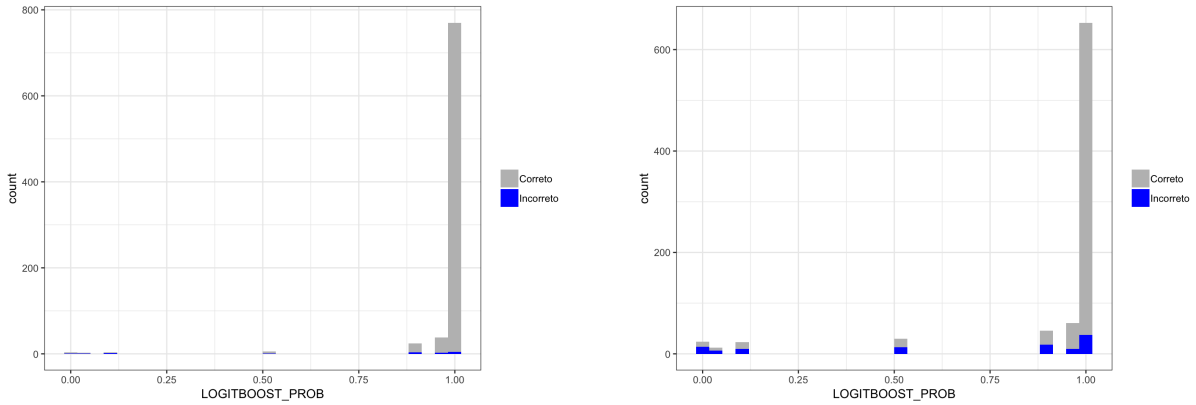
	COVERAGE	RECALL
n >= 1	1.00	0.87
n >= 2	1.00	0.88
n >= 3	0.95	0.90
n >= 4	0.79	0.96
n >= 5	0.60	0.97

Tabela 4.4: Ensemble Agreement Coverage e Recall para variável Abstract.

Considerando que 90 % é frequentemente um padrão de confiabilidade inter-codificadores, podemos está usando um acordo de 3 *Ensemble Agreement* com esses dados no qual obtemos um *Coverage* de 95 %.

Como última analítica, podemos traçar a distribuição das probabilidades do modelo *Boosting* se eles estavam corretos versus se estavam incorretos Figura 4.1 e a distribuição das probabilidades dos erros e acertos dividido por cada classe Figura 4.2.

Na Distribuição das Probabilidades de *Boosting* conforme Figura 4.1 nota-se que a Variável Resumo obteve maior pontos de incorreções.

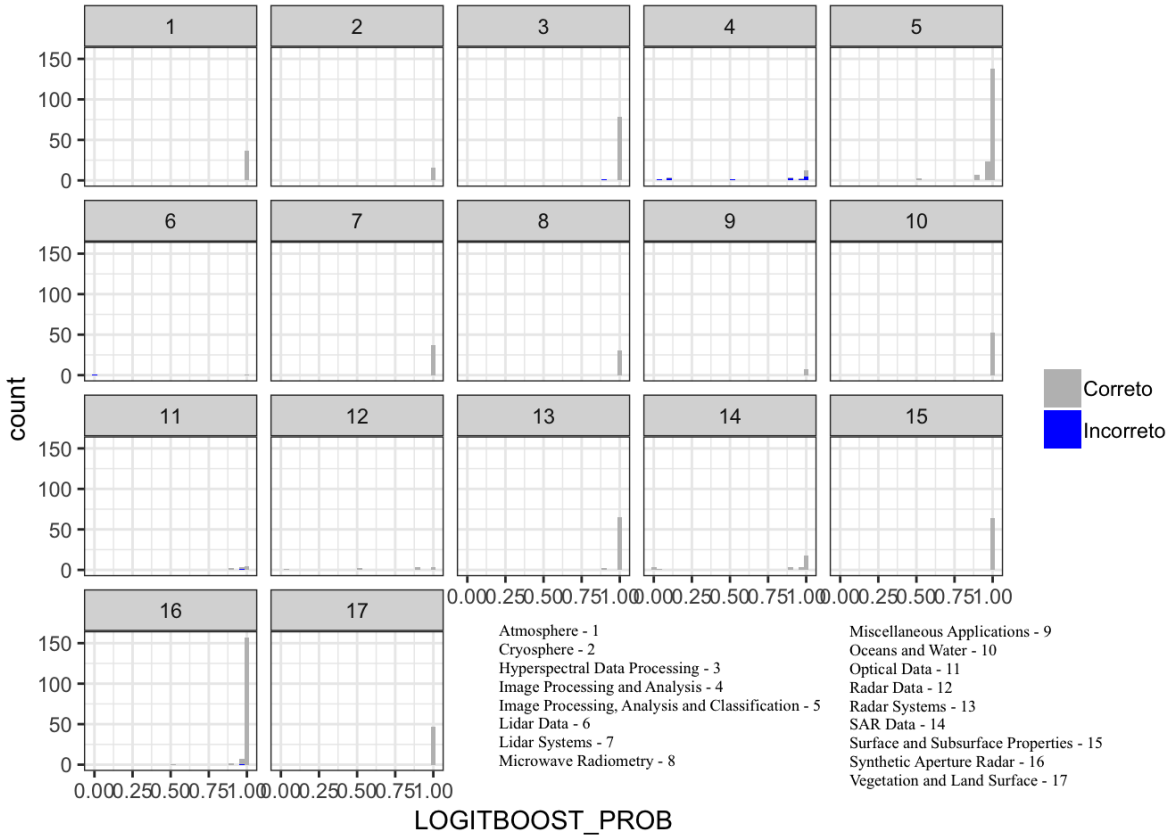


(a) Variável Título.

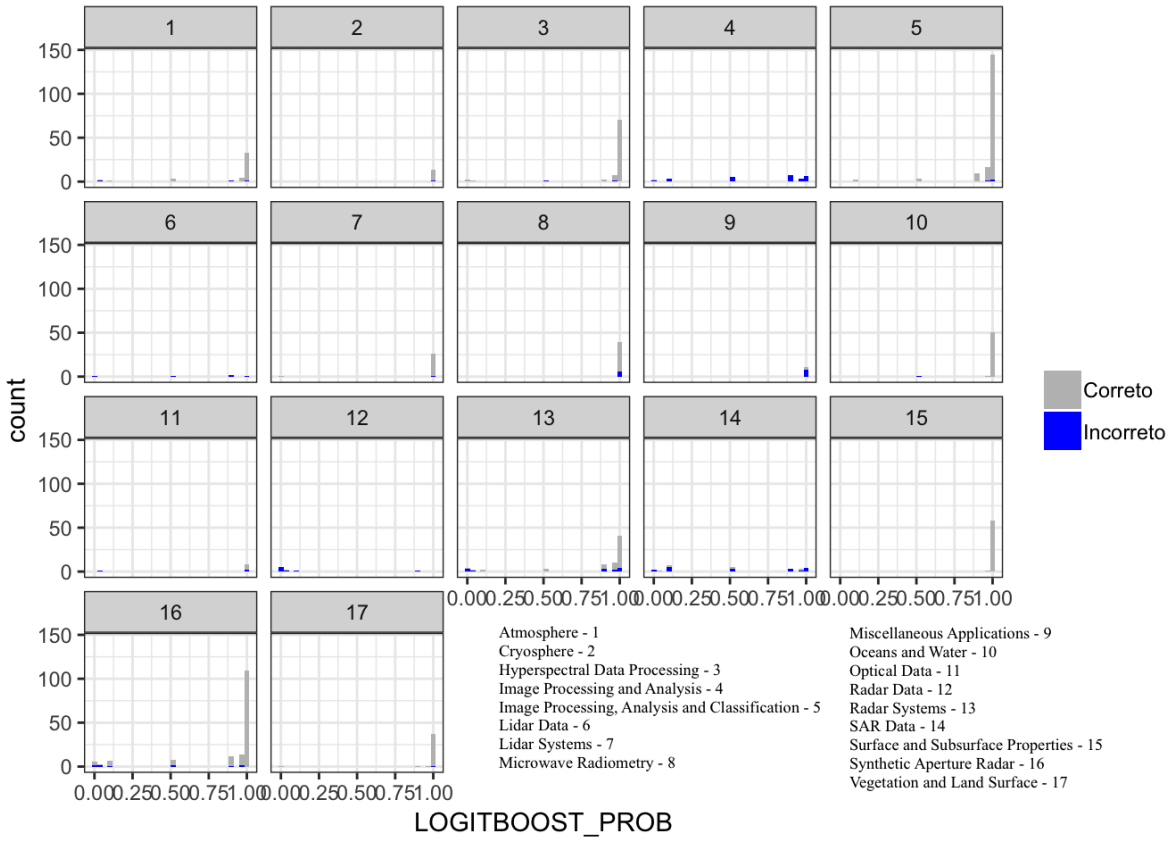
(b) Variável Resumo.

Figura 4.1: Distribuição das Probabilidades de Boosting contra se eles estavam corretos versus incorretos.

Na Distribuição das Probabilidades de *Boosting* conforme Figura 4.2 nota-se que a Variável *Abstract* teve maior pontos de incorreções por classe. Também é possível observar que a classe com maior índice de classificação incorreta é a 4 *Image Processing and Analysis* tanto para variável Título quanto para Resumo. Para o Resumo podemos notar que outras classes também se destacam com um número visível de observações incorretas como a 13, 14 e 16.



(a) Variável Título.



(b) Variável Resumo.

Figura 4.2: Distribuição das Probabilidades de Boosting corretos versus incorretos por Classe.

Podemos observar o desempenho da RNN (Long Short-Term Memory - LSTM) na Figura 4.3 e o desempenho com a implementação de uma camada CNN/Convnets denominado como Recurrent Convolutional Neural Networks - CNN/RNN observado na Figura 4.6 e 4.7.

Ao adicionar uma camada CNN antes do LSTM, permitimos que o LSTM veja sequências de blocos em vez de sequências de palavras.

A rede RNN obteve uma Accuracy de 55 % em 30 *epoch* de treinamento levando em média 44 minutos para o treinamento. Podemos observar como o custo e a acurácia (de treino) evoluem ao longo do treinamento. Vemos que, foi preciso muitas interações de treino, para chegarmos em uma região de custo baixo conforme Figura 4.3.

Nos diagramas da Figura 4.4 podemos ver a evolução das distribuições dos pesos, vieses e ativações de cada camada para a RNN onde a *bias* é a camada que usa um vetor de polarização, o *kernel* é a matriz de pesos usado para a transformação linear das entradas e a *recurrent_kernel* matriz de pesos, usada para a transformação linear do estado recorrente.

Nos histogramas Figura 4.5 onde mostramos a distribuição, resumo dos 3 pesos, a parte sombreada mostra aqueles pesos que são realmente ativados durante o treinamento, essas curvas representam percentis como o máximo, a média e mediana, o gráfico do histograma nos permite traçar qualquer variáveis.

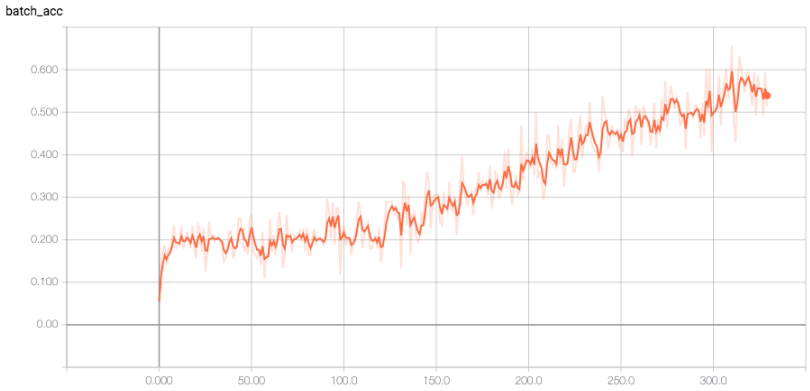
Para a RCNN treinamos os dados com 2 *epoch* de treinamento levamos em média 6 segundos para o treinamento.

Na Figura 4.6 podemos ver como o custo e a acurácia (de treino) evoluem ao longo do treinamento. Vemos que, com bem poucas iterações de treino, já chegamos em uma região de custo bem baixo e com uma acurácia satisfatória de 94 %.

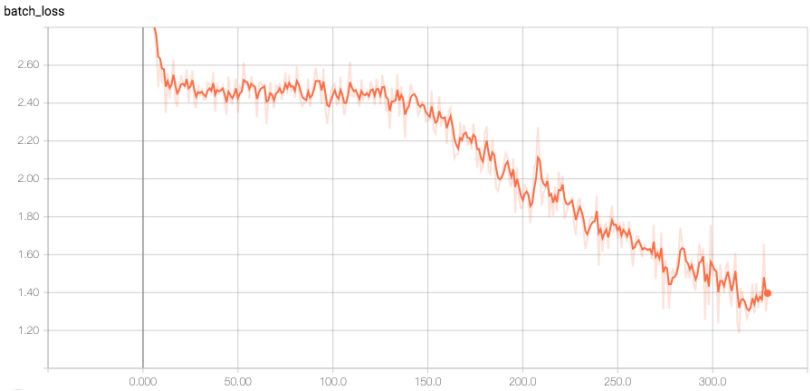
Conseguimos está mesma acurácia de 94 % com o algoritmo *Boosting* para o conjunto de dados *Abstract*, porém com um tempo de treinamento em média de 1.200 vezes maior que a RCNN.

Nos diagramas de distribuição Figura 4.8 e 4.9 podemos ver a evolução das distribuições dos pesos, vieses e ativações de cada camada para a rede RCNN. Os histogramas está mostrando como os valores de nossos pesos mudam com o treinamento e como os dados mudam ao longo do tempo. No histograma temos 3 eixos; tempo (eixo x), valor (eixo y) e frequência / densidade de valores (eixo z), onde a parte mais escura representa dado mais antigos e as mais claras representam dados mais recentes. Um valor mais alto no eixo z significa que o vetor contém mais valores próximos desse valor específico.

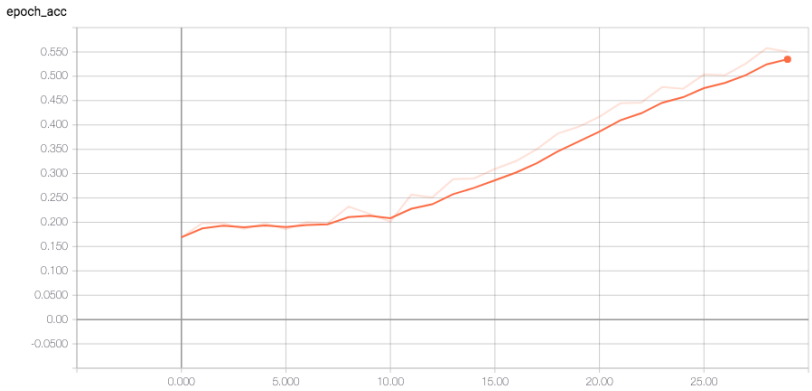
Observa-se na Figura 4.8 que a camada *Lstm/bias* tem um pico em 0 no início do treinamento já a camada *CNN/bias* Figura 4.9 a ativação da primeira camada tem um pico no final do treinamento, o que indica que muitos neurônios estão “mortos”, isto é, na região da CNN em que o valor é zero e não há gradiente. Isso indica que talvez pudéssemos manter a nossa performance diminuindo o custo computacional em termos de neurônios, mas para isso precisaríamos de uma função de ativação que tivesse algum gradiente na parte negativa do seu domínio.



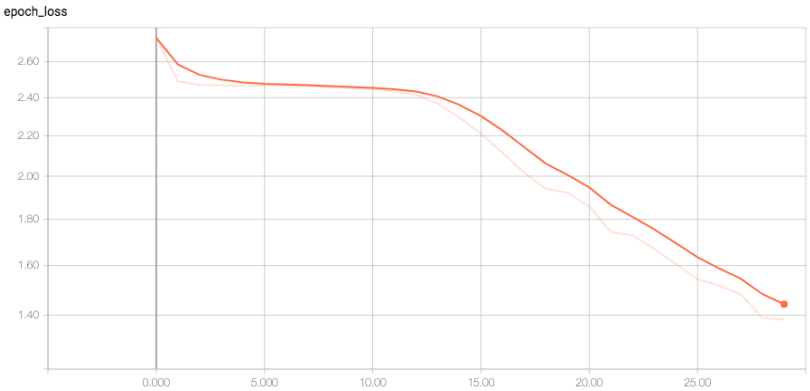
(a) Batch Acc.



(b) Batch Loss.

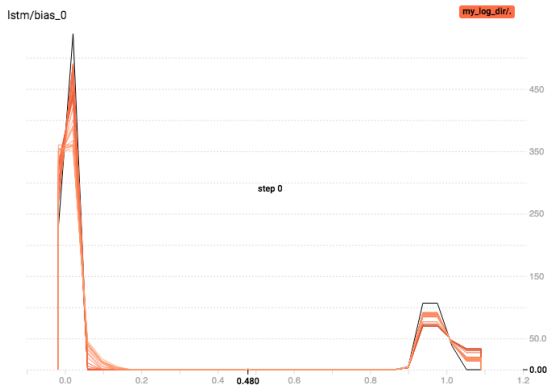


(c) Epoch Acc.

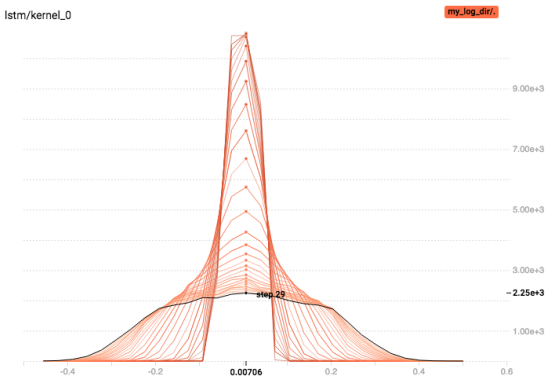


(d) Epoch Loss.

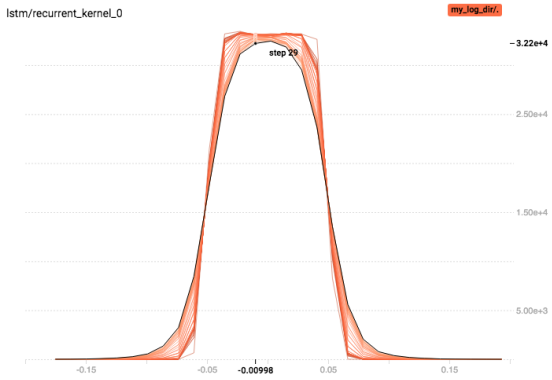
Figura 4.3: Desempenho do Modelo RNN-LSTM para variável Abstract.



(a) Lstm Bias.



(b) Lstm Kernel.



(c) Lstm Recurrent Kernel.

Figura 4.4: Distribuição dos pesos, vies e ativação de cada camada RNN.

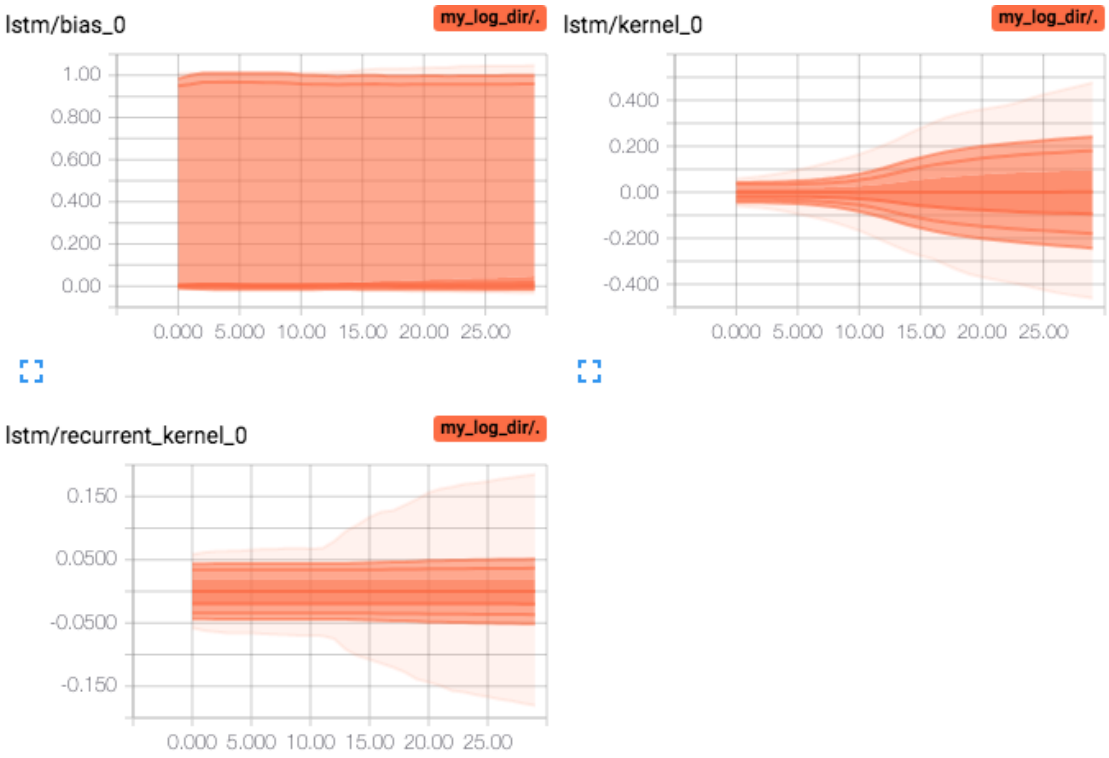
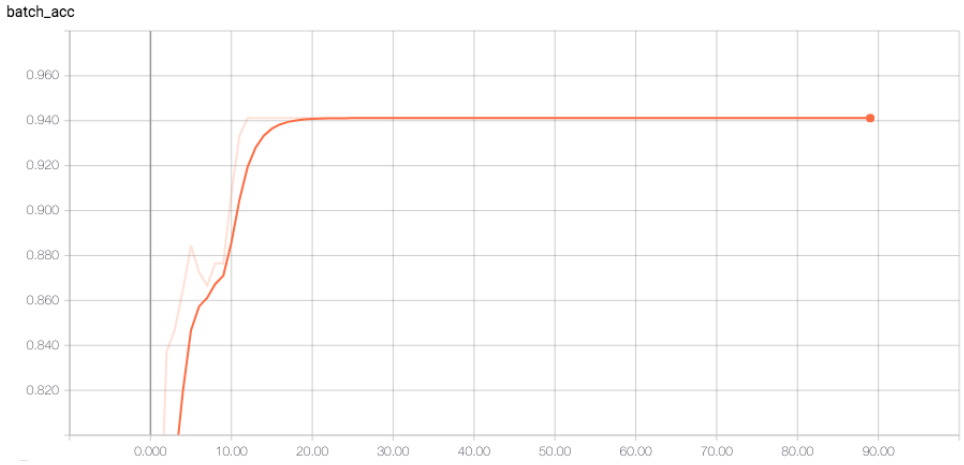
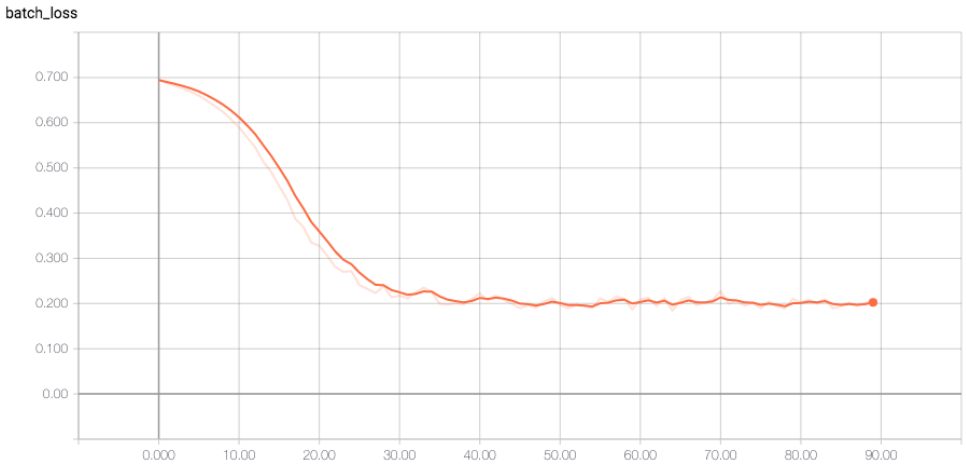


Figura 4.5: Desempenho do Modelo RNN-LSTM para variável Abstract.



(a) Batch Acc.



(b) Batch Loss.

Figura 4.6: Desempenho do Modelo RCNN para variável Abstract.

É possível notar também que a camada LSTM em comparação com a camada CNN tem menos interações de treino, ou seja poucas ativações da RNN em relação a CNN.

Podemos observar a evolução das distribuições dos pesos, vieses e ativações de cada camada da incorporação de palavras também conhecida como *Embedding* Figura 4.10.

Abaixo Figura 4.11 temos uma demostração de como o modelo pode ser visualizado no LIME, testado apenas para uma categoria classe (*Cryosphere*) onde podemos notar, em azul as palavras que enfatizam a classificação e em vermelho as que não o fazem, e suas probabilidades e porcentagem de ocorrência. A previsão é então explicada por um "explainer"que destaca as palavras mais importantes para o modelo.

Podemos observar na Figura 4.11b que a probabilidade do texto 4 (case 4) pertencer a classe *Cryosphere* é de 68 % e seu explanation de 78 % para classificação n=2, ou seja a partir de apenas 2 palavras relevantes presentes no texto. Na Figura 4.11c podemos observar uma entrada de texto no *Siny* para classificação a partir de 2 palavras relevantes (*number of word to select*), com precisão de 99 % de chances de não pertencer a esta categoria (*Cryosphere*).

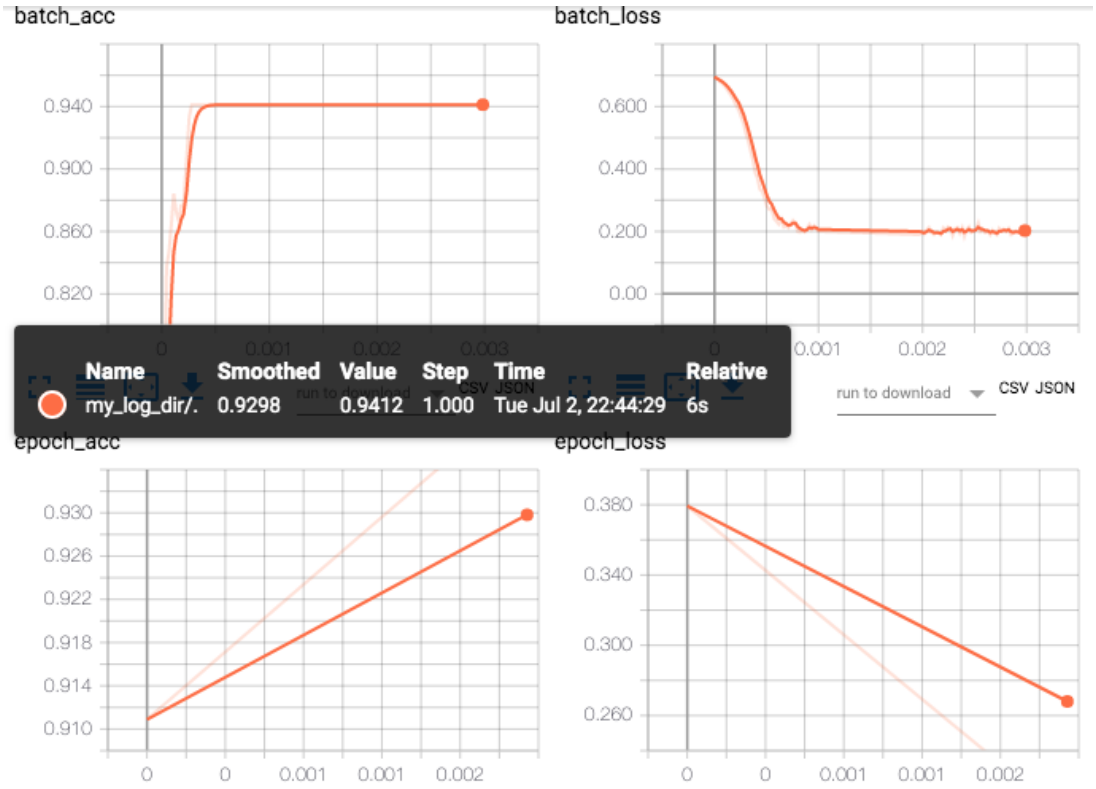


Figura 4.7: Desempenho do Modelo RCNN para variável Abstract.

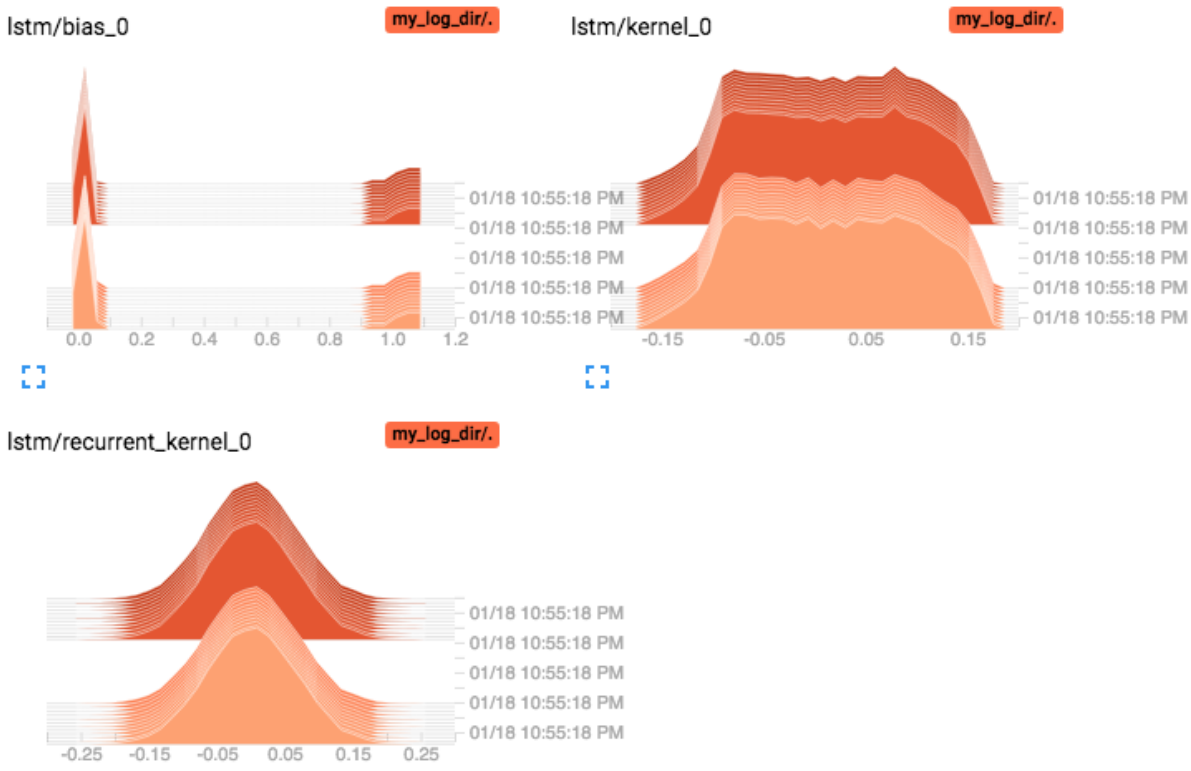


Figura 4.8: Histograma da camada LSTM Modelo RCNN para variável Abstract.

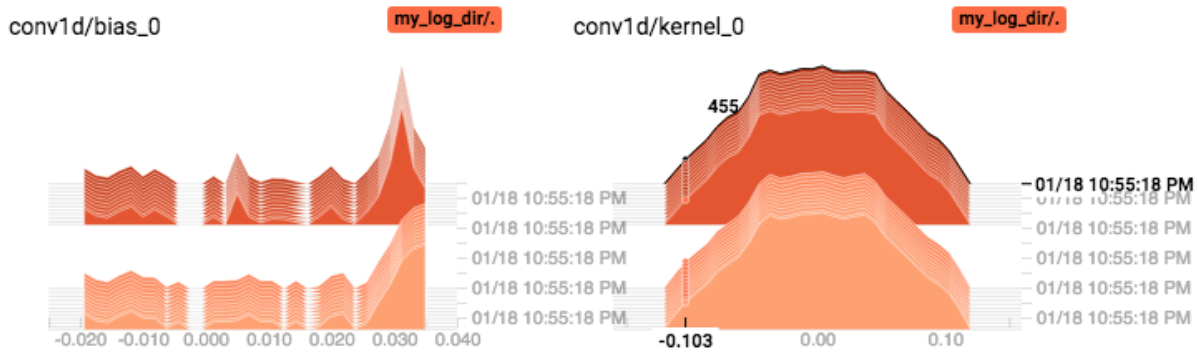
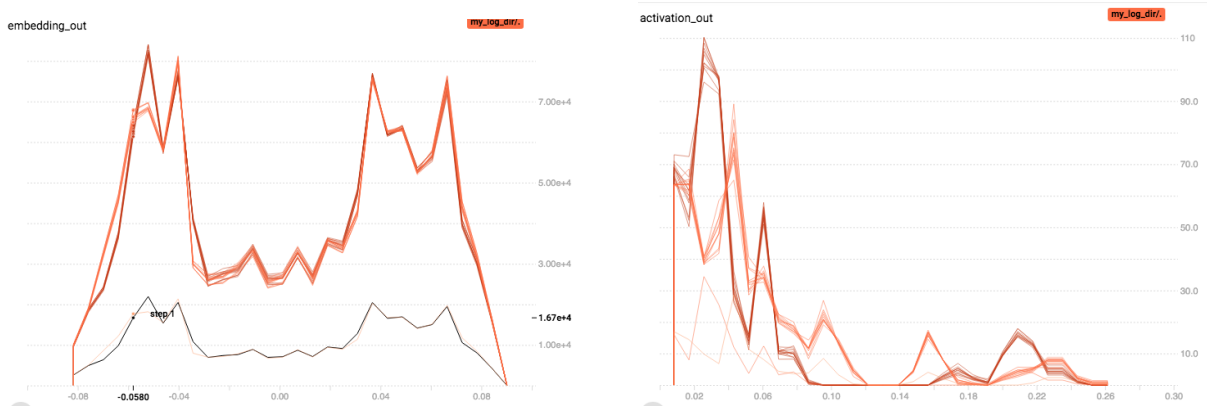


Figura 4.9: Histograma da camada CNN Modelo RCNN para variável Abstract.



(a) Embedding. (b) Activation.

Figura 4.10: Histograma de distribuição da camada Embedding para variável Abstract.

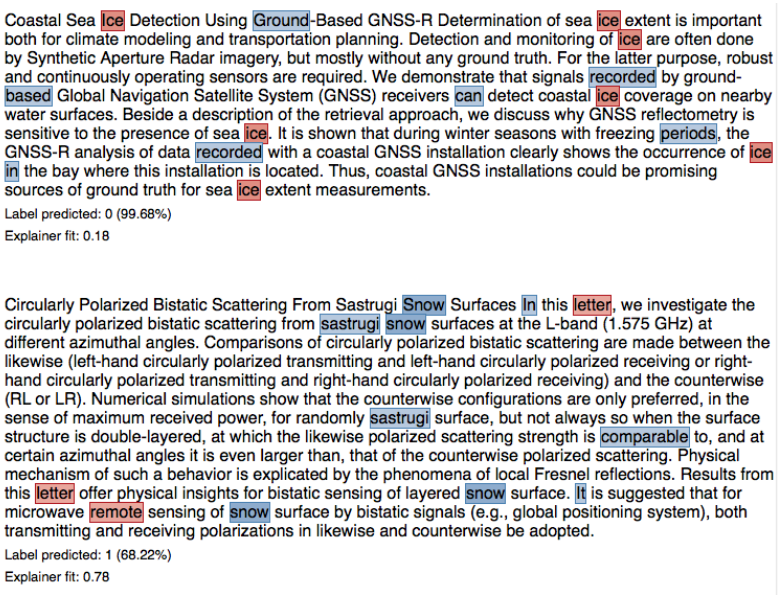
A partir dos dados apresentados, pode-se observar que os modelos (SVM, SLDA, Bagging, RF, TREE e Maxent) conseguiram bons resultados para a classificação dos textos a partir do título dos artigos, porém quando classificamos os textos a partir dos resumos, nota-se uma queda bastante acentuada em todos os modelos.

O modelo *Boosting* conseguiu apresentar um grau de aderência bastante robusto para a variável resumo, no entanto tende a perder sua precisão quando o volume de dados aumenta, assim como sua performance de tempo de treinamento.

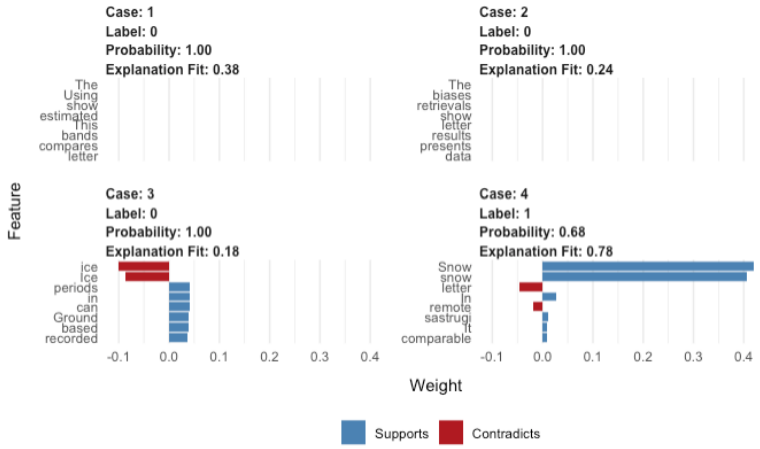
O modelo Recurrent Convolutional Networks - RCNN alcançou uma *Precisão* de 94 % com 2 epoch de treinamento em 6s para os dados do resumo, superando assim os modelos clássicos implementado neste trabalho mostrado na tabela 4.2 em termos de performance de treinamento.

A RNN se mostrou insatisfatória para o conjunto de dados *Abstract* em termos de precisão e se mostrou inferior a RCNN em termos de performance.

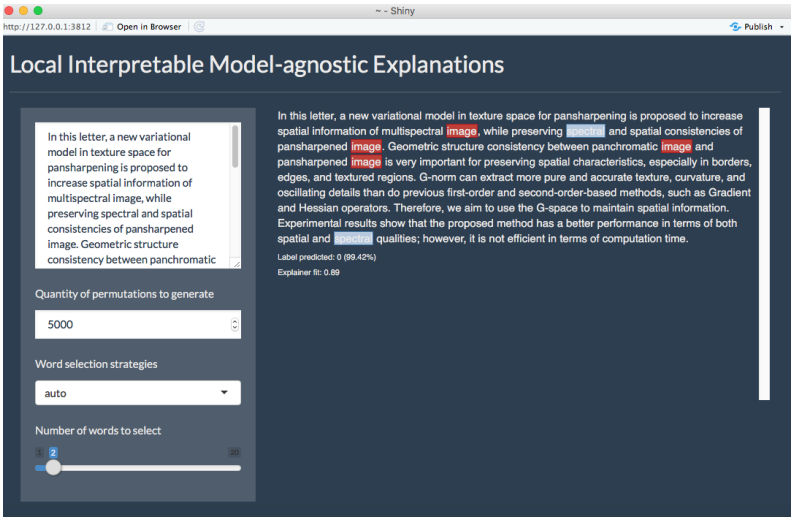
Até o momento não foi desenvolvido um sistema capaz de fazer a interface entre o usuário e a construção do índice de forma automática.



(a) Palavras Relevantes do Texto.



(b) Probabilidade de Ocorrência.



(c) Siny

Figura 4.11: Palavras relevantes para categorização (Cryosphere) usando o LIME.

Esse capítulo apresentou as vantagens e desvantagens no uso de métodos de referências em classificação e também de novos métodos utilizando redes neurais profundas. Finalizamos com o próximo capítulo, no qual serão relatados os principais impactos e contribuições deste trabalho e os passos futuros.

5

CONCLUSÕES

ESTE capítulo abordará o avanço que o presente trabalho traz para o cenário científico e porquê isso é interessante. Além disso, apresentará sugestões para trabalhos futuros.

5.1 Considerações Finais

O trabalho apresentado realizou estudo e aplicação das técnicas de mineração de textos com o objetivo de criar o índice da revista IEEE Geoscience and Remote Sensing Letters, utilizando alguns modelos de classificação automática de documentos. A prova de conceito aqui efetuada demonstrou a viabilidade de aplicação desta solução, evidenciando que o índice pode ser construído de forma semi-automatizada sem perda significante de qualidade.

Primeiro foi realizados levantamento dos dados e entendimento.

Após esta fase, foram executadas as etapas de pré-processamento e limpeza dos dados, com o objetivo de transformar os textos em uma representação capaz de ser utilizada pelos algoritmos e redes de aprendizagem de máquina.

Foram utilizados algoritmos conhecidos na literatura para classificação de textos (Maximum Entropy Modeling, Support Vector Machine, Bootstrap Aggregating, Boosting, Redes neurais da NNET, Random Forest, Análise discriminante linear escalada (SLDA), Decision Trees e Naïve Bayes). Entretanto boa parte deles alcançou resultados satisfatórios, porém com uma tendência a reduzir sua capacidade de precisão quando comparado a grande volume de dados.

Optou-se por utilizar as redes neurais profundas em especial as RNN e RCNN.

Introduzimos redes neurais convolucionais recorrentes para classificação do texto. Nosso modelo captura informações contextuais com a estrutura recorrente e constrói a representação do texto usando uma rede neural convolucional. O experimento demonstra que nosso modelo supera os modelos clássicos e a RNN usando dois conjuntos de dados de classificação de texto diferentes em termos de performace.

No entanto, ao implementar essa técnica de classificação multi-label utilizando deep learning, os índices de acerto para o conjunto de dados Abstract superou os Modelos clássicos implementado neste trabalho, chegando a uma *precisão* de 94 % com uma performace de 6 segundos.

Ponderando os resultados expostos, podemos concluir que o classificador automático é uma solução plausível para o problema de construção do índice da revista.

Desse modo, com a utilização dos métodos aqui descrito é possível reduzir o tempo gasto com trabalhos manuais. Este fator dará aceleração ao processo, resultando em benefícios para IEEE Geoscience and Remote Sensing Society, a sociedade responsável pela edição do periódico IEEE Geoscience and Remote Sensing Letters e a comunidade acadêmica.

5.2 Trabalhos futuros

Como trabalhos futuros pretende-se:

- Aumentar o conjunto de dados e avaliar a performace da RCNN frente aos métodos clássicos.
- Avaliar a RNN e RCNN por outras métricas de avaliação (Precision, Recall e F-Score).
- Usar o LIME para explicar os modelos.
- Avaliar o desempenho dos Modelos.
- Implementar a CapsNet ou Capsules Net e avaliar o desempenho frente a RCNN.

REFERÊNCIAS BIBLIOGRÁFICAS

Aggarwal, C. C. & Zhai, C. (2012), *Mining text data*, Springer Science & Business Media.

Almiron, M. G., Lopes, B., Oliveira, A. L., Medeiros, A. C., Frery, A. C. et al. (2010), 'On the numerical accuracy of spreadsheets', *Journal of Statistical Software* **34**(4), 1–29.

Aly, M. (2005), 'Survey on multiclass classification methods', *Neural Netw* **19**, 1–9.

Amari, S.-i., Cichocki, A. & Yang, H. H. (1996), A new learning algorithm for blind signal separation, *in* 'Advances in neural information processing systems', pp. 757–763.

Anderson, J. R. (1983), 'A spreading activation theory of memory', *Journal of verbal learning and verbal behavior* **22**(3), 261–295.

Arampatzis, A., Van Der Weide, T. P., van Bommel, P. & Koster, C. H. (2000), 'Linguistically motivated information retrieval'.

Arras, L., Horn, F., Montavon, G., Müller, K.-R. & Samek, W. (2017), 'What is relevant in a text document?: An interpretable machine learning approach', *PloS one* **12**(8), e0181142.

Azevedo, A. I. R. L. & Santos, M. F. (2008), 'Kdd, semma and crisp-dm: a parallel overview', *IADS-DM* .

Bahdanau, D., Cho, K. & Bengio, Y. (2014), 'Neural machine translation by jointly learning to align and translate', *arXiv preprint arXiv:1409.0473* .

Bao, Y. & Ishii, N. (2002), Combining multiple k-nearest neighbor classifiers for text classification by reducts, *in* 'International Conference on Discovery Science', Springer, pp. 340–347.

Bayes, T. (1763), 'Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s', *Philosophical transactions of the Royal Society of London* (53), 370–418.

Bekkerman, R., El-Yaniv, R., Tishby, N. & Winter, Y. (2003), 'Distributional word clusters vs. words for text categorization', *Journal of Machine Learning Research* **3**(Mar), 1183–1208.

Bengio, Y., Ducharme, R., Vincent, P. & Jauvin, C. (2003), 'A neural probabilistic language model', *Journal of machine learning research* **3**(Feb), 1137–1155.

Berger, A. L., Pietra, V. J. D. & Pietra, S. A. D. (1996), 'A maximum entropy approach to natural language processing', *Computational linguistics* **22**(1), 39–71.

Bi, Y., Bell, D., Wang, H., Guo, G. & Greer, K. (2004), Combining multiple classifiers using dempster's rule of combination for text categorization, *in* 'International Conference on Modeling Decisions for Artificial Intelligence', Springer, pp. 127–138.

Bick, E. (1998), Structural lexical heuristics in the automatic analysis of portuguese, *in* 'Proceedings of the 11th Nordic Conference of Computational Linguistics (Nodalida 1998)', pp. 44–56.

Bod, R. (1995), *Enriching linguistics with statistics: Performance models of natural language*, Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Breiman, L. (1996), 'Bagging predictors', *Machine learning* **24**(2), 123–140.

Breiman, L. (1997), Arcing the edge, Technical report, Technical Report 486, Statistics Department, University of California at ....

Breiman, L. (2001), 'Random forests', *Machine learning* **45**(1), 5–32.

Breiman, L. (2017), *Classification and regression trees*, Routledge.

Cerri, R., de Carvalho, A. C. P. & Freitas, A. A. (2011), 'Adapting non-hierarchical multilabel classification methods for hierarchical multilabel classification', *Intelligent Data Analysis* **15**(6), 861–887.

Cho, S.-B. & Lee, J.-H. (2003), Learning neural network ensemble for practical text classification, *in* 'International Conference on Intelligent Data Engineering and Automated Learning', Springer, pp. 1032–1036.

Collingwood, L., Jurka, T., Boydstun, A. E., Grossman, E., van Atteveldt, W. et al. (2013), 'Rtexttools: A supervised learning package for text classification'.

Collingwood, L. & Wilkerson, J. (2012), 'Tradeoffs in accuracy and efficiency in supervised learning methods', *Journal of Information Technology & Politics* **9**(3), 298–318.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. & Kuksa, P. (2011), 'Natural language processing (almost) from scratch', *Journal of Machine Learning Research* **12**(Aug), 2493–2537.

Cortes, C. & Vapnik, V. (1995), 'Support-vector networks', *Machine learning* **20**(3), 273–297.

De Lucca, J. & Nunes, M. d. G. V. (2002), 'Lematização versus stemming', *USP, UFSCar, UNESP, São Carlos, São Paulo* .

Dictionary, O. (1986), 'Thesaurus.(1997)', *New York City, NY: Harper Collins* .

Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D. & Weingessel, A. (2008), 'Misc functions of the department of statistics (e1071), tu wien', *R package* **1**, 5–24.

Ding, L. & Salem, M. B. (2018), A novel architecture for automatic document classification for effective security in edge computing environments, *in* '2018 IEEE/ACM Symposium on Edge Computing (SEC)', IEEE, pp. 416–420.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. & Darrell, T. (2015), Long-term recurrent convolutional networks for visual recognition and description, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2625–2634.

Drucker, H., Wu, D. & Vapnik, V. N. (1999), 'Support vector machines for spam categorization', *IEEE Transactions on Neural networks* **10**(5), 1048–1054.

Duda, R. O., Hart, P. E. & Stork, D. G. (2012), *Pattern classification*, John Wiley & Sons.

Dumais, S., Platt, J., Heckerman, D. & Sahami, M. (1998), Inductive learning algorithms and representations for text categorization, *in* 'Proceedings of the seventh international conference on Information and knowledge management', ACM, pp. 148–155.

Elman, J. L. (1990), 'Finding structure in time', *Cognitive science* **14**(2), 179–211.

Eskin, E., Weston, J., Noble, W. S. & Leslie, C. S. (2003), Mismatch string kernels for svm protein classification, *in* 'Advances in neural information processing systems', pp. 1441–1448.

Feinerer, I. (2013), 'Introduction to the tm package text mining in r', *Accessible en ligne: http://cran. r-project. org/web/packages/tm/vignettes/tm. pdf* .

Feng, G., Guo, J., Jing, B.-Y. & Hao, L. (2012), 'A bayesian feature selection paradigm for text classification', *Information Processing & Management* **48**(2), 283–302.

Frank, E., Chui, C. & Witten, I. (2000), Text categorization using compression models, *in* 'Data Compression Conference, 2000. Proceedings. DCC 2000', IEEE, p. 555.

Frants, V. J., Shapiro, J. & Voiskunskii, V. G. (1997), *Automated information retrieval: theory and methods*, Emerald Group Publishing Limited.

French, J. C., Brown, D. E. & Kim, N.-H. (1997), 'A classification approach to boolean query reformulation', *Journal of the American Society for Information Science* **48**(8), 694–706.

Freund, Y. & Schapire, R. E. (1997), 'A decision-theoretic generalization of on-line learning and an application to boosting', *Journal of computer and system sciences* **55**(1), 119–139.

Friedl, M. A. & Brodley, C. E. (1997), 'Decision tree classification of land cover from remotely sensed data', *Remote sensing of environment* **61**(3), 399–409.

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Mikolov, T. et al. (2013), Devise: A deep visual-semantic embedding model, *in* 'Advances in neural information processing systems', pp. 2121–2129.

Fukushima, K. (1988), 'Neocognitron: A hierarchical neural network capable of visual pattern recognition.', *Neural networks* **1**(2), 119–130.

Gonzalez, M. & Lima, V. L. S. (2003), Recuperação de informação e processamento da linguagem natural, *in* 'XXIII Congresso da Sociedade Brasileira de Computaçao', Vol. 3, pp. 347–395.

Gopal, S. & Yang, Y. (2010), Multilabel classification with meta-level features, *in* 'Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 315–322.

Graves, A. (2013), 'Generating sequences with recurrent neural networks', *arXiv preprint arXiv:1308.0850* .

Graves, A. & Jaitly, N. (2014), Towards end-to-end speech recognition with recurrent neural networks, *in* 'International Conference on Machine Learning', pp. 1764–1772.

Graves, A., Mohamed, A.-r. & Hinton, G. (2013), Speech recognition with deep recurrent neural networks, *in* '2013 IEEE international conference on acoustics, speech and signal processing', IEEE, pp. 6645–6649.

Graves, A., Wayne, G. & Danihelka, I. (2014), 'Neural turing machines', *arXiv preprint arXiv:1410.5401* .

Grossberg, S. (1988), 'Nonlinear neural networks: Principles, mechanisms, and architectures', *Neural networks* **1**(1), 17–61.

Günal, S., Ergin, S., Gülmezoğlu, M. B. & Gerek, Ö. N. (2006), On feature extraction for spam e-mail detection, *in* 'International Workshop on Multimedia Content Representation, Classification and Security', Springer, pp. 635–642.

Hayes, P. J., Andersen, P. M., Nirenburg, I. B. & Schmandt, L. M. (1990), Tcs: a shell for content-based text categorization, *in* 'Artificial Intelligence Applications, 1990., Sixth Conference on', IEEE, pp. 320–326.

Hayes, P. J. & Weinstein, S. P. (1991), Adding value to financial news by computer, *in* 'Proceedings First International Conference on Artificial Intelligence Applications on Wall Street', IEEE, pp. 2–8.

Haykin, S. (1994), *Neural networks: a comprehensive foundation*, Prentice Hall PTR.

Hebb, D. O. et al. (1949), 'The organization of behavior: A neuropsychological theory'.

Hinton, G. E. (1986), Learning distributed representations of concepts, *in* 'Proceedings of the eighth annual conference of the cognitive science society', Vol. 1, Amherst, MA, p. 12.

Hinton, G. E. & Salakhutdinov, R. R. (2006), 'Reducing the dimensionality of data with neural networks', *science* **313**(5786), 504–507.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hopfield, J. J. (1982), 'Neural networks and physical systems with emergent collective computational abilities', *Proceedings of the national academy of sciences* **79**(8), 2554–2558.

Hotho, A., Nürnberger, A. & Paaß, G. (2005), A brief survey of text mining., *in* 'Ldv Forum', Citeseer, pp. 19–62.

Huang, G. B., Lee, H. & Learned-Miller, E. (2012), Learning hierarchical representations for face verification with convolutional deep belief networks, *in* 'Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on', IEEE, pp. 2518–2525.

Hull, D. A. (1996), 'Stemming algorithms: A case study for detailed evaluation', *Journal of the American Society for Information Science* **47**(1), 70–84.

Iyyer, M., Manjunatha, V., Boyd-Graber, J. & Daumé III, H. (2015), Deep unordered composition rivals syntactic methods for text classification, *in* 'Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)', Vol. 1, pp. 1681–1691.

Jackson, P. & Moulinier, I. (2007), *Natural language processing for online applications: Text retrieval, extraction and categorization*, Vol. 5, John Benjamins Publishing.

Jacobs, P. S. (1992), Joining statistics with nlp for text categorization, *in* 'Proceedings of the third conference on Applied natural language processing', Association for Computational Linguistics, pp. 178–185.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013), *An introduction to statistical learning*, Vol. 112, Springer.

Joachims, T. (1996), A probabilistic analysis of the rocchio algorithm with tfidf for text categorization., Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science.

Joachims, T. (1998), Text categorization with support vector machines: Learning with many relevant features, *in* 'European conference on machine learning', Springer, pp. 137–142.

Joachims, T. (1999), Transductive inference for text classification using support vector machines, *in* 'ICML', Vol. 99, pp. 200–209.

Joachims, T. (2002), *Learning to classify text using support vector machines: Methods, theory and algorithms*, Vol. 186, Kluwer Academic Publishers Norwell.

Johnson, D. E., Oles, F. J., Zhang, T. & Goetz, T. (2002), 'A decision-tree-based symbolic rule induction system for text categorization', *IBM Systems Journal* **41**(3), 428–437.

Johnson, R. & Zhang, T. (2014), 'Effective use of word order for text categorization with convolutional neural networks', *arXiv preprint arXiv:1412.1058* .

Jones, K. S. (1997), *Readings in information retrieval*, Morgan Kaufmann.

Jurafsky, D. & Martin, J. H. (2014), *Speech and language processing*, Vol. 3, Pearson London.

Jurka, T. P. (2012), 'Maxent: an r package for low-memory multinomial logistic regression with support for semi-automated text classification', *The R Journal* **4**(1), 56–59.

Kalchbrenner, N., Grefenstette, E. & Blunsom, P. (2014), 'A convolutional neural network for modelling sentences', *arXiv preprint arXiv:1404.2188* .

Kalt, T. & Croft, W. (1996), A new probabilistic model of text classification and retrieval, Technical report, Technical Report IR-78, University of Massachusetts Center for Intelligent ....

Kim, S.-B., Han, K.-S., Rim, H.-C. & Myaeng, S. H. (2006), 'Some effective techniques for naive bayes text classification', *IEEE transactions on knowledge and data engineering* **18**(11), 1457–1466.

Kim, Y. (2014), 'Convolutional neural networks for sentence classification', *arXiv preprint arXiv:1408.5882* .

Koller, D. & Sahami, M. (1997), Hierarchically classifying documents using very few words, Technical report, Stanford InfoLab.

Korde, V. & Mahender, C. N. (2012), 'Text classification and classifiers: A survey', *International Journal of Artificial Intelligence & Applications* **3**(2), 85.

Kotsiantis, S. B., Zaharakis, I. & Pintelas, P. (2007), 'Supervised machine learning: A review of classification techniques', *Emerging artificial intelligence applications in computer engineering* **160**, 3–24.

Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S. & Barnes, L. E. (2017), Hdltex: Hierarchical deep learning for text classification, *in* 'Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on', IEEE, pp. 364–371.

Krenn, B. & Samuelson, C. (1994), 'The linguist's guide to statistics, 1997', *A compendiurn for a course in Statistical Approaches in Computational Linguisticsheld at tlie University of Saarland* **1995**.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

Kudo, T. & Matsumoto, Y. (2004), A boosting algorithm for classification of semi-structured text, *in* 'Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing', pp. 301–308.

Kuhn, M. & Johnson, K. (2013), *Applied predictive modeling*, Vol. 26, Springer.

Larkey, L. S. & Croft, W. B. (1996), Combining classifiers in text categorization, *in* 'Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 289–297.

Le, Q. & Mikolov, T. (2014), Distributed representations of sentences and documents, *in* 'International Conference on Machine Learning', pp. 1188–1196.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989), 'Backpropagation applied to handwritten zip code recognition', *Neural computation* **1**(4), 541–551.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

Lee, J. Y. & Dernoncourt, F. (2016), 'Sequential short-text classification with recurrent and convolutional neural networks', *arXiv preprint arXiv:1603.03827* .

Leopold, E. & Kindermann, J. (2002), 'Text categorization with support vector machines. how to represent texts in input space?', *Machine Learning* **46**(1-3), 423–444.

Leslie, C., Eskin, E. & Noble, W. S. (2001), The spectrum kernel: A string kernel for svm protein classification, *in* 'Biocomputing 2002', World Scientific, pp. 564–575.

Lewis, D. D. (1991), Evaluating text categorization i, *in* 'Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991'.

Lewis, D. D. (1992), Feature selection and feature extraction for text categorization, *in* 'Proceedings of the workshop on Speech and Natural Language', Association for Computational Linguistics, pp. 212–217.

Lewis, D. D. (1998), Naive (bayes) at forty: The independence assumption in information retrieval, *in* 'European conference on machine learning', Springer, pp. 4–15.

Lewis, D. D. & Ringuette, M. (1994), A comparison of two learning algorithms for text categorization, *in* 'Third annual symposium on document analysis and information retrieval', Vol. 33, pp. 81–93.

Liaw, A., Wiener, M. et al. (2002), 'Classification and regression by randomforest', *R news* **2**(3), 18–22.

Lim, H. S. (2004), Improving knn based text classification with well estimated parameters, *in* 'International Conference on Neural Information Processing', Springer, pp. 516–523.

Lovins, J. B. (1968), 'Development of a stemming algorithm', *Mech. Translat. & Comp. Linguistics* **11**(1-2), 22–31.

Lu, Y. & Tanne, M. (2017), 'Systems for and methods of finding relevant documents by analyzing tags'. US Patent 9,715,542.

Luhn, H. P. (1958), 'The automatic creation of literature abstracts', *IBM Journal of research and development* **2**(2), 159–165.

Luong, T., Socher, R. & Manning, C. (2013), Better word representations with recursive neural networks for morphology, *in* 'Proceedings of the Seventeenth Conference on Computational Natural Language Learning', pp. 104–113.

Madsen, R. E., Sigurdsson, S., Hansen, L. K. & Larsen, J. (2004), Pruning the vocabulary for better context recognition, *in* 'Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on', Vol. 2, IEEE, pp. 483–488.

Masand, B., Linoff, G. & Waltz, D. (1992), Classifying news stories using memory based reasoning, *in* 'Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 59–65.

Mcauliffe, J. D. & Blei, D. M. (2008), Supervised topic models, *in* 'Advances in neural information processing systems', pp. 121–128.

McCallum, A. K. (1996), 'Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering', *http://www.cs.cmu.edu/mccallum/bow/* .

McCallum, A. K. (2002), 'Mallet: A machine learning for language toolkit'.

McCallum, A., Nigam, K. et al. (1998), A comparison of event models for naive bayes text classification, *in* 'AAAI-98 workshop on learning for text categorization', Citeseer, pp. 41–48.

McCulloch, W. S. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *The bulletin of mathematical biophysics* **5**(4), 115–133.

Medsker, L. & Jain, L. C. (1999), *Recurrent neural networks: design and applications*, CRC press.

Mena-Chalco, J. P. & Junior, R. M. C. (2009), 'scriptLattes: an open-source knowledge extraction system from the Lattes platform', *Journal of the Brazilian Computer Society* **15**(4), 31–39.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013), 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781* .

Miller, W. T., Werbos, P. J. & Sutton, R. S. (1995), *Neural networks for control*, MIT press.

Mishra, S., Sturm, B. L. & Dixon, S. (2017), Local interpretable model-agnostic explanations for music content analysis, *in* 'Proc. ISMIR'.

Mitchell, T. M. et al. (1997), 'Machine learning. wcb'.

Mnih, V., Heess, N., Graves, A. et al. (2014), Recurrent models of visual attention, *in* 'Advances in neural information processing systems', pp. 2204–2212.

Moens, M.-F. (2001), 'Automatic indexing and abstracting of document texts'.

Moens, M.-F. (2006), *Automatic indexing and abstracting of document texts*, Vol. 6, Springer Science & Business Media.

Nardiello, P., Sebastiani, F. & Sperduti, A. (2003), Discretizing continuous attributes in adaboost for text categorization, *in* 'European Conference on Information Retrieval', Springer, pp. 320–334.

Nigam, K., Lafferty, J. & McCallum, A. (1999), Using maximum entropy for text classification, *in* 'IJCAI-99 workshop on machine learning for information filtering', Vol. 1, pp. 61–67.

Nigam, K., McCallum, A., Thrun, S. & Mitchell, T. (1998), Using em to classify text from labeled and unlabeled documents, Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G. S. & Dean, J. (2013), 'Zero-shot learning by convex combination of semantic embeddings', *arXiv preprint arXiv:1312.5650* .

Oquab, M., Bottou, L., Laptev, I. & Sivic, J. (2014), Learning and transferring mid-level image representations using convolutional neural networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1717–1724.

Orengo, V. M., Buriol, L. S. & Coelho, A. R. (2006), A study on the use of stemming for monolingual ad-hoc portuguese information retrieval, *in* 'Workshop of the Cross-Language Evaluation Forum for European Languages', Springer, pp. 91–98.

Osuna, E., Freund, R. & Girosit, F. (1997), Training support vector machines: an application to face detection, *in* 'Computer vision and pattern recognition, 1997. Proceedings., 1997 IEEE computer society conference on', IEEE, pp. 130–136.

Palacio, M., Ferrero, S. & Frery, A. (2018), 'Revisiting the effect of spatial resolution on information content based on classification results', *arXiv preprint arXiv:1807.10333* .

Pang, B., Lee, L. & Vaithyanathan, S. (2002), Thumbs up?: sentiment classification using machine learning techniques, *in* 'Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10', Association for Computational Linguistics, pp. 79–86.

Pennington, J., Socher, R. & Manning, C. (2014), Glove: Global vectors for word representation, *in* 'Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)', pp. 1532–1543.

Pereira, F., Mitchell, T. & Botvinick, M. (2009), 'Machine learning classifiers and fmri: a tutorial overview', *Neuroimage* **45**(1), S199–S209.

Peters, A., Hothorn, T. & Lausen, B. (2002), 'ipred: Improved predictors', *R News* **2**(2), 33–36. **URL:** *http://CRAN.R-project.org/doc/Rnews/*

Poole, D. L., Mackworth, A. K. & Goebel, R. (1998), *Computational intelligence: a logical approach*, Vol. 1, Oxford University Press New York.

Porter, M. F. (1980), 'An algorithm for suffix stripping', *Program* **14**(3), 130–137.

Quinlan, J. R. (1986), 'Induction of decision trees', *Machine learning* **1**(1), 81–106.

Reilly, D. L. & Cooper, L. N. (1995), An overview of neural networks: early models to real world systems, *in* 'How We Learn; How We Remember: Toward An Understanding Of Brain And Neural Systems: Selected Papers of Leon N Cooper', World Scientific, pp. 300–321.

Ren, M., Kiros, R. & Zemel, R. (2015), Exploring models and data for image question answering, *in* 'Advances in neural information processing systems', pp. 2953–2961.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016), Why should i trust you?: Explaining the predictions of any classifier, *in* 'Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining', ACM, pp. 1135–1144.

Ripley, B. & Venables, W. (2011), 'nnet: Feed-forward neural networks and multinomial log-linear models', *R package version* **7**(5).

Rosenblatt, F. (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.', *Psychological review* **65**(6), 386.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1985), Learning internal representations by error propagation, Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Russell, S. J. & Norvig, P. (2016), *Artificial intelligence: a modern approach*, Malaysia; Pearson Education Limited,.

Sahami, M., Dumais, S., Heckerman, D. & Horvitz, E. (1998), A bayesian approach to filtering junk e-mail, *in* 'Learning for Text Categorization: Papers from the 1998 workshop', Vol. 62, Madison, Wisconsin, pp. 98–105.

Saif, H., Fernández, M., He, Y. & Alani, H. (2014), 'On stopwords, filtering and data sparsity for sentiment analysis of twitter'.

Salakhutdinov, R., Tenenbaum, J. B. & Torralba, A. (2013), 'Learning with hierarchical-deep models', *IEEE transactions on pattern analysis and machine intelligence* **35**(8), 1958–1971.

Salton, G., Wong, A. & Yang, C.-S. (1975), 'A vector space model for automatic indexing', *Communications of the ACM* **18**(11), 613–620.

Scapini, I. (1995), 'Relações entre itens lexicais', *Poersch, JM; Wertheimer, AMC; Ouro, MEP; Ludwig, EM* pp. 393–429.

Scavuzzo, J. M., Trucco, F., Espinosa, M., Tauro, C. B., Abril, M., Scavuzzo, C. M. & Frery, A. C. (2018), 'Modeling dengue vector population using remotely sensed data and machine learning', *Acta tropica* **185**, 167–175.

Schapire, R. E., Freund, Y., Bartlett, P., Lee, W. S. et al. (1998), 'Boosting the margin: A new explanation for the effectiveness of voting methods', *The annals of statistics* **26**(5), 1651–1686.

Schapire, R. E. & Singer, Y. (2000), 'Boostexter: A boosting-based system for text categorization', *Machine learning* **39**(2-3), 135–168.

Scherer, D., Müller, A. & Behnke, S. (2010), Evaluation of pooling operations in convolutional architectures for object recognition, *in* 'Artificial Neural Networks–ICANN 2010', Springer, pp. 92–101.

Schneider, K.-M. (2005), Techniques for improving the performance of naive bayes for text classification, *in* 'International Conference on Intelligent Text Processing and Computational Linguistics', Springer, pp. 682–693.

Sebastiani, F. (2002), 'Machine learning in automated text categorization', *ACM computing surveys (CSUR)* **34**(1), 1–47.

Shannon, C. E. & McCarthy, J. (2016), *Automata Studies.(AM-34)*, Vol. 34, Princeton University Press.

Shen, Y., He, X., Gao, J., Deng, L. & Mesnil, G. (2014), Learning semantic representations using convolutional neural networks for web search, *in* 'Proceedings of the 23rd International Conference on World Wide Web', ACM, pp. 373–374.

Silva, C. & Ribeiro, B. (2003), The importance of stop word removal on recall values in text categorization, *in* 'Neural Networks, 2003. Proceedings of the International Joint Conference on', Vol. 3, IEEE, pp. 1661–1666.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. & Potts, C. (2013), Recursive deep models for semantic compositionality over a sentiment treebank, *in* 'Proceedings of the 2013 conference on empirical methods in natural language processing', pp. 1631–1642.

Soukhanov, A. H., Ellis, K. & Severynse, M. (1992), *The American heritage dictionary of the English language*, Houghton Mifflin.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: A simple way to prevent neural networks from overfitting', *Journal of Machine Learning Research* **15**, 1929–1958.
URL: *http://jmlr.org/papers/v15/srivastava14a.html*

Stamatatos, E., Fakotakis, N. & Kokkinakis, G. (2000), 'Automatic text categorization in terms of genre and author', *Computational linguistics* **26**(4), 471–495.

Sukhbaatar, S., Weston, J., Fergus, R. et al. (2015), End-to-end memory networks, *in* 'Advances in neural information processing systems', pp. 2440–2448.

Sutskever, I., Martens, J. & Hinton, G. E. (2011), Generating text with recurrent neural networks, *in* 'Proceedings of the 28th International Conference on Machine Learning (ICML-11)', pp. 1017–1024.

Sutskever, I., Vinyals, O. & Le, Q. V. (2014), Sequence to sequence learning with neural networks, *in* 'Advances in neural information processing systems', pp. 3104–3112.

Tan, S., Wang, Y. & Wu, G. (2011), 'Adapting centroid classifier for document categorization', *Expert Systems with Applications* **38**(8), 10264–10273.

Turian, J., Ratinov, L. & Bengio, Y. (2010), Word representations: a simple and general method for semi-supervised learning, *in* 'Proceedings of the 48th annual meeting of the association for computational linguistics', Association for Computational Linguistics, pp. 384–394.

Tuszynski, J. (2012), 'catools: Tools: moving window statistics, gif, base64, roc auc, etc., r package version 1.17. 1', *URL http://CRAN. R-project. org/package= caTools.[accessed 01 April 2014]* .

Tuszynski, J. & Orphaned, M. (2013), 'Package 'catools''.

Uysal, A. K. & Gunal, S. (2014), 'The impact of preprocessing on text classification', *Information Processing & Management* **50**(1), 104–112.

Van Linh, N., Anh, N. K., Than, K. & Dang, C. N. (2017), 'An effective and interpretable method for document classification', *Knowledge and Information Systems* **50**(3), 763–793.

Vapnik, V. (2006), *Estimation of dependences based on empirical data*, Springer Science & Business Media.

Vapnik, V. (2013), *The nature of statistical learning theory*, Springer science & business media.

Venables, W. & Ripley, B. (2002), 'Modern applied statistics with s springer-verlag', *New York* .

Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T. & Saenko, K. (2015), Sequence to sequence-video to text, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 4534–4542.

Vieira, R. (2000), 'Textual co-reference annotation: a study on definite descriptions'.

Vinyals, O., Toshev, A., Bengio, S. & Erhan, D. (2015), Show and tell: A neural image caption generator, *in* 'Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on', IEEE, pp. 3156–3164.

Wagner, W. (2010), 'Steven bird, ewan klein and edward loper: Natural language processing with python, analyzing text with the natural language toolkit', *Language Resources and Evaluation* **44**(4), 421–424.

Wang, P., Xu, J., Xu, B., Liu, C., Zhang, H., Wang, F. & Hao, H. (2015), Semantic clustering and convolutional neural network for short text categorization, *in* 'Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)', Vol. 2, pp. 352–357.

Wang, X., Jiang, W. & Luo, Z. (2016), Combination of convolutional and recurrent neural network for sentiment analysis of short texts, *in* 'Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers', pp. 2428–2437.

Webster, J. J. & Kit, C. (1992), Tokenization as the initial phase in nlp, *in* 'Proceedings of the 14th conference on Computational linguistics-Volume 4', Association for Computational Linguistics, pp. 1106–1110.

Weiss, S. M. & Indurkhya, N. (1993), 'Optimized rule induction', *IEEE Expert* **8**(6), 61–69.

Wirth, R. & Hipp, J. (2000), Crisp-dm: Towards a standard process model for data mining, *in* 'Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining', Citeseer, pp. 29–39.

Yang, Y. (1999), 'An evaluation of statistical approaches to text categorization', *Information retrieval* **1**(1-2), 69–90.

Yang, Y. & Liu, X. (1999), A re-examination of text categorization methods, *in* 'Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 42–49.

Yih, W.-t., He, X. & Meek, C. (2014), Semantic parsing for single-relation question answering, *in* 'Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)', Vol. 2, pp. 643–648.

Zhang, X., Zhao, J. & LeCun, Y. (2015), Character-level convolutional networks for text classification, *in* 'Advances in neural information processing systems', pp. 649–657.

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

Zipf, G. K. (1949), 'Human behaviour and the principle of least-effort. cambridge ma edn', *Reading: Addison-Wesley* .

Zou, W. Y., Socher, R., Cer, D. & Manning, C. D. (2013), Bilingual word embeddings for phrase-based machine translation, *in* 'Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing', pp. 1393–1398.

Este trabalho foi redigido em LaTeX utilizando uma modificação do estilo IC-UFAL. As referências bibliográficas foram preparadas no Mendeley e administradas pelo BibTeX com o estilo LaCCAN. O texto utiliza fonte Fourier-GUT*enberg* e os elementos matemáticos a família tipográfica Euler Virtual Math, ambas em corpo de 12 pontos.