Pontifícia Universidade Católica
do Rio de Janeiro

Frederico Shu

APLICANDO APRENDIZADO DE MÁQUINA À SUPERVISÃO DO MERCADO DE CAPITAIS

Classificação e extração de informações de documentos financeiros

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC-Rio.

Orientador: Prof. Álvaro de Lima Veiga Filho

Rio de Janeiro,
setembro de 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Frederico Shu

APLICANDO APRENDIZADO DE MÁQUINA À SUPERVISÃO DO MERCADO DE CAPITAIS

Classificação e extração de informações de documentos financeiros

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Álvaro de Lima Veiga Filho
Orientador
Departamento de Engenharia Elétrica - PUC-Rio

Prof. Karla Tereza Figueiredo Leite
UERJ

Prof. Christian Nunes Aranha
Rede Entropia

Rio de Janeiro, 30 de setembro de 2021

Frederico Shu

Graduou-se em Engenharia de Produção na Universidade Federal do Rio de Janeiro em 1993. Cursou Mestrado em Administração no Instituto COPPEAD/UFRJ e especialização em Engenharia na Universidade de Tsukuba (Japão) entre 1994 e 1996. Trabalhou no mercado financeiro, consultoria de estratégia e comércio exterior. Analista de mercado de capitais na Comissão de Valores Mobiliários (CVM) desde 2009, atuou em supervisão de companhias abertas e educação financeira. Principal autor dos estudos "The Application of Behavioural Insights to Financial Literacy and Investor Education Programmes and Initiatives" (CVM/OCDE) e "O Mercado de Dívida Corporativa no Brasil" (CVM). Atualmente trabalha no Centro de Desenvolvimento em Ciência de Dados da CVM.

Ficha catalográfica

Shu, Frederico

Aplicando aprendizado de máquina à supervisão do mercado de capitais : classificação e extração de informações de documentos financeiros / Frederico Shu ; orientador: Álvaro de Lima Veiga Filho. – 2021.
195 f. : il. color. ; 29,7 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2021.
Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Aprendizado de máquina. 3. Processamento de linguagem natural. 4. Classificação de textos. 5. Extração de informações. 6. Aprendizado profundo. I. Veiga Filho, Álvaro de Lima. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

Agradecimentos

Ao meu orientador Professor Álvaro de Lima Veiga Filho, cujo apoio foi fundamental desde o início e durante toda a minha jornada acadêmica na PUC-Rio. Muito obrigado pela confiança depositada, pelos conhecimentos transmitidos em três disciplinas e pelo estímulo para a realização deste trabalho.

Aos meus pais Shu Tang Yung e Shu Chai Chen Mei, pela educação, paciência e incentivo em todos os momentos da minha vida.

À minha esposa Mirtes Ho, pela sua companhia cheia de paciência, compreensão e carinho.

Aos Professores Karla Figueiredo, Marley Vellasco, Cristiano Fernandes e Cláudia Freitas, por contribuírem decisivamente para a minha nova formação por meio de suas valiosas disciplinas.

Ao Professor Fabricio Mello, pela ajuda e estímulo para encarar este desafio acadêmico, além de sua amizade sempre intelectualmente estimulante.

Ao Superintendente Bruno Luna, pelo apoio e incentivo para me capacitar em prol de uma cultura voltada a dados na CVM.

Ao Marcos Cotrim, pela extração dos dados para este e outros trabalhos, e a todos que auxiliaram o meu ingresso nesta nova carreira de cientista de dados.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Código de Financiamento 001 e da PUC-Rio.

Resumo

Shu, Frederico; Veiga Filho, Álvaro de Lima. **Aplicando aprendizado de máquina à supervisão do mercado de capitais: classificação e extração de informações de documentos financeiros**. Rio de Janeiro, 2021. 195p. Dissertação de Mestrado – Departamento de Engenharia de Elétrica. Pontifícia Universidade Católica do Rio de Janeiro.

A análise de documentos financeiros não estruturados é uma atividade essencial para a supervisão do mercado de capitais realizada pela Comissão de Valores Mobiliários (CVM). Formas de automatização que reduzam o esforço humano despendido no processo de triagem de documentos são vitais para a CVM lidar com a escassez de recursos humanos e a expansão do mercado de valores mobiliários. Nesse contexto, a dissertação compara sistematicamente diversos algoritmos de aprendizado de máquina e técnicas de processamento de texto, a partir de sua aplicação em duas tarefas de processamento de linguagem natural – classificação de documentos e extração de informações – desempenhadas em ambiente real de supervisão de mercados. Na tarefa de classificação, os algoritmos clássicos proporcionaram melhor desempenho que as redes neurais profundas, o qual foi potencializado pela aplicação de técnicas de subamostragem e comitês de máquinas (*ensembles*). A precisão atual, estimada entre 20% e 40%, pode ser aumentada para mais de 90%, com a aplicação dos algoritmos testados. A arquitetura BERT foi capaz de extrair informações sobre aumento de capital e incorporação societária de documentos financeiros. Os resultados satisfatórios obtidos em ambas as tarefas motivam a implementação futura em regime de produção dos modelos estudados, sob a forma de um sistema de apoio à decisão. Outra contribuição da dissertação é o CVMCorpus, um corpus constituído para o escopo deste trabalho com documentos financeiros entregues por companhias abertas brasileiras à CVM entre 2009 e 2019, que abre possibilidades de pesquisa futura linguística e de finanças.

Palavras-chave

Aprendizado de máquina; processamento de linguagem natural; classificação de textos; extração de informações; aprendizado profundo.

Abstract

Shu, Frederico; Veiga Filho, Álvaro de Lima (advisor). **Applying Machine Learning to Capital Markets Supervision: Classification and Information Extraction from Financial Documents.** Rio de Janeiro, 2021. 195p. Dissertação de Mestrado – Departamento de Engenharia de Elétrica. Pontifícia Universidade Católica do Rio de Janeiro.

The analysis of unstructured financial documents is key to the capital markets supervision performed by Comissão de Valores Mobiliários (Brazilian SEC or CVM). Systems capable of reducing human effort involved in the task of screening documents and outlining relevant information, for further manual review, are important tools for CVM to deal with the shortage of human resources and expansion of the Brazilian securities market. In this regard, this dissertation presents and discusses the application of several machine learning algorithms and text processing techniques to perform two natural language processing tasks—document classification and information extraction—in a real market supervision environment. In the classification exercise, classic algorithms achieved a better performance than deep neural networks, which was enhanced by applying under-sampling techniques and ensembles. Using the tested algorithms can improve the current precision rate from 20%–40% to more than 90%. The BERT network architecture was able to extract information from financial documents on capital increase and mergers. The successful results obtained in both tasks encourage future implementation of the studied models in the form of a decision support system. Another contribution of this work is the CVMCorpus, a corpus built to produce datasets for the tasks, with financial documents released between 2009 and 2019 by Brazilian companies, which opens possibilities of future linguistic and finance research.

Keywords

Machine learning; natural language processing; text classification; information extraction; deep learning.

Sumário

Lista de figuras

Lista de tabelas

Lista de gráficos

1
Introdução

1.1.
Contexto e motivação

O mercado de capitais, ou mercado de valores mobiliários, é a subdivisão do mercado financeiro onde os agentes com recursos financeiros excedentes para investir os emprestam diretamente aos agentes deficitários em capital. Os valores mobiliários (p. ex., ações e cotas de fundos de investimento) são títulos entregues aos investidores que representam as condições estabelecidas no empréstimo.

A Comissão de Valores Mobiliários (CVM), criada em 1976, é a autarquia federal responsável pela fiscalização, normatização, disciplina e desenvolvimento do mercado de capitais brasileiro.

A empresa que deseja emitir ações ou outros valores mobiliários deve se registrar previamente na CVM. Ao obter o registro, ela ganha a denominação de companhia de capital aberto ou, simplesmente, companhia aberta.

Um dos mandatos da CVM é assegurar que todos os investidores possam ter acesso a informações corretas, claras e completas. Quando as companhias disponibilizam as suas informações no portal da CVM[1] e no seu próprio sítio, assume-se que estas últimas se tornam públicas. Pode-se afirmar que a CVM possui em seu banco de dados corporativo todas as divulgações realizadas pelas companhias abertas brasileiras, por força das normas vigentes.

O desenvolvimento de sistemas de apoio à decisão é estratégico para a CVM. O uso de inteligência e novas tecnologias para melhorar as atividades de supervisão e fiscalização é um objetivo finalístico do órgão. Racionalizar a alocação de recursos humanos é fundamental para ele enfrentar os desafios impostos pela austeridade fiscal e esperada expansão do mercado de capitais.

1 http://www.cvm.gov.br

A título de ilustração, o patrimônio total dos fundos de investimento brasileiros aumentou de R$ 1,4 trilhão em 2009 para R$ 5,5 trilhões em 2019, enquanto a quantidade de fundos ativos passou de 8.798 para 19.299 no mesmo intervalo[2]. O volume crescente de informações demandadas das companhias abertas, por conta da evolução do mercado, reforça a necessidade de desenvolver ferramentas computacionais de análise de documentos. Um exemplo é a audiência pública promovida pela CVM em março de 2021, para discutir a futura exigência de dados ambientais, sociais e de governança corporativa (ESG).

1.2. Processamento de linguagem natural, aprendizado de máquina e o contexto do mercado de capitais

A importância do processamento de linguagem natural (PLN) – definida por Jurafsky e Martin [1] como o campo interdisciplinar que estuda o desempenho automatizado de tarefas úteis envolvendo linguagem humana – para a tomada de decisões financeiras é crescente. Segundo Lewis e Young [2], a análise calcada unicamente em métricas, índices e outros dados quantitativos é incapaz de capturar as nuances contidas nas comunicações realizadas em linguagem não estruturada pelos agentes econômicos. Portanto, o conteúdo qualitativo da comunicação de eventos financeiros representa insumo importante para os participantes do mercado financeiro realizarem avaliações e pesquisas, assim como planejarem e executarem ações diversas[3]. O PLN permite a avaliação sistemática de grandes volumes de documentos financeiros para encontrar padrões, realizar comparações, observar tendências e identificar pontos relevantes.

A expansão, ocorrida nas duas últimas décadas, do volume de informações não estruturadas que são disponibilizadas para o mercado financeiro reforça a importância do uso de técnicas automatizadas de processamento de texto. Alguns exemplos dessas informações são conferências telefônicas sobre resultados

2 https://www.anbima.com.br/pt_br/informar/estatisticas/fundos-de-investimento/fi-consolidado-historico.htm (consulta realizada em 10.11.2020 às 17h33).

3 O caso das demonstrações financeiras ilustra bem a importância da informação qualitativa. As notas explicativas são sempre iniciadas com o aviso de que representam parte integrante das demonstrações contábeis e costumam preencher um número de páginas maior que o ocupado pelas tabelas, balanços e demonstrativos.

financeiros, *press releases*, relatórios, além de conteúdos disponíveis em portais financeiros e em mídias sociais. Esse volume crescente representa desafios e oportunidades para investidores, governos e pesquisadores.

O emprego de algoritmos de aprendizado de máquina (AM) representou um avanço relevante para a resolução de problemas de PLN, principalmente em comparação com os métodos baseados em regras, cuja confecção exige o conhecimento de especialistas [3]. O emprego de regras também está sujeito ao risco de o seu conhecimento originador estar incompleto ou sujeito a vieses pessoais dos especialistas. O AM possibilitou o desenvolvimento de sistemas especializados (*expert systems*) mais flexíveis e escaláveis, cuja correção e manutenção não necessariamente dependem de intervenção humana.

O corpus, definido pelo linguista Sinclair [4] como "uma coleção de textos em formato eletrônico e selecionados para caracterizar um idioma"[4], provê a base material e o campo de testes para o desenvolvimento de sistemas de PLN [5]. Portanto, especial atenção deve ser dada à sua construção. Segundo McEnery [6], a escolha do regime de coleta de dados deve ser criteriosa e é necessário observar os princípios de balanceamento, representatividade e comparabilidade na etapa de montagem do corpus. Lewis e Young argumentam que a maior barreira à adoção mais ampla das técnicas de PLN nas pesquisas sobre informações financeiras é a ausência de condições de replicabilidade. Para lidar com esse problema, é preciso reportar com detalhamento adequado os procedimentos de construção dos *corpora* utilizados. Devido à linguagem peculiar empregada no domínio das finanças, a criação de *corpora* especializados é praticamente mandatória para se conseguir responder às perguntas de pesquisa desejadas.

O PLN e a decorrente utilização de corpus apresentam um rol de aplicações promissoras para a atividade de supervisão dos mercados financeiros. Especificamente na área de *Suptech*, definida como o uso de tecnologias inovadoras no acompanhamento de mercados, existem iniciativas pioneiras que permitem aos reguladores aumentar sua capacidade preditiva, economizar recursos e melhorar a qualidade de seus dados.

4 Tradução livre de "A corpus is a collection of pieces of language text in electronic form, selected according to external criteria to represent, as far as possible, a language or language variety as a source of data for linguistic research." [4]

1.3.
Objetivos da dissertação e formulações dos problemas

A CVM trabalha com a metodologia de Supervisão Baseada em Risco (SBR) para priorizar a sua atuação. A metodologia direciona a atuação para a prevenção dos riscos que representam ameaças ao cumprimento dos mandatos legais da CVM.

Em seu Plano Bienal de SBR, a CVM estabeleceu ações específicas de acompanhamento de operações societárias, por alterarem a natureza ou a estrutura social das empresas. Geralmente propostas por administradores e/ou acionistas controladores de companhias, tais operações podem trazer consequências desvantajosas aos acionistas minoritários, como o fenômeno da "diluição injustificada". Esta é definida como a redução da participação acionária percentual do investidor, de forma anômala e geralmente involuntária. Tal redução acarreta perda de valor econômico da participação detida.

Um tipo de operação societária relevante é a reorganização societária. Há três modalidades principais de reorganizações societárias:

i. Incorporação: é o tipo mais frequente e consiste na absorção de uma empresa por outra.
ii. Cisão: consiste na transferência de parte do patrimônio de uma empresa para outra.
iii. Fusão: é a união de duas companhias para formar uma nova sociedade.

Outro tipo de operação societária relevante, também por acarretar diluição injustificada, é o aumento de capital pela emissão de ações (ou outros instrumentos conversíveis em ações) por meio de subscrição privada, ou seja, sem oferta pública.

A triagem dos documentos fornecidos pelas companhias abertas é a etapa preparatória para o processo de supervisão de operações societárias. Atualmente, os analistas da CVM contam com uma ferramenta de *business intelligence* para selecionar documentos, por busca de palavras-chave contidas no assunto do documento, o qual é de preenchimento livre pela companhia. A ferramenta não examina o teor do documento. Por ser baseado em regras, o processo atual de

seleção está sujeito à ocorrência de falsos negativos, ou seja, textos com conteúdo relevante que não foram identificados pelo mecanismo de triagem por não apresentarem palavras-chave. O processo também gera falsos positivos, quando a presença da palavra-chave destaca um documento que não é de interesse. Como o custo de não capturar um documento importante para a supervisão é alto, os analistas da CVM utilizam uma gama abrangente de palavras-chave e consequentemente estima-se que 70 a 80% dos documentos selecionados são falsos positivos, i.e., o processo atual apresenta precisão entre 20 e 30%.

Na sequência, o analista realiza uma pré-análise de cada documento selecionado para identificar características das operações societárias que, ao serem confrontadas com critérios estabelecidos, definirão se é necessário ou não abrir um processo administrativo de acompanhamento da operação. A ferramenta atual não infere o tipo de operação societária abordada no documento.

Portanto, pode-se modelar a triagem de documentos, que compreende as etapas de seleção e pré-análise, na forma de dois problemas ou tarefas de PLN:

i. um problema de **classificação** de documentos financeiros; e
ii. um problema de **extração de informações** sobre operações societárias.

A dissertação testa a aplicação de algoritmos de AM no processo de triagem de documentos. A aplicação resolve os dois problemas de PLN sobre uma base de documentos contendo deliberações e características dos dois tipos de operações societárias descritos anteriormente:

1. **reorganização societária**; e
2. **aumento de capital**.

O problema proposto de classificação de documentos financeiros consiste em selecionar os documentos que contêm informações sobre operações societárias e inferir os seus tipos.

O problema proposto de extração de informações sobre operações societárias compreende a identificação de duas características, descritas a seguir, presentes nos dois tipos de operações.

a) Nas operações de **incorporação**, deseja-se determinar **o agente e o paciente do evento de incorporação**. Em outras palavras, buscaram-se o nome da companhia que realizou a incorporação e o da que foi incorporada.
b) Nas operações de **aumento de capital**, almeja-se identificar **a quantidade de novas ações emitidas**.

Os resultados da dissertação, consubstanciados nos modelos de AM que obtiveram os melhores resultados nos dois problemas, serão empregados em um sistema de PLN que avaliará documentos submetidos pelas companhias abertas à CVM, em substituição ao processo vigente de triagem.

1.4. Contribuições da dissertação

Como será visto no Capítulo 2, a literatura corrente compreende pesquisas sobre PLN e AM potencialmente conversíveis em aplicações úteis para a supervisão de mercados. A literatura também reúne descrições de ferramentas de PLN e AM utilizadas por reguladores estrangeiros sem maior detalhamento técnico. Nesse contexto, a dissertação provê as seguintes contribuições.

i. Preenche uma lacuna da bibliografia, por ser o primeiro registro científico de aplicação de AM na execução de duas tarefas de PLN desempenhadas em ambiente **real** de supervisão do mercado de capitais.
ii. Constitui um trabalho pioneiro de PLN baseado em um corpus de grande porte em língua portuguesa pertencente ao domínio financeiro, formado pelo universo de documentos divulgados por companhias abertas brasileiras.
iii. Efetua uma comparação sistemática de diversas técnicas e algoritmos de AM, assim como de métodos de PLN, no tocante à aplicação e ao desempenho obtido nas tarefas propostas.

1.5. Estrutura da dissertação

A dissertação compõe-se de sete capítulos. A introdução explica o contexto formado pelo mercado de capitais, a CVM e sua relação com o PLN e o AM, além dos problemas abordados neste trabalho. O segundo capítulo comenta os trabalhos relacionados que versaram sobre o emprego de técnicas de AM e/ou que construíram *corpora* específicos de domínio para uso em pesquisas e aplicações no mercado de capitais mundial. O referencial teórico composto pelos algoritmos aplicados na resolução dos problemas propostos e alguns conceitos básicos de AM e PLN é apresentado no terceiro capítulo. A quarta parte descreve a metodologia empregada na resolução do problema de classificação de documentos financeiros, assim como apresenta e discute os resultados obtidos. O quinto capítulo aborda a resolução do problema de extração de informações sobre operações societárias. Apresenta-se no sexto capítulo outro produto da dissertação, consubstanciado em um corpus de domínio financeiro: o CVMCorpus. O sétimo capítulo conclui a dissertação e discute futuros possíveis trabalhos.

2
Aplicações de processamento de linguagem natural no mercado de capitais

Este capítulo aborda os trabalhos científicos e aplicações realizadas por reguladores do mercado de capitais com abordagens semelhantes à adotada nesta dissertação, isto é, que utilizam técnicas de AM e/ou corpora de domínio especialmente construídos para a resolução de problemas de PLN aplicáveis ao mercado de capitais.

2.1.
Construção de corpora no domínio do mercado de capitais

Um corpus específico de domínio, ou simplesmente corpus de domínio, compreende um conjunto de textos que pode ser considerado suficientemente representativo de uma área específica, qual seja, o próprio domínio.

A posse de um corpus especializado permite elaborar uma terminologia do domínio abordado nos textos de modo automatizado e sem o viés a que um especialista humano eventualmente estaria sujeito, caso ele fosse encarregado que elencar os termos relevantes. No entanto, a validação humana, ainda assim, é necessária. A partir da terminologia ou concomitantemente à sua elaboração, pode-se construir uma ontologia – recurso semântico destinado a conceituar, estruturar e representar claramente os conhecimentos de um domínio para que possam ser compartilhados. A extração de termos também possibilita gerar glossários automaticamente, produzir mecanismos de busca, classificar textos, entre outras aplicações.

No entanto, o primeiro, e talvez o mais imediato, motivo para construir um corpus de domínio é levantado por Cohen et al. [7] e Thompson et al. [8], quando

defendem que as soluções de PLN válidas para o domínio geral usualmente fracassam ao serem aplicadas a áreas especializadas e que as características dos dados textuais em domínios especializados diferem de formas e em níveis variados daquelas dos textos em linguagem geral. Outro argumento a favor do uso de um corpus de domínio é o fato de as palavras frequentemente possuírem sentidos distintos, conforme os contextos em que são empregadas. Por exemplo, no campo financeiro o termo "volátil" caracteriza grande variação (em geral) de preço, em vez da propriedade química de se reduzir a vapor. Além disso, é comum que palavras habitualmente usadas como sinônimos no cotidiano indiquem conceitos diferentes dentro de determinados domínios, como é o caso de "despesa" e "custo" em contabilidade e finanças.

Além disso, a linguagem das comunicações financeiras é altamente especializada, dada a profusão de termos técnicos, siglas, abreviaturas e palavras em inglês, como se observa em outros campos do conhecimento. Não é de se estranhar, portanto, que os estudos dedicados a compreender as relações entre linguagem e fenômenos financeiros demandem a utilização de corpora específicos do domínio.

O caso da língua portuguesa aparenta ser crítico para a linguística de corpus de domínio. Lopes [9] afirma que corpora científicos em vernáculo são escassos e disponibiliza em seu portal[5] nove conjuntos, porém nenhum deles versa sobre finanças ou economia.

A aplicação do princípio de *full and fair disclosure*, segundo o qual a melhor forma de proteger o investidor é disponibilizar-lhe todas as informações necessárias para tomar decisão de investimento, proporciona material público sobre empresas farto e rico em informações, o qual tem se constituído em fonte de dados interessantes para estudos. Kogan et al. [10] montaram um corpus com relatórios financeiros anuais submetidos pelas companhias abertas listadas no mercado norte-americano à *Securities and Exchange Commission* (SEC), órgão regulador do mercado de capitais dos EUA. Seu objetivo foi inferir como os comentários sobre as situações operacionais e financeiras, divulgados pelas companhias nos formulários 10-K, influenciavam a volatilidade dos retornos de

5 https://www.inf.pucrs.br/peg/lucelenelopes/ll_crp.html

suas ações – uma medida de risco financeiro. Com modelos de AM treinados e testados no conjunto de 248 milhões de tokens distribuídos em 27 mil formulários públicos, os pesquisadores obtiveram desempenho igual ou superior ao obtido com o valor-base, constituído pela volatilidade histórica. Os comentários dos formulários 10-K também foram analisados por Humpherys et al. [11] em estudo que gerou mais de 70 citações. Os pesquisadores observaram que conteúdos fraudulentos apresentaram com maior grau de exagero nos aspectos positivos e de dissimulação dos pontos negativos, além de usarem palavras mais longas e maior número de pausas (marcas de pontuação por frases), na tentativa de distrair o leitor com informações irrelevantes, apesar de a diversidade lexical ser menor. Lee et al. [12] montaram um corpus que alinhou temporalmente divulgações de eventos financeiros, presentes em 13.600 relatórios 8-K entregues à SEC, e variações de preços de ações de companhias do índice S&P 500. Eles constataram que os modelos de previsão melhoraram seu desempenho ao incorporar atributos linguísticos, baseados na presença de unigramas de conotação negativa nos textos.

Dentre os trabalhos que consistiram primariamente na montagem e disponibilização pública de corpora de domínio, destaca-se o de Daudert & Ahmadi [13], que reuniu[6] 2.600 relatórios financeiros (180 milhões de tokens), divulgados pelas 60 maiores companhias francesas listadas em bolsa, para gerar o primeiro corpus em francês do domínio financeiro. Os autores demonstraram o potencial do corpus para aplicações em PLN, por meio de experimentos envolvendo o modelo de vetorização de palavras Word2vec e predição de erros gramaticais em frases. El Haj et al. [14] desenvolveram um método automatizado para extrair e compartimentar em seções os conteúdos de relatórios anuais de empresas listadas na bolsa de Londres, dispostos em arquivos PDF. Os pesquisadores disponibilizaram publicamente um corpus[7] constituído por mais de 31.000 relatórios (197 milhões de palavras) publicados entre 2002 e 2017 e sua ferramenta de extração[8].

No tocante a estudos brasileiros, cabe destacar o estudo pioneiro de 2016 de Garcia [15], que investigou o fenômeno da ambiguidade lexical presente nos

6 https://github.com/CoFiF/Corpus
7 http://www.research.lancs.ac.uk/portal/en/datasets/annual-reports-key-sections-corpora-2003-to-2017(ee12c021-eddd-4258-a682-8daf3e4bbe90).html
8 https://github.com/drelhaj/CFIE-FRSE

relatórios contábeis com o uso de dois corpora: o primeiro era composto pelo conjunto de normas brasileiras que rege a produção de demonstrativos financeiros e o segundo, pelos relatórios financeiros relativos ao ano de 2013, emitidos por seis companhias abertas nacionais. O autor observou que termos contábeis consagrados como “lucro” e “despesa” adquiriam sentidos diferentes conforme o tipo de relatório e a empresa emitente. Ele concluiu que a contabilidade não consegue transmitir seu conteúdo de forma clara, inequívoca e uniforme aos seus usuários (administradores, sócios, investidores etc.). Aguiar [16] reuniu todos os relatórios de administração entre 1995 e 2009 das companhias listadas na antiga Bovespa para examinar se o seu teor ajudava a explicar o comportamento das ações sem, contudo, ter encontrado evidências suficientes. Alves [17] montou um corpus com mais de dois milhões de mensagens em português publicadas na rede social Twitter entre 2013 e 2015, a respeito de nove companhias abertas brasileiras, para testar um simulador de compra e venda de ações usando técnicas de análise de sentimentos. Apesar da variedade de estudos citados anteriormente, a aplicação de métodos de linguística computacional na análise da comunicação oral e escrita sobre temas financeiros apresenta diversas lacunas que poderiam ser preenchidas ao utilizar um corpus de domínio. Em artigo de 2019, El Haj et al. [18] apontam que a falta de corpora especializados (mesmo em inglês) restringe a replicabilidade e o avanço incremental da pesquisa científica. Na ausência de um corpus consolidado, não se sabe se uma lista de palavras ou expressões desenvolvida a partir de uma fonte (p. ex., balanços de empresas) é válida para analisar aspectos linguísticos de outra (como os fatos relevantes, ainda que publicados pelas mesmas companhias). Os autores consideram a linguística de corpus como uma das quatro ferramentas de linguística computacional capazes de impulsionar o estudo do significado no discurso financeiro. Os pesquisadores ressaltam que as divulgações de firmas norte-americanas não representam necessariamente uma linguagem financeira internacional (porque a regulação, as práticas comerciais e as normas culturais diferem entre os países), o que reforça a necessidade de geração de corpora de domínio em vernáculo.

2.2.
Aplicações baseadas em aprendizado de máquina

Em sua revisão de literatura, Koshiyama et al. [19] listaram seis áreas de aplicações de algoritmos de AM no mercado de capitais:

i. PLN
ii. Detecção de fraudes
iii. Regulação / *Compliance*
iv. Gerenciamento de riscos
v. Otimização de portfólios
vi. Negociação automatizada

Serão analisados a seguir os dois primeiros temas devido à sua pertinência com a supervisão de mercados e com o problema de classificação.

O problema se enquadra na subdivisão do PLN denominada mineração de textos, definida por Moreno e Redondo [20] como a extração automatizada de informações novas a partir de textos. Bach et al. [21] revisaram 123 artigos publicados de 2005 a 2019 que tratavam de utilização de métodos de mineração de textos em finanças e destacaram as seguintes técnicas:

i. extração de palavras-chave, que inclui a transformação de frases em conjunto de tokens ou unidades fundamentais de processamento de textos (p. ex.: palavras e numerais);
ii. reconhecimento de entidades mencionadas (*named-entity recognition*);
iii. predição de gênero do interlocutor para aperfeiçoar a comunicação com os clientes e o planejamento de ações de marketing;
iv. análise de sentimentos;
v. extração de tópicos; e
vi. análise de redes sociais.

Alguns exemplos são trazidos a seguir. McClane [22] identificou a repetição contínua de textos (*boilerplate*) em prospectos de ofertas públicas por buscas de palavras-chave e cálculo de similaridade por cosseno, e em seguida aplicou a análise de componentes principais (PCA) para agrupar os trechos semelhantes,

possibilitando-lhe distinguir os trechos relevantes dos genéricos. O artigo concluiu que os esforços da CVM norte-americana (*Securities and Exchange Commission* ou SEC) para simplificar a linguagem dos prospectos foram bem-sucedidos por um período de cinco anos e tiveram impacto positivo na precificação dos ativos emitidos, além de recomendar o emprego de ferramentas de PLN e AM pelo regulador para identificar boas práticas de divulgação de informações. Badawi [23] aplicou algoritmos de AM (Naïve Bayes, máquinas de vetores de suporte ou SVM, *Random Forests* etc.) para prever quais casos judiciais resultariam em acordos ou seriam favoráveis à companhia reclamada, a partir da análise da petição inicial da reclamação, obtendo mais de 70% de acurácia.

A detecção de fraudes e outras irregularidades no mercado de capitais é uma aplicação da tarefa de classificação de texto bastante pesquisada com o auxílio de ferramentas textuais e de AM. Brown et al. [24] identificaram os tópicos abordados nos comentários que os administradores realizaram sobre o desempenho financeiro de suas companhias (os chamados formulários 10-K que as empresas submetem à SEC), com o uso do modelo *Latent Dirichlet Allocation* (LDA). Os tópicos foram úteis para identificar violações às regras de divulgações de resultados financeiros. Hoberg e Lewis [25] aplicaram o LDA aos formulários 10-K publicados durante 13 anos e concluíram que empresas que emitem comentários menos detalhados e com maior ênfase em aspectos positivos possuem maior chance de produzir fraudes contábeis. Li et al. [26] testaram sete algoritmos supervisionados de AM e identificar as negociações de ações que apresentaram tentativas de manipulação de mercado, segundo rotulagem realizada pela CVM da China. Wang et al. [27] aplicaram redes neurais recorrentes (RNN) para identificar casos de manipulação de mercado, obtendo melhores resultados que os algoritmos tradicionais, como Naïve Bayes, SVM e *K-Nearest Neighbors*. Dong et al. [28] também utilizaram a análise de tópicos para, junto com outros atributos linguísticos, treinar modelos SVM que classificaram documentos conforme a presença de indícios de fraude.

Segundo Koshiyama et al., existem poucas aplicações de extração de informações no domínio do mercado de capitais que contemplam a arquitetura de rede neural profunda *Bidirectional Encoder Representations from Transformers*

(BERT), a ser estudada no Capítulo 3. Merecem destaque o artigo de Araci [29], que construiu o FinBERT, um modelo de linguagem para desempenhar tarefas de PLN com conteúdo financeiro. O modelo foi treinado em um corpus composto por 1,8 milhão de artigos financeiros da Reuters e obteve performance superior às de outros tipos de redes neurais profundas em tarefas de classificação de frases, títulos de notícias e *tweets*. O FinBERT conseguiu classificar sentimentos satisfatoriamente mesmo dispondo de um conjunto diminuto de treinamento (500 observações), graças ao conhecimento adquirido do corpus. Hiew et al. [30] desenvolveram um índice de sentimento sobre ações individuais com base em um modelo BERT treinado com publicações em redes sociais. Koshiyama et al. vislumbram ser possível desenvolver aplicações do BERT nas seis áreas por eles citadas.

Os reguladores dos mercados financeiros também pesquisam ativamente aplicações de inteligência artificial em *Suptech*. O *Financial Stability Board* (FSB) – órgão internacional que monitora e emite recomendações para o sistema financeiro mundial – destaca o uso de PLN pela CVM australiana (*Australian Securities and Investments Commission* ou ASIC) [31] para extrair entidades e seus relacionamentos mencionados em textos não estruturados, com a finalidade de detectar indícios de atividades fraudulentas e lavagem de dinheiro. A autoridade monetária de Singapura (*Monetary Authority of Singapore* ou MAS) [32] criou um sistema para detectar indícios de lavagem de dinheiro e outros tipos de atividades suspeitas em transações financeiras. A SEC emprega algoritmos não supervisionados para analisar o tom da linguagem utilizada nas comunicações provenientes de consultores de investimento, de forma a identificar riscos presentes no comportamento desses participantes do mercado [31]. A Unidade de Inteligência Financeira da Itália (UIF) [33] está testando um modelo que agrupa participantes do mercado de ouro e utiliza análise de componentes principais (PCA) para identificar participantes com comportamento similar a outros que praticaram ilícitos relacionados com lavagem de dinheiro ou financiamento a terrorismo. O banco central holandês (DNB) [34] testou modelos de regressão logística e redes neurais profundas do tipo *Multilayer Perceptron* (MLP) para

detectar situações potenciais de *stress* bancário, tendo obtido resultados satisfatórios com dados passados.

3
Referencial teórico

Este capítulo descreve conceitos básicos de PLN que compõem o arcabouço teórico necessário para se trabalhar com textos não estruturados. Em seguida, apresentam-se a terminologia e os algoritmos de AM empregados na dissertação, assim como as principais técnicas de subamostragem, visto que a resolução do problema de classificação lidou com dados desbalanceados. Será dada ênfase às ferramentas e conceitos aplicáveis a problemas de classificação e extração de informações.

3.1.
Conceitos básicos de processamento de linguagem natural

Esta seção aborda conceitos básicos de PLN sob o ponto de vista prático e aplicado na resolução dos problemas propostos.

3.1.1.
Pré-processamento

Em geral, dados brutos provenientes de linguagem humana não estão prontos para serem processados por algoritmos de AM. O texto escrito ou falado é corrido e não estruturado, ou seja, não se encontra dividido em campos legíveis por máquinas. Além disso, as bibliotecas que implementam algoritmos de AM exigem que as palavras estejam dispostas em determinados formatos de dados, como matrizes (*arrays*) ou listas, e é recomendável que algumas palavras ou elementos do texto sejam removidos ou modificados para se obter o melhor desempenho dos algoritmos. Portanto, é necessário um **pré-processamento** que

consiste em adequar os dados de linguagem natural. As etapas de pré-processamento a serem desempenhadas (assim como a sua ordem de execução) dependem do idioma da linguagem trabalhada, do formato e do estado em que os dados de linguagem se encontram, do tipo de problema a ser resolvido, do domínio dos dados e outros fatores.

No caso de texto escrito, uma etapa importante de pré-processamento é a **tokenização** ou transformação de texto corrido em tokens. Um **token** ou **termo** é a unidade fundamental de processamento de textos. Ele usualmente definido como uma palavra, um número, um sinal de pontuação ou um elemento gráfico do texto. Normalmente identifica-se os tokens por estarem apartados por espaços no texto mas os sinais de pontuação costumam vir junto de outro token. Há problemas em que é interessante analisar **bigramas** (dois tokens adjacentes), **trigramas** (3 tokens seguidos) e/ou **n-gramas** presentes nos textos e passíveis de serem considerados como um token, devido ao seu significado particular distinto dos significados dos seus tokens constituintes. Por exemplo, o bigrama "Casa Branca" possui uma conotação distinta dos significados de "casa" e "branca", sendo comum representá-lo como o token Casa_Branca. Outro exemplo é a decomposição de contrações comuns no português, como "d'água" e "nele", útil para tarefas de PLN que usam informações sintáticas.

Outra etapa de pré-processamento é a **segmentação** do texto em frases e parágrafos. Diversos algoritmos e representações de linguagem necessitam que o texto de entrada seja particionado e a escolha natural recai sobre esses dois tipos de segmentos. O ponto costuma separar frases, porém abreviaturas, alíneas, números e outros elementos são situações ambíguas que podem demandar a confecção de regras ou até o emprego de algoritmos de AM para serem resolvidas.

A **limpeza** de texto representa um passo importante de pré-processamento que elimina os elementos com nenhum ou pouco conteúdo semântico. No primeiro caso enquadram-se usualmente os cabeçalhos, números de página, espaços em excesso, marcas de parágrafo, formatação HTML etc. O segundo caso ocorre em tarefas de PLN para as quais é interessante remover tokens ou elementos gráficos irrelevantes, como pontuação, números e até as chamadas

stopwords **–** palavras de alta frequência como "de", "o" e outros artigos e preposições com pouco significado.

A **normalização** do texto consiste em uniformizar as formas das palavras. Transformar todas as letras em minúsculas é um procedimento simples que elimina a ambiguidade causada por substantivos comuns no início de frases mas dificulta a identificação de nomes próprios. A **radicalização** (mais comumente referida como ***stemming***) é a redução das palavras à raiz, por exemplo, "aprendizado" e "aprendeu" são diminuídas para "aprend". A **lematização** reduz as palavras à forma de dicionário ou lema: "aprendeu" se torna "aprender" e "gata" é convertida para "gato". Os dois procedimentos anteriores podem ser executados com o auxílio de regras, listas, dicionários e algoritmos.

3.1.2. Representações numéricas de tokens

Algoritmos de AM trabalham com representações numéricas. Por esse motivo, as tarefas de PLN baseadas em AM demandam um pré-processamento especial para transformar linguagem em dados numéricos. O modelo de representação (ou **vetorização**) adotado pode influenciar diretamente o desempenho do algoritmo na execução de uma tarefa de PLN.

A representação numérica de tokens e documentos mais comum faz uso do **modelo de espaço vetorial**, que representa os documentos a serem analisados em um espaço vetorial comum, ou seja, cada documento será expresso por um vetor de números. Existem diversas formas de montar esse vetor e um procedimento intuitivo é se basear nos tokens presentes nos documentos. Desse modo, um conjunto de documentos (de treinamento ou teste, p. ex.) é representado por uma **matriz documento-termo** (***document-term matrix***) onde cada linha representa um documento e cada coluna, um token. Dessa forma, as colunas representam todos os termos que figuram ao menos em um documento, ou seja, formam o **vocabulário** do conjunto de documentos.

As representações numéricas usuais são:

i. ***One-hot encoding***: é um modo intuitivo de representação que consiste em codificar a presença ou não do token com os valores um e zero. A matriz

documento-termo resultante é, portanto, esparsa e binária. Por atribuir peso ou importância igual a todos os tokens com valor um e o mesmo ocorrendo para o valor zero, essa representação possui pouco poder de discriminação e por isso sua aplicação é limitada.

ii. ***Bag-of-words*** (**BOW**): utiliza a frequência absoluta dos termos em cada documento. Dessa forma, a matriz resultante ainda é esparsa mas com variados valores inteiros. BOW ou saco de palavras traz a ideia de que os tokens estão misturados, pois a representação é incapaz de expressar as suas posições nas frases. Geralmente, as colunas da matriz documento-termo são dispostas em ordem alfabética.

iii. **Tf-idf**: pondera a frequência absoluta f(t) de cada token t para evitar que ela atribua peso desproporcional à sua relevância. Mesmo depois de eliminar as *stopwords*, é usual que os documentos ainda apresentem termos muito frequentes em comum, os quais não servem para distinguir os textos. Para refletir esse aspecto, emprega-se o conceito de frequência inversa dos documentos ou *inverse document frequency* (idf). O idf de t é dado por:

$$idf(t) = \ln \frac{N}{df(t)} \tag{1}$$

Na equação, N é o número total de documentos e o *document frequency* df(t) é a quantidade de documentos que apresentam t pelo menos uma vez. Assim, se t está presente em muitos documentos, ele apresentará alto df(t) e, consequentemente, baixo idf(t). O índice tf-idf é obtido ponderando-se f(t) por idf(t), como mostrado em (2). Tokens que ocorrem muitas vezes em um número limitado de documentos apresentarão tf-idf elevado e terão alto poder de discriminação sobre esses textos.

$$tf\text{-}idf(t) = f(t) \cdot idf(t) = f(t) \cdot \ln \frac{N}{df(t)} \tag{2}$$

Representa-se cada documento i por um vetor v_i de p índices tf-idf, o qual é usualmente normalizado pela sua norma euclidiana, segundo (3). Dessa forma, documentos longos e curtos do mesmo corpus serão todos representados por vetores de comprimento unitário, eliminando eventuais

influências do tamanho do documento no desempenho das tarefas de PLN subsequentes.

$$v_{i\,norm} = \frac{v_i}{\sqrt{\sum_{t=1}^{p} (tf\text{-}idf(t))^2}} \tag{3}$$

A biblioteca Scikit-learn [35] aplica suavização ao idf(t) para calcular o tf-idf de um termo t. É somado um ao numerador e ao denominador de (1), o que equivale a criar um documento extra contendo t. O Scikit-learn ainda soma um ao resultado do logaritmo, para evitar para que o idf(t) de um termo presente em todos os documentos seja nulo. Os vetores resultantes são normalizados em relação a todos os p documentos da base apresentada à biblioteca, segundo (3). A fórmula final do td-idf(t) é:

$$tf-idf(t) = \frac{f(t)}{1+\ln\left(\frac{N+1}{df+1}\right)} \left(\sum_{i=1}^{p} [tf-idf(t)]^2\right)^{-\frac{1}{2}} \tag{4}$$

iv. **Word2vec**: método proposto em 2013 por Mikolov et al. [36], então pesquisadores do Google, destaca-se por representar cada token por um vetor dotado de informações semânticas. O vetor é determinado de forma não supervisionada com um corpus escolhido pelo usuário, usando uma rede *Multilayer Perceptron* (vide seção 3.2.6.2.1) com uma camada escondida com p unidades. O usuário deve designar um dos dois tipos de tarefas a seguir para a rede:

iv.1. ***Continuous Bag-of-Words*** (CBOW): a rede percorre as frases do corpus com uma janela móvel de n tokens adjacentes, que formam um contexto. A rede tenta predizer um token do contexto baseado no conhecimento dos demais (n-1) tokens. Portanto, a rede receberá como entrada (n-1) vetores *one-hot* de comprimento N (número de tokens do vocabulário do corpus). A rede prediz N probabilidades do token desconhecido ser igual a cada um dos N tokens do vocabulário e atualiza seus pesos constituintes, com base na identificação real do token analisado. Todos os contextos possíveis do corpus são

submetidos à rede e as etapas de predição e atualização, repetidas até os pesos convergirem. Os pesos que unem a camada escondida à camada de saída (que prediz as probabilidades) formam uma matriz N x p. Cada linha da matriz conterá o vetor representativo de um token do vocabulário.

iv.2. ***Skip-gram***: pode ser entendido como o inverso do CBOW. A rede realiza o mesmo percurso com janela móvel e a diferença consiste em tentar predizer os (n-1) tokens do contexto, conhecendo apenas um deles. As etapas de predição e atualização de pesos são análogas às do CBOW. Após convergência, os pesos que unem a entrada à camada escondida formam a matriz de vetores representativos dos tokens do vocabulário.

O treinamento em janelas móveis provê conteúdo semântico aos vetores. Por exemplo, se no corpus o token "conselho" é frequentemente acompanhado por "fiscal", os vetores dos dois tokens serão próximos ou similares e seu produto escalar terá valor perto de um.

Nos três primeiros métodos de representação, pode-se considerar as colunas da matriz documento-termo como vetores representativos dos termos, com comprimento igual ao número total de documentos do conjunto de treinamento ou teste, podendo chegar a milhares de documentos. Para o word2vec, é usual adotar comprimento igual a 100, 200 ou 300, portanto esse método consegue mapear os vetores de um espaço grande de documentos para outro de menos dimensões. Esse mapeamento para dimensões reduzidas chama-se ***embedding*** e os vetores de tokens resultantes são denominados ***word embeddings***. O conceito de transferência de aprendizado também é aplicável a *word embeddings*: eles podem ser gerados por treinamentos em grandes corpora e fornecidos para uso por terceiros, em forma de um dicionário de termos e vetores. *Word embeddings* pré-treinados em diversos idiomas estão disponíveis na Internet[9].

9 O portal http://www.nilc.icmc.usp.br/embeddings fornece *word embeddings* para o português, por exemplo.

3.1.3.
Critérios para seleção de atributos em PLN

Manning et al. [37] apresentam dois critérios principais de seleção de atributos para problemas de classificação de texto. O primeiro deles se baseia na **informação mútua** (IM), que mensura o volume informacional que a ausência ou a presença de um token contribui para a decisão de classificação. A IM de um termo t é definida por:

$$I(U,C)=\sum_{e_t\in(1,0)}\sum_{e_c\in(1,0)} p(U=e_t,C=e_c)\log_2\frac{p(U=e_t,C=e_c)}{p(U=e_t)p(C=e_c)} \tag{5}$$

Na fórmula, U e C são variáveis aleatórias que assumem valores e_t=1 e e_c=1 se o documento contém t e pertence à classe c, respectivamente, e e_t=0 e e_c=0 caso contrário. Utilizando os estimadores de máxima verossimilhança das probabilidades, (5) pode ser reescrita como:

$$\begin{aligned} I(U,C)=&\frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_1N_1}+\frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_0N_1}+\frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_1N_0}+\\ &+\frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_0N_0},\\ &onde\ N_1=N_{10}+N_{11}\ ,\ N_0=N_{00}+N_{01}\ e\ N=N_0+N_1 \end{aligned} \tag{6}$$

N_{ab} é a quantidade de documentos tal que e_t=a e e_c=b, N_t é o número de documentos que contêm t e N é o número total de documentos da amostra. A IM é máxima quando a presença e a ausência do token indica perfeitamente se o documento pertence ou não a uma classe. Os k atributos que possuírem os maiores escores de IM serão selecionados.

O segundo critério é o teste do qui-quadrado (χ^2) para testar a independência entre o evento da ocorrência de t e o de pertencimento à c. A estatística de teste é:

$$\chi^2(D,t,c)=\sum_{e_t\in(0,1)}\sum_{e_c\in(0,1)}\frac{(N_{e_te_c}-E_{e_te_c})^2}{E_{e_te_c}} \tag{7}$$

Na equação, D é conjunto de treinamento, N é a frequência observada em D e E é a frequência esperada. Assim, E_{11} é a frequência esperada de t e c ocorrerem em um documento, sob a hipótese de que a presença do termo e a classe do documento são independentes. Portanto, E_{11} será dada por:

$$E_{11} = N\, p(t)\, p(c) = N \frac{N_{11}+N_{10}}{N} \frac{N_{11}+N_{01}}{N} \quad (8)$$

A estatística de teste mede o desvio entre a frequência esperada E e a observada N. A hipótese nula é de independência entre os eventos e os valores críticos são dados pela distribuição qui-quadrada com m-1 graus de liberdade, sendo m o número de classes.

Manning et al. advertem que usar o teste por si só como seletor de atributos é inadequado, porque a correção de Yates necessária para um grau de liberdade dificulta a obtenção de significância estatística e pelo fato de o nível de significância usual de 5% ser irrelevante em relação à quantidade total de tokens normalmente possuída pelos textos (50 entre 1.000 tokens não farão diferença no resultado da classificação). Portanto, os autores recomendam que os atributos sejam selecionados por sua importância relativa, medida pela estatística de teste. Valores maiores indicarão maior dependência entre as ocorrências do termo e da classe, ou seja, maior importância.

Outro critério consiste em selecionar os tokens mais frequentes em cada classe. A frequência pode ser tanto medida pelo número de documentos da classe c que contêm t (frequência de documentos) ou a quantidade de ocorrências de t nos documentos de c (frequência de coleção).

Os três critérios descritos se enquadram nos métodos de seleção de atributos do tipo ***filter***, que analisam a relação das variáveis independentes com a classe e independem de algoritmo específico de AM para a sua execução. Em geral, os métodos *filter* são mais simples e rápidos que os do tipo ***wrapper***, os quais avaliam subconjuntos de atributos com base no desempenho de um algoritmo. Os métodos ***embedded*** representam um meio-termo entre os dois anteriores: os subconjuntos de atributos são avaliados e ajustados a cada iteração do treinamento do algoritmo de AM. A regressão Lasso é um exemplo de método *embedded*.

3.2. Aprendizado de máquina para classificação de documentos e extração de informações

Entre as diversas definições existentes de AM, uma versão bastante aderente à metodologia adotada para a resolução dos problemas propostos é a de Masini et al. [38], segundo a qual AM é a combinação de algoritmos automatizados e métodos estatísticos com o objetivo de encontrar padrões em conjuntos abundantes de dados.

Segundo Witten et al. [39], as ferramentas de AM podem ser divididas em três grupos:

i. Técnicas clássicas, que inferem diretamente de atributos selecionados ou construídos pelo usuário. Alguns algoritmos pertencentes a essa categoria são a regressão linear, o Naïve Bayes e as árvores de decisão.
ii. Técnicas de aprendizado de representações (*representation learning*), as quais transformam atributos em representações intermediárias antes de realizar as previsões. Um exemplo de representação intermediária são os componentes principais.
iii. Técnicas de aprendizado profundo (*deep learning*), que usam o aprendizado de representações para criar atributos mais complexos por meio de múltiplas camadas de representação.

A literatura também divide os algoritmos de AM em:

- Supervisionados: cuja execução depende da existência de uma variável dependente, comumente chamada de classe.
- Não supervisionados: analisam somente as associações e os padrões existentes entre os atributos, segundo Hastie et al. [40].
- Semissupervisionados: procuram alavancar o uso de métodos supervisionados em dados sem rótulos (i.e., sem anotações das classes aos quais pertencem), a partir de poucos dados rotulados.

A literatura de aprendizado estatístico de máquina (p. ex., [39] e [40]) coloca ênfase em algoritmos supervisionados e os indica para resolver problemas

de classificação de textos e extração de informações (vide [37], [1] e outros numerosos artigos).

Os algoritmos supervisionados testados na presente dissertação dividem-se em dois grupos:

i. Algoritmos ditos clássicos, os quais se baseiam nas técnicas clássicas de AM descritas anteriormente.
ii. Redes neurais profundas.

3.2.1. Conceitos básicos de aprendizado de máquina

Os problemas de aprendizado supervisionado geralmente possuem como objetivo construir modelos preditivos a partir de uma amostra de dados obtidos de uma população. A amostra é composta de **observações** de variáveis de entrada X, também chamadas de **preditores**, **atributos** ou **variáveis independentes**. A amostra também possui valores de uma variável Y cujos valores se deseja predizer no modelo final, denominada **variável de saída, variável dependente** ou **resposta**. Quando Y é uma variável categórica, ela também é chamada de **classe**, **categoria** ou **rótulo** (*label)*.

A modelagem pretende estabelecer uma função que relaciona X e Y após alimentar o algoritmo de AM por repetidas vezes os valores das duas variáveis nas observações disponíveis. Portanto, um modelo é a representação da função produzida pelo algoritmo de AM (ou método estatístico de aprendizagem), por meio da estimação de seus **coeficientes** ou **parâmetros**. Os algoritmos da AM possuem em geral um ou mais **hiperparâmetros** – parâmetros do modelo que precisam ser fornecidos pelo usuário antes de executar o algoritmo e permanecem fixos durante o processo de aprendizagem. Os algoritmos produzem resultados variados em função dos valores dos hiperparâmetros.

Geralmente o algoritmo de AM busca resolver um problema de **otimização**, que consiste em maximizar ou minimizar o valor de uma **função objetivo**, calculada em função das previsões por ele realizadas e os valores reais das respostas. A forma de função objetivo mais comum nas bibliotecas disponíveis em linguagem Python, usada nos experimentos da dissertação, é a **função perda** (*loss*

function), que deve ser minimizada. Ao final de cada iteração, o algoritmo prevê os valores da variável dependente, calcula o valor da função objetivo e automaticamente muda seus parâmetros, segundo regras e heurísticas.

O desempenho de um modelo é aferido por meio de uma ou mais **métricas**, também calculadas em função dos valores previstos e reais da variável dependente. As métricas servem para selecionar o melhor modelo gerado por um algoritmo e podem ser comparadas às métricas de modelos produzidos por outros algoritmos. Geralmente, o modelo de melhor métrica é o que será colocado em produção, mas é necessário levar em conta outros fatores práticos, como o tempo de execução do modelo para realizar novas previsões. É aconselhável que a métrica e a função objetivo sejam diferentes, para evitar resultados indesejáveis[10].

Para que um modelo seja útil, é preciso que ele consiga realizar previsões em dados novos ou futuros, sobre os quais não há informação sobre as respostas. No entanto, para que seja viável medir o desempenho de um modelo, é preciso dispor de observações com valores conhecidos das variáveis de saída que formarão um **conjunto de teste**, no qual serão calculadas métricas. O algoritmo será alimentado com as observações reunidas em um **conjunto de treinamento**, para gerar (ou treinar) um modelo. Os conjuntos de treinamento e teste devem ser disjuntos, pois as observações de teste não devem ser submetidas ao algoritmo durante o processo de aprendizagem, sob pena de os valores das métricas calculados no conjunto de teste superestimarem os valores esperados delas para uma nova observação. A capacidade de um modelo treinado de desempenhar bem no conjunto de teste ou em outros dados novos é denominada **generalização**. O objetivo final do AM é produzir um modelo que solucione o problema com boa generalização.

Uma condição para se obter boa generalização é o gerenciamento satisfatório do compromisso entre **viés** e **variância** (*bias-variance trade-off*). Seja $\hat{f}$ a função estimada por um algoritmo de AM e x_{0i}, uma observação do conjunto de teste com resposta y_{0i}, então o erro médio quadrático (MSE) do conjunto de

10 Por exemplo, se a métrica e a função objetivo forem a revocação (*recall*) de uma determinada classe, o modelo selecionado tende a ser o que prevê que todas as observações serão dessa categoria, pois a revocação será 100% nessa hipótese. No entanto, o modelo é obviamente inservível.

teste (uma métrica de uma variável de saída contínua) é definido pelo valor esperado a seguir.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}[y_{0i} - \hat{f}(x_{0i})]^2 \quad (9)$$

O valor esperado do MSE do conjunto de teste pode ser decomposto em três fatores: a variância de $\hat{f}(x_0)$, o quadrado de seu viés e a variância do erro irredutível ou intrínseco ε_0, ou seja:

$$E[y_0 - \hat{f}(x_0)]^2 = Var[\hat{f}(x_0)] + viés(\hat{f}(x_0))^2 + Var[\epsilon_0] \quad (10)$$

Pela fórmula observa-se que o erro mínimo é obtido quando a variância e o viés são baixos, pois todos os termos da decomposição mostrada são não negativos. A variância de um modelo é o grau de alteração que ele sofre ao ser treinado com um conjunto de treinamento diferente. O viés é o erro que surge quando se estabelece de uma forma incorreta ou demasiado simples a relação entre X e Y. Nesse caso, o modelo erra sistematicamente os valores de Y no conjunto de treinamento por mais observações que sejam agregadas a ele. Modelos dotados de mais parâmetros são mais flexíveis e tendem a diminuir o viés, porém geram maior variância. Portanto, a estratégia bem-sucedida consiste em aumentar a flexibilidade do modelo para reduzir o viés enquanto a variância é mantida em nível baixo e, assim, o MSE esperado do conjunto de teste cairá. Por outro lado, modelos flexíveis demais são altamente aderentes ao conjunto de treinamento e, em geral, possuem alta variância. Por conseguinte, geram um MSE do conjunto de teste mais elevado. Esse é o fenômeno do **sobreajuste** (*overfitting*) que a boa prática recomenda evitar, por ser o oposto da generalização. Uma técnica muito utilizada opara esse objetivo é a **regularização**, que consiste em aplicar uma penalização na função objetivo proporcional ao número e ao valor dos parâmetros. Outra técnica aplicável é *early stopping*, utilizado em redes neurais artificiais. Ambas procuram deslocar o ponto ótimo da posição original para que o processo de otimização se encerre antes de ocasionar sobreajuste.

Uma solução para o *trade-off* seria determinar o ponto ótimo onde o viés, representado pelo erro de predição no conjunto de treinamento, é mínimo e imediatamente "antes" (imaginando uma progressão de treinamentos ou de

aumento do grau de flexibilidade dos modelos) do início do aumento da variância, denotada pelo erro de predição no conjunto de teste. Entretanto, nem sempre o engenheiro dispõe de um fluxo de observações novas para formar esse conjunto. O método de **validação cruzada** oferece uma solução para a escassez, ao utilizar um **conjunto de validação** (*validation set* ou *hold-out set*) para estimar o desempenho do modelo no conjunto de teste. O conjunto de validação é extraído aleatoriamente do conjunto de treinamento e, caso a quantidade de observações disponíveis for limitada, separa-se de 15 a 30% delas para validação. A parcela restante do conjunto de treinamento mantém o nome original.

Na modalidade de validação cruzada chamada ***k-fold***, o conjunto de treinamento é dividido em k partes com o mesmo número de observações. O mesmo algoritmo será executado k vezes, sendo que em cada divisão (*fold*) do conjunto de treinamento, uma das parcelas será o conjunto de validação e as k-1 partes restantes formarão o novo conjunto de treinamento. Dessa forma, serão gerados k modelos treinados com a mesma configuração de hiperparâmetros, sobre conjuntos diferentes de treinamento e validação. Geralmente empregam-se valores k=5 ou k=10 para realizar validação cruzada, por oferecerem uma relação satisfatória entre viés e variância para estimar o erro de predição para o conjunto de teste [40].

O procedimento de validação cruzada permite testar e selecionar os hiperparâmetros que integrarão um modelo com maior grau de certeza quanto à sua capacidade de generalização. Isto porque a validação cruzada é menos propensa a sobrestimar o erro do conjunto de teste (ou **erro de generalização**) que um conjunto de validação fixo, além de sua estimativa possuir menor variância [41]. O erro geral da validação cruzada *k-fold* será a média dos erros obtidos nas k divisões (*folds*). Escolhe-se o conjunto de hiperparâmetros com o qual se obtém o menor erro geral (ou o melhor valor de outra métrica) da validação cruzada.

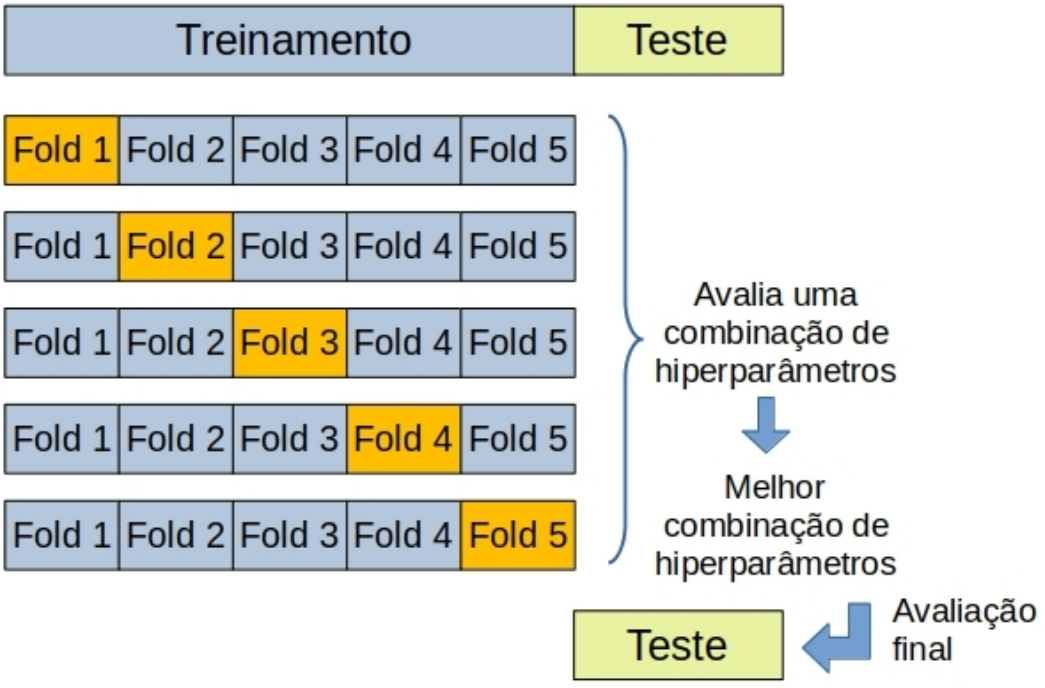
Se o número disponível de observações for suficiente, gera-se um terceiro conjunto aleatoriamente a partir do conjunto de treinamento, que é comumente chamado de conjunto de teste apesar de não ser formado por observações novas ou desconhecidas. Esse conjunto servirá para medir os desempenhos finais dos

modelos treinados e com todos os seus hiperparâmetros determinados, permitindo selecionar o melhor modelo, portanto.

A literatura divide-se quanto aos conjuntos requeridos pelo método da validação cruzada. Autores como James et al. [41] colocam ser suficiente usar dois conjuntos de observações (treinamento e validação) enquanto outros, como Aggarwal [42], defendem a importância do terceiro conjunto (teste) por não ser "contaminado" pelo conhecimento embutido nos conjuntos de treinamento e validação, nem estar sujeito a sobreajuste já que não é exposto aos algoritmos durante os treinamentos. Goodfellow et al. [43] afirmam que erro calculado sobre o conjunto de validação, por ser usado no treinamento, subestima o erro de generalização. Para o caso de três conjuntos, a divisão das observações é efetuada em duas etapas: primeiro separa-se o conjunto de teste e depois é realizada a validação cruzada ou a separação de um conjunto de validação fixo.

A Figura 1 mostra esquematicamente a validação cruzada *k-fold* com três conjuntos e k=5. Em cada execução, o *fold* do conjunto de validação está indicado com a cor laranja e o do conjunto de treinamento, com a cor azul.

Figura 1 - Validação cruzada k-fold com k=5 e conjunto de teste



Fonte: [35]

3.2.2. Definições dos problemas de classificação de documentos e extração de informações

Aggarwal [42] define formalmente o problema de classificação de documentos da seguinte forma. O conjunto de treinamento com n documentos é representado por uma matriz documento-termo $M_{n \times p}$. A sua i-ézima linha apresenta um vetor de atributos x_i e o documento i correspondente possui uma classe rotulada $y_i \in \{c_1,\ldots, c_K\} = C$, sendo C o conjunto de valores possíveis das classes. Um algoritmo de AM é treinado com M e o vetor de rótulos $(y_1,\ldots, y_n)^T$ para gerar um modelo δ. Modelos de classificação recebem os atributos x_j de um documento j do conjunto de teste, cuja classe real é Y_j, e estimam as probabilidades de Y_j assumir o valor de cada uma das K classes de C. Portanto, a classe $\hat{y}_j$ inferida será a maior probabilidade condicional calculada por δ, isto é:

$$\hat{y}_j = \underset{c_k \in C}{argmax}\, \hat{P}(Y_j = c_k \mid x_j), \text{ onde } \hat{P}(Y_j = c_i \mid x_j) = \delta(x_j), \forall c_i \in C \tag{11}$$

Em problemas de classificação de documentos em múltiplas categorias, é comum adotar, como função perda, a entropia cruzada categórica (*categorical cross-entropy*). O valor da perda L_i, para um documento i do conjunto de treinamento, é dada por:

$$L_i = -\sum_{k=1}^{K} y_k^i \cdot \log \hat{P}_k^i \tag{12}$$

Na fórmula, o algoritmo em treinamento estima as probabilidades $\hat{P}_k^i$ – condicionais, como em (11) – de a classe observada y_i assumir o valor c_k de cada uma das K classes. y^i_k é uma variável binária que assume o valor um, se y_i for igual a c_k, e zero, caso contrário. A entropia cruzada categórica penaliza mais fortemente predições muito confiantes (alta probabilidade estimada) que estão erradas e predições que acertam com pouca confiança.

O treinamento buscará, então, minimizar a função objetivo F(T) dada pelo somatório dos valores da função perda anterior, para todos os documentos do conjunto de treinamento T:

$$F(T) = -\sum_{i \in T} \sum_{k=1}^{K} y_k^i \cdot \log \hat{P}_k^i \tag{13}$$

Problemas de extração de informações modelados sob a forma de perguntas e respostas (*question answering*) também utilizam em geral a entropia cruzada categórica, para calcular a função perda. Nesses problemas, o algoritmo de AM estima as probabilidades de cada token i de uma dada passagem de texto (contexto) ser os tokens de início $(\hat{P}^i_{início})$ e fim $(\hat{P}^i_{fim})$ da resposta à pergunta analisada. A função perda para o token i pode ser definida como a média entre as entropias cruzadas referentes às duas probabilidades anteriores (14). $y^i_{início}$ e y^i_{fim} serão igual a um se i for o token de início e fim da resposta rotulada, respectivamente, e zero, caso contrário.

$$L_i = -\frac{1}{2}(y^i_{início} \cdot \log \hat{P}^i_{início} + y^i_{fim} \cdot \log \hat{P}^i_{fim}) \tag{14}$$

A perda relativa ao contexto pode ser calculada pela média dos valores que a função perda assume para os tokens constituintes. O algoritmo procurará minimizar a função objetivo representada pela média dos valores da função perda, relativos a todos os contextos do conjunto de treinamento.

Para responder a uma pergunta proposta, seleciona-se o intervalo (*span*) do contexto analisado que apresenta a maior probabilidade, calculada como segue. O algoritmo fornece como saída escores não normalizados $E^i_{início}$ e E^i_{fim} para o token i, os quais são processados pela função softmax (equação (63)), retornando $\hat{P}^i_{início}$ e $\hat{P}^i_{fim}$. A probabilidade de um intervalo situado entre os tokens i e j (j ≥ i) ser a resposta rotulada é o valor da função softmax aplicada sobre $E^i_{início} + E^j_{fim}$. Se a probabilidade máxima for obtida com i = j = 0 (posição do token [CLS] sem significado que indica o início do contexto), o algoritmo inferirá que o contexto não consegue responder à pergunta.

3.2.3. Métricas de desempenho

As métricas de desempenho usuais de problemas de classificação são o *recall* (revocação), a precisão e a acurácia. As duas primeiras métricas

determinam a fração recuperada de documentos de uma classe k e a proporção que esses documentos representam entre todos os documentos recuperados, respectivamente, sendo calculadas de acordo com (15). A acurácia define a proporção de classificações corretas retornadas por um modelo e, portanto, é mais comum determiná-la de forma global conforme (16), embora seja também possível calculá-la para uma dada classe.

$$Recall(k) = \frac{TP_k}{TP_k + FN_k}, Precisão(k) = \frac{TP_k}{TP_k + FP_k} \tag{15}$$

$$Acurácia = \frac{1}{N}\sum_{i=1}^{N} I(\hat{y}_i = y_i), \text{ onde } I(\hat{y}_i = y_i) = \begin{cases} 1, \text{ se } \hat{y}_i = y_i \\ 0, \text{ caso contrário} \end{cases} \tag{16}$$

Nas fórmulas, TP_k, FN_k e FP_k são as quantidades de verdadeiros positivos, falsos negativos e falsos positivos recuperados e pertencentes à classe k. N é o número total de observações, enquanto $\hat{y}_i$ e y_i são as classes predita e observada da observação i, respectivamente.

Para problemas em que é importante considerar o compromisso (*trade-off*) entre o *recall* e a precisão, utiliza-se a família de métricas F-beta – média harmônica que atribui peso β ao *recall*. F-beta é dada por:

$$F_\beta(k) = (1+\beta^2)\frac{Precisão(k)\cdot Recall(k)}{\beta^2 \cdot Precisão(k) + Recall(k)} \tag{17}$$

O desempenho de um modelo de classificação pode ser aferido pela macromédia ou micromédia dos valores de uma métrica, calculados para todas ou algumas classes do problema. A macromédia considera todas as classes como igualmente importantes, portanto é definida como a média aritmética entre os valores da métrica em questão. A micromédia leva em conta a contribuição de cada classe conforme o número de observações pertencentes a ela. Assim, o cálculo da micromédia envolve as contagens dos resultados das classificações (falsos positivos, falsos negativos etc.) que compõem o cálculo da métrica para cada classe analisada. No caso de F2 (i.e., β=2), a sua micromédia microF2 relativa a duas classes a e b é calculada a partir das micromédias da precisão e *recall* das classes. A fórmula de microF2 é dada por (18).

$$MicroF\,2_{a,b} = \frac{5 \cdot Micromédia(precisão) \cdot Micromédia(recall)}{4\,Micromédia(precisão) + Micromédia(recall)}$$
$$MicroP = Micromédia(precisão) = \frac{TP_a + TP_b}{TP_a + TP_b + FP_a + FP_b} \quad (18)$$
$$MicroR = Micromédia(recall) = \frac{TP_a + TP_b}{TP_a + TP_b + FN_a + FN_b}$$

No tocante à avaliação de extração de informações, a métrica **F1** é utilizada para aferir a performance de algoritmos estado da arte em resolver problemas de compreensão de texto escrito, modelados sob a forma de respostas encontradas no texto a perguntas previamente formuladas. F1 compara os tokens da resposta predita por um algoritmo de AM com os da resposta rotulada. F1 é definida por:

$$F1 = \frac{2 \cdot precisão \cdot recall}{precisão + recall}, \text{ onde } precisão = \frac{TP}{TP + FP} \text{ e } recall = \frac{TP}{TP + FN}$$
$TP =$ nº de tokens comuns entre as respostas correta e predita (19)
$FP =$ nº de tokens presentes na resposta predita e ausentes na correta
$FN =$ nº de tokens presentes na resposta correta e ausentes na predita

Outra métrica empregada é a **EM** (*Exact Match*), que mede a proporção de respostas preditas exatamente iguais às corretas.

3.2.4. Algoritmos clássicos de aprendizado de máquina

Problemas de classificação de texto podem ser resolvidos com uma gama razoavelmente extensa de algoritmos clássicos de AM, que pode ser consultada nas revisões disponíveis na literatura (p.ex, [44] e [45]).

3.2.4.1. Regressão Logística

A regressão logística é a variante da regressão linear aplicável a problemas de classificação, por modelar a probabilidade de ocorrência dos valores de uma variável dependente categórica. A probabilidade de a variável resposta Y pertencer à classe k, dentro de um universo de K classes possíveis, dado que o conjunto X

de p atributos assume um vetor de valores observados $x = X_1, X_2, \ldots, X_p$, é calculada pela função logística a seguir:

$$p(Y=k|X=x) = \frac{\exp(\beta_{0k} + x^T \beta_k)}{\sum_{i=1}^{K} \exp(\beta_{0i} + x^T \beta_i)} \tag{20}$$

Na equação anterior, β é uma matriz de coeficientes de dimensões p x K e β_k é um vetor da coluna k dessa matriz. β_0 é o vetor de interceptos da regressão.

Quando K ≥ 2, o método é denominado **regressão logística multinomial**. Naturalmente, esse método prevê que (20) deve ser aplicada para estimar a probabilidade de cada classe.

Outra forma de realizar uma classificação em mais de duas categorias é executar K repetições do modelo binomial que divide as observações em uma classe-alvo e o conjunto formado pelo restante das classes. Esse método é chamado *one-versus-rest* ou ovr. Tanto na forma multinomial como na ovr, o método atribui a observação à classe que apresentar a maior probabilidade prevista.

Os coeficientes lineares ou parâmetros referentes à classe k, denominados $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$, são estimados por máxima verossimilhança e de forma algorítmica, em que cada iteração resolve um problema de mínimos quadrados iterativamente ponderados (IRLS). Segundo Hastie et al. [40], o algoritmo geralmente converge devido à função de log verossimilhança log L(β_0, β) ser côncava. Os coeficientes estimados são dados por:

$$\hat{\beta}, \hat{\beta}_0 = \underset{\beta_0, \beta}{argmax}\, logL(\beta_0, \beta)$$

$$\hat{\beta}, \hat{\beta}_0 = \underset{\beta_0, \beta}{argmax} \{ \frac{1}{N} \sum_{i=1}^{N} [\sum_{j=1}^{K} y_{ik} (\beta_{0j} + x_i^T \beta_j) - \log (\sum_{j=1}^{K} \exp (\beta_{0j} + x_i^T \beta_j))] \} \tag{21}$$

Na equação anterior, y_{ik} é uma variável indicativa que assume os valores 1 ou 0, a depender se a variável resposta Y da observação i indicar que esta pertence à classe k ou a qualquer outra classe respectivamente.

A fim de evitar o sobreajuste, a regressão logística pode ser regularizada por meio das penalizações L_1 ou L_2. Esses casos são denominados regressões **Lasso**

[46] e **Ridge** [47] em suas formas logísticas, respectivamente. Ambas incluem um termo no problema anterior de maximização da log verossimilhança para diminuir o valor dos coeficientes em módulo, ou seja, em direção a zero. A versão L_1 é mostrada a seguir. O hiperparâmetro $\lambda \geq 0$ controla o grau de encolhimento dos coeficientes da regressão linear, com exceção do intercepto β_0.

$$\hat{\beta}, \hat{\beta}_0 = \underset{\beta_0, \beta}{argmax} \{ logL(\beta_0, \beta) - \lambda \sum_{i=1}^{K} \sum_{j=1}^{p} |\beta_{ji}| \} \tag{22}$$

Os coeficientes da regressão penalizada L_2 são estimados como segue:

$$\hat{\beta}, \hat{\beta}_0 = \underset{\beta_0, \beta}{argmax} \{ logL(\beta_0, \beta) - \lambda \sum_{i=1}^{K} \sum_{j=1}^{p} \beta_{ji}^2 \} \tag{23}$$

Enquanto a regressão Lasso zera alguns coeficientes, a Ridge não os encolhe até o valor nulo, a não ser que $\underline{\lambda}$ seja infinito. Devido a essa propriedade, a Lasso também é utilizada para selecionar atributos que serão aproveitados em outros algoritmos.

Em 2005, Zou e Hastie [48] propuseram a regularização **Elastic net**, que combina ambas as versões L_1 e L_2 em um único problema de otimização, expresso a seguir:

$$\hat{\beta}, \hat{\beta}_0 = \underset{\beta_0, \beta}{argmax} [logL(\beta_0, \beta) + \lambda P_\alpha(\beta)] \tag{24}$$

P_α é a penalização Elastic net, dada por:

$$P_\alpha(\beta) = (1-\alpha)\frac{1}{2}\|\beta\|_{L_2}^2 + \alpha\|\beta\|_{L_1} = \sum_{i=1}^{K} \sum_{j=1}^{p} [\frac{1}{2}(1-\alpha)\beta_{ji}{}^2 + \alpha|\beta_{ji}|] \tag{25}$$

A penalidade Elastic net representa um meio-termo entre a regularização da regressão Ridge ($\alpha = 0$) e a da Lasso ($\alpha = 1$). A Elastic net é útil quando o número de atributos é muito maior que o de observações ou na situação em que as variáveis independentes são muito correlacionadas [49]. Neste último caso, a Elastic net elimina a tendência do Lasso de escolher sem critério definido uma variável qualquer de um grupo de variáveis correlacionadas.

3.2.4.2. Naïve Bayes Multinomial

O classificador Naïve Bayes multinomial (MNB) assume a premissa "ingênua" de que, para uma classe k, os atributos X_i são independentes entre si. Nessa hipótese, a probabilidade de a variável resposta Y pertencer à classe k, dado o vetor x de atributos observados, é:

$$p(y_k|X=x)=\frac{1}{Z}p(y_k)\prod_{i=1}^{p}p(x_i|y_k), \quad \text{onde } Z=\prod_{i=1}^{p}p(x_i) \text{ e } y_k=\begin{cases}1, \text{ se } Y=k\\0, \text{ se } Y\neq k\end{cases} \tag{26}$$

É usual que a premissa não se aplique a problemas de PLN, visto que a ocorrência de uma palavra normalmente depende da presença da(s) anterior(es). No entanto, o classificador costuma oferecer resultados satisfatórios porque costuma inferir um valor de probabilidade muito maior para a categoria selecionada do que para as demais, apesar de as estimativas divergirem consideravelmente das probabilidades reais. Segundo Manning et al. [37], o MNB consegue bom desempenho quando há vários atributos igualmente importantes que contribuem para a decisão de classificação, além de ser robusto à presença de atributos ruidosos e a mudanças de conceito (*concept drifts*). Um exemplo desse último fenômeno é o uso de documentos de períodos temporais distintos entre os quais os conceitos se alteraram, como o conceito subjacente de presidente que passou de Michel Temer para Jair Bolsonaro. Outra vantagem do classificador é a sua rápida execução, por conta de sua simplicidade.

A partir de (26), estabelece-se que a classe a ser escolhida pelo MNB é a que possui a maior probabilidade estimada. É conveniente aplicar logaritmo no lado direito de (26) para evitar erros de ponto flutuante devido ao produto de várias probabilidades. Assim:

$$k(d_i)=\underset{k\leq K}{argmax}\,\hat{p}(Y=k|d_i)=\underset{k\leq K}{argmax}\,[\,p(Y=k)\prod_{j=1}^{p}\hat{p}(t_{ji}|Y=k)]$$
$$k(d_i)=\underset{k\leq K}{argmax}\,[\log\hat{p}(Y=k)+\sum_{j=1}^{p}\log\hat{p}(t_{ji}|Y=k)] \tag{27}$$

As equações anteriores utilizam terminologia adaptada ao PLN, onde d_i é o documento de ordem $\underline{i}$ cuja classe atribuída pelo MNB é $k(d_i)$ e t_{ji} é a palavra de ordem $\underline{j}$ integrante de d_i. A classe $k(d_i)$ dada por (27) é denominada classe *maximum a posteriori* (k_{map}). A segunda equação, de forma logarítmica, é mais utilizada para evitar multiplicações de decimais que conduzem a quantidades demasiadamente pequenas. As duas equações suprimiram o fator $\underline{Z}$ da equação (26) porque ele é constante para todas as classes.

Por máxima verossimilhança, a probabilidade *a priori* de o documento d_i pertencer à classe $\underline{k}$ e a probabilidade condicional do termo t_{ji} são estimadas como:

$$\hat{p}(Y=k)=\frac{n_k}{N} \quad \text{e} \quad \hat{p}(t_{ji}|Y=k)=\frac{f_{tk}}{\sum_{t'\in V} f_{t'k}} \tag{28}$$

Nas fórmulas, n_k é o número de documentos que de fato pertencem à classe $\underline{k}$, $\underline{N}$ é a quantidade total de documentos, f_{tk} é a frequência absoluta do termo t_{ji} nos documentos da classe $\underline{k}$, V é o conjunto global de termos (ou vocabulário) e $f_{t'k}$ é o número total de ocorrências de um termo $\underline{t}$' nos documentos da classe $\underline{k}$. Portanto, o somatório representa o total de ocorrências de t_{ji} em todos os documentos da classe $\underline{k}$. A estimativa da probabilidade condicional corresponde à frequência relativa de t_{ti} no conjunto de documentos da classe $\underline{k}$.

Para evitar que termos ausentes no conjunto de treinamento tenham probabilidade nula ao estimá-la com (28), utiliza-se a técnica da suavização de Laplace, que consiste em adicionar um termo (geralmente igual a um) a cada contagem de termos. Assim, a forma suavizada de (28) é:

$$\hat{p}(t_{ji}|Y=k)=\frac{f_{tk}+1}{B+\sum_{t'\in V} f_{t'k}} \tag{29}$$

Na equação anterior, B é o número total de termos do vocabulário V.

3.2.4.3. Naïve Bayes Complementar (*Complement Naïve Bayes* ou CNB)

O CNB foi proposto por Rennie et al. [50] em 2003 para lidar com as fraquezas do NB ao lidar com dados desbalanceados e aquelas advindas da premissa de independência entre os atributos. Os autores mostraram que o NB tende a preferir as classes mais prevalentes na amostra. A variante CNB mitiga esses efeitos ao estimar os parâmetros com todas as classes exceto a que está em análise (daí o termo complementar). Dessa forma, a regra de classificação do CNB com suavização de Laplace é:

$$k(d_i) = \underset{k \leq K}{argmax}\left[\log \hat{p}(Y=k) - \sum_{i=1}^{p} f_{ti} \log \frac{f_{t\tilde{k}}+1}{B+\sum_{t' \in V} f_{t'\tilde{k}}}\right] \tag{30}$$

Na fórmula, f_{ti} é a frequência absoluta do termo t no documento d_i e $\tilde{k}$ compreende todas as classes exceto a categoria k. O sinal negativo do segundo termo denota que o documento é mais facilmente atribuído à classe k quanto menor for a frequência relativa suavizada de t_{ji} no complemento de k.

O WCNB (*Weight-normalized CNB*) é uma variante do CNB que normaliza o vetor de pesos da frequência f_{ti}, isto é, o termo logarítmico suavizado de (30). Essa correção visa a corrigir os pesos que o NB atribui aos termos de forma enviesada, quando a premissa de independência é aplicada a dados dependentes entre si, como é o caso de textos. O vetor de pesos normalizado $\hat{w}_{kt}$ é:

$$\hat{w}_{kt} = \log \frac{\hat{\theta}_{kt}}{\sum_{t' em docs de k} |\log \hat{\theta}_{kt'}|}, \text{ onde } \hat{\theta}_{kt} = \hat{p}(t|Y=k) = \frac{f_{t\tilde{k}}+1}{B+\sum_{t' \in V} f_{t'\tilde{k}}} \tag{31}$$

Rennie et al. reportaram em [50] que o CNB obteve desempenho comparável a algoritmos sofisticados, como o SVM, e superior ao NB em tarefas de classificação de textos.

3.2.4.4. *K-Nearest Neighbors* (KNN)

O classificador KNN [51] utiliza os K pontos do conjunto de treinamento mais próximos da observação x a ser categorizada. Diferentes tipos de distâncias, como a Euclidiana e Manhattan, podem ser empregados para caracterizar a proximidade entre os vetores de observações. Esses K pontos formarão o conjunto N_0. Em seguida, estima-se a probabilidade condicional da classe k como sendo a fração de pontos de N_0 cujos valores de resposta são iguais a k. Assim:

$$\hat{p}(Y=k|X=x) = \frac{1}{K}\sum_{i\in N_0} y_i \tag{32}$$

Na equação anterior, y_i é a variável indicativa da variável resposta Y relativa à observação i ser igual ou não a k, como visto em (26). O KNN alocará a observação na classe cuja probabilidade condicional calculada por (32) for máxima.

O desempenho do algoritmo depende fundamentalmente do número estipulado de pontos mais próximos, isto é, do valor de K.

3.2.4.5. *Nearest Shrunken Centroids*

Proposto em 2002 por Tibshirani et al. [52], este algoritmo classifica a observação à classe cujo centroide é o mais próximo e, similarmente à regressão Lasso, aplica um procedimento de regularização que zera atributos irrelevantes para a classificação. A descrição a seguir utiliza a terminologia do artigo original. Sejam x_{ij} o valor do atributo i referente à observação j, n_k o número de observações de treinamento pertencentes à classe k e C_k o conjunto de índices das observações da classe k. A quantidade total de observações é n e o número total de classes é igual a K.

Para o atributo i, o centroide $\bar{x}_{ik}$ da classe k e o centroide geral do atributo $\bar{x}_i$ são calculados da seguinte forma:

$$\bar{x}_{ik} = \sum_{j \in C_k} \frac{x_{ij}}{n_k} \quad \text{e} \quad \bar{x}_i = \sum_{j=1}^{n} \frac{x_{ij}}{n_k} \tag{33}$$

A regularização explicada a seguir reduz (encolhe) $\bar{x}_{ik}$ na direção de $\bar{x}_i$, normalizando pelo desvio-padrão intraclasse agrupado (*pooled*) s_i para conferir maior peso aos atributos mais estáveis nas observações da mesma classe.

O primeiro passo do algoritmo é construir uma estatística de teste d_{ik} tal que:

$$d_{ik} = \frac{\bar{x}_{ik} - \bar{x}_i}{m_k (s_i + s_0)}, \text{ onde } s_i^2 = \frac{1}{n-K} \sum_k \sum_{j \in C_k} (x_{ij} - \bar{x}_{ik})^2 \quad \text{e} \quad m_k = \left(\frac{1}{n_k} + \frac{1}{n}\right)^{\frac{1}{2}} \tag{34}$$

Na fórmula, s_0 é uma constante positiva que evita possíveis valores de d_{ik} demasiado grandes, ocasionados por atributos com baixos valores. O algoritmo reduz d_{ik} em direção a zero, gerando a estatística reduzida d'_{ik} e o centroide de classe encolhido x'_{ik}, dados por:

$$\bar{x}'_{ik} = \bar{x}_i + m_k (s_i + s_0) d'_{ik} \tag{35}$$

$$d'_{ik} = sign(d_{ik})(|d_{ik}| - \Delta)_+, \text{ onde } t_+ = \begin{cases} t, \text{ para } t > 0 \\ 0, \text{ para } t \leq 0 \end{cases} \tag{36}$$

De acordo com (36), Δ é o hiperparâmetro de regularização com patamar suave (*soft thresholding*) que reduz d_{ik} a um valor resultante positivo, ou a zero se o resultado for negativo. Ao aumentar Δ, mais atributos serão zerados para a classificação. Importante observar que Δ pode assumir qualquer valor positivo, não estando limitado a um.

O algoritmo se destaca pela simplicidade e rapidez de execução.

3.2.4.6. Análises Discriminantes Linear e Quadrática (LDA e QDA)

Usando a mesma notação dos modelos MNB e CNB, o Teorema de Bayes estabelece que:

$$p_k(X) = p(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^{K} \pi_i f_i(x)} \tag{37}$$

Na fórmula, π_i é a probabilidade *a priori* de uma observação aleatória pertencer à classe $\underline{i}$ e $p_k(x)$ é a probabilidade *a posteriori* de uma observação, que assume o vetor $\underline{x}$ de atributos observados, ser da classe $\underline{k}$.

O algoritmo LDA assume que a função densidade de probabilidade $f_k(x)$ de uma observação da classe $\underline{k}$ é uma normal multivariada e que a matriz de variância e covariância Σ é comum a todas as classes. Ao substituir a expressão analítica da distribuição gaussiana em (37) e aplicando logaritmo em ambos os lados da equação, obtém-se:

$$\delta_k(x) = \log p_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k \tag{38}$$

O classificador LDA aplica em (38) os seguintes estimadores da probabilidade *a priori*, média e matriz de variância e covariância de cada classe:

$$\pi_k = \frac{n_k}{N} \quad , \quad \hat{\mu}_k = \frac{1}{n_k} \sum_{i\,tal\,que\,y_i = k} x_i \quad , \quad \hat{\Sigma} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{i\,tal\,que\,y_i = k} (x_i - \hat{\mu}_k)^2 \tag{39}$$

O classificador alocará a observação de teste na classe cujo valor obtido por (38) é o maior dentre os valores relativos às classes. A denominação linear é justificada pelo fato de $\delta_k(x)$ ser uma função linear de $\underline{x}$.

Segundo James et al. [41], a LDA oferece estimativas mais estáveis de parâmetros que os da regressão logística, nos casos em que as classes estão bem separadas. Além disso, a LDA é mais estável que a regressão quando se dispõe de poucas observações disponíveis. Por esses motivos, em problemas de classificação multiclasse a LDA costuma mais empregada que a regressão logística.

A QDA também trabalha sob a hipótese de observações distribuídas normalmente porém assume que cada classe possui a sua própria matriz de variância e covariância Σ_k. O classificador QDA $\delta_k(x)$ é expresso por:

$$\delta_k(x) = -\frac{1}{2} x^T \Sigma_k^{-1} x + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \log |\Sigma_k| + \log \pi_k \tag{40}$$

Devido ao número maior de parâmetros a serem estimados, a QDA é recomendada para conjuntos grandes de dados e oferece viés menor que a LDA, porém sua variância é maior.

A regularização para a LDA e a QDA é usualmente efetuada reduzindo-se a matriz de variâncias e covariâncias. Uma forma frequente na literatura é a análise de discriminantes regularizada (RDA), a qual estima Σ_k como um compromisso entre a matriz comum da LDA e as matrizes individualizadas da QDA:

$$\hat{\Sigma}_k = \alpha \hat{\Sigma}_k + (1-\alpha)\hat{\Sigma} \tag{41}$$

Para a QDA, a biblioteca Python Scikit-learn realiza a seguinte regularização:

$$\hat{\Sigma}_k = (1-\beta)\hat{s}^2{}_k + \beta p \cdot I \tag{42}$$

Na equação, p é o número total de atributos, I é a matriz identidade e $\hat{s}^2{}_k = S^2/(N-1)$, onde S é a matriz diagonal de valores singulares obtida pela decomposição SVD da matriz centrada x_k de observações da classe k, isto é, $x_k = USV^T$. A estimação por SVD evita o custoso cálculo direto do estimador amostral $\hat{\Sigma}_k$.

Para a LDA, a biblioteca oferece a possibilidade de reduzir a matriz comum Σ na direção de uma matriz diagonal de variâncias (tal qual descrito em [40]), ou seja:

$$\hat{\Sigma} = \gamma\hat{\Sigma} + (1-\gamma)\hat{\sigma}^2 I \tag{43}$$

3.2.4.7. Máquina de Vetores de Suporte ou *Support Vector Machine* (SVM)

Proposto por Cortes e Vapnik em 1995 [53], o algoritmo SVM é um dos mais empregados em problemas de classificação por conta de sua flexibilidade, custo computacional aceitável e capacidade de modelar de forma não linear. Essa última característica se deve ao uso do truque do kernel, explicado a seguir. Uma **função kernel** K possibilita calcular o produto escalar entre duas observações X_1 e X_2 em um espaço ampliado V (i.e., com mais dimensões que o original X), sem precisar efetuar a transformação das observações para V. Analiticamente, seja a função h, que mapeia observações de X para V sob a forma h: X→V. A função kernel K efetua, portanto, a seguinte transformação sobre X_1 e X_2:

$$K(X_1, X_2) = f(X_1 \cdot X_2) = h(X_1) \cdot h(X_2) \tag{44}$$

Portanto, a função kernel K é uma função do produto escalar $X_1 \cdot X_2$ que permite determinar o produto vetorial entre $h(X_1)$ e $h(X_2)$ em V, dispensando o cálculo da transformação de X_i para $h(X_i)$. A seleção de uma função kernel adequada é facilitada pelo teorema de Mercer, o qual afirma que toda função simétrica definida semipositiva é um kernel. Duas funções kernel bastante utilizadas no SVM são:

$$\text{Polinomial: } K(X_1, X_2) = (1 + X_1 \cdot X_2')^d = (1 + \sum_{i=1}^{p} x_{i1} x_{i2})^d \tag{45}$$

$$\text{Radial: } K(X_1, X_2) = \exp(-\gamma \|X_1 - X_2\|^2) = \exp[-\gamma \sum_{i=1}^{p} (x_{i1} - x_{i2})^2] \tag{46}$$

O classificador SVM de um vetor de observações x do conjunto de teste, dado um conjunto de treinamento composto por n observações (x_i = atributos, y_i = classes) será dado por:

$$f(x) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i h(x) \cdot h(x_i) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i K(x, x_i) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\beta}^T h(x) \tag{47}$$

No caso binário, a classe indicada por f(x) dependerá do sinal da função. O último termo de (47) mostra que o classificador é linear no espaço ampliado h(x), acessado pela função kernel K. Os coeficientes α_i são iguais a zero em todas as observações i, exceto nos chamados de vetores de suporte, que são as observações que definem o hiperplano classificador. Por essa razão, o SVM é robusto a *outliers* e funciona bem com amostras pequenas, pois poucos vetores de suporte na zona limítrofe (não influenciáveis por observações distantes, portanto) são suficientes para delimitar o hiperplano classificador. Os coeficientes α_i são estimados pelo seguinte problema de maximização, em sua forma dual:

$$\max_{\alpha_1, \ldots \alpha_n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \begin{cases} 0 \le \alpha_i \le C, \forall i = 1,2 \ldots n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{48}$$

O hiperparâmetro não negativo C regula o custo atribuído a classificações errôneas e controla o *trade-off* entre viés e variância. Valores menores de C levam

a um maior número de vetores de suporte e o classificador se ajusta menos aos dados, i.e., é menos rigoroso, resultando em maior viés e menor variância. Portanto, o inverso de C pode ser interpretado como uma medida de regularização.

Os coeficientes originais β são estimados pela equação:

$$\hat{\beta}=\sum_{i=1}^{N}\hat{\alpha}_i y_i x_i \tag{49}$$

O intercepto β_0 é calculado a partir de uma das condições de Karush-Kuhn-Tucker [54], explicitada a seguir e aplicável a (48). ε_i é uma variável de folga que indica a posição da observação i em relação ao hiperplano classificador e suas margens. Segundo Hastie et al. [40], é recomendado usar no cálculo apenas os vetores de suporte situados sobre as margens e, dessa forma, ε_i será igual a zero para todos eles. O intercepto será a média dos valores obtidos por (50).

$$\alpha_i[y_i(x_i^T\beta+\beta_0)-(1-\epsilon_i)]=0 \tag{50}$$

A forma primal equivalente a (48) é dada por (51) e corresponde a aplicar a regularização L_1, representada pelo segundo termo da soma. A regularização L_2 está mostrada em (52), sujeita às mesmas condições.

$$\min_{\beta_0,\beta}\sum_{j=1}^{p}\beta_j^2+C\sum_{i=1}^{n}\epsilon_i, \text{ sujeito a } y_i f(x_i)\geqslant 1-\epsilon_i, \forall i=1,\ldots,n; \text{ e } \epsilon_i\geqslant 0 \tag{51}$$

$$\min_{\beta_0,\beta}\sum_{j=1}^{p}\beta_j^2+C\sum_{i=1}^{n}\epsilon_i^2 \tag{52}$$

Hastie et al. [40] utilizam a nomenclatura SVM unicamente para métodos que empregam kernels não lineares enquanto o classificador de vetores de suporte (*Support Vector Classifier* ou SVC) representa o caso linear. Nesta dissertação, o termo SVM é aplicado nos dois casos.

3.2.4.8. Random Forest, *Bagging* e *Boosting*

A **árvore de decisão** ou método CART[11] é um algoritmo de AM que divide o espaço de preditores, isto é, todos os possíveis valores dos atributos X_1, X_2,…, X_n, em regiões não sobrepostas. Em um problema de classificação, a cada observação do conjunto de teste que cair em uma determinada região do espaço será atribuída a classe mais frequente (classe modal) dentre as das observações do conjunto de treinamento que estão na mesma região. A árvore é construída a partir de um nó inicial com todos os atributos e valores possíveis e segue o método de repartição binária recursiva (*recursive binary splitting*). O método prevê que cada nó será dividido em dois galhos, onde o preditor X_i e o ponto de corte $\underline{s}$ repartirão o espaço de preditores nas regiões $\{X \mid X_i < s\}$ e $\{X \mid X_i \geq s\}$. X_i e $\underline{s}$ devem ser escolhidos para proporcionar a máxima redução na **entropia** $\underline{D}$, definida como:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{53}$$

Na equação, $\hat{p}_{mk}$ é a proporção das observações do conjunto de treinamento que estão na m-ézima região e pertencem à k-ézima classe. Quando a entropia é próxima de zero (ela é sempre não negativa), as proporções $\hat{p}_{mk}$ estarão todas ao redor de zero ou um e, desse modo, diz-se que o m-ézimo nó é puro. A medida de pureza de um nó é dada pelo índice de Gini:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{54}$$

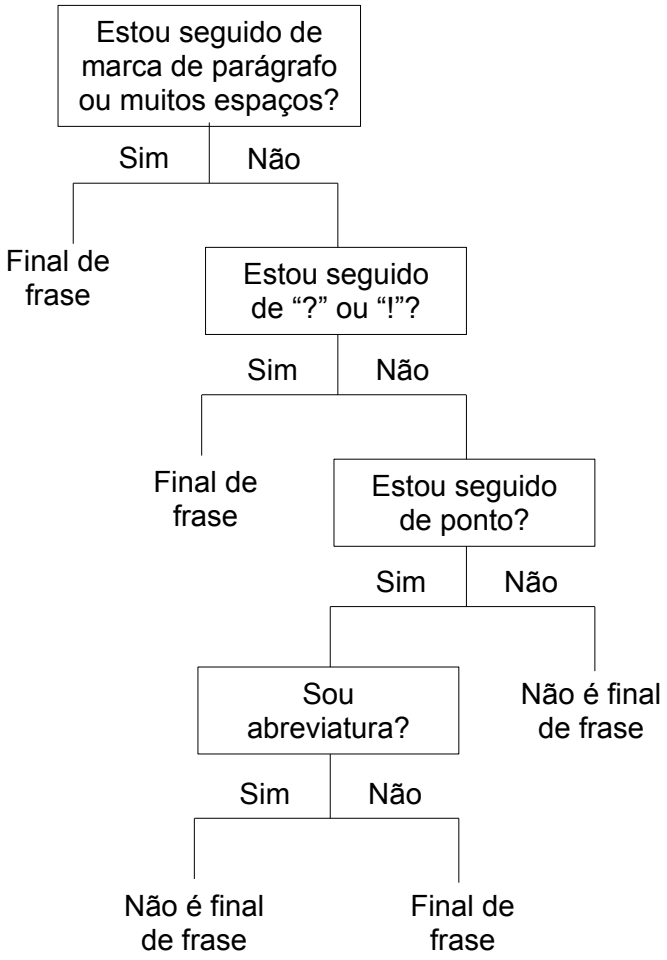
O índice de Gini também pode ser aplicado para avaliar se uma divisão é adequada ou não.

A Figura 2 mostra um exemplo de árvore de decisão com m=5 regiões, **nós terminais** ou **folhas** para avaliar se um caractere indica ou não o final de uma frase. Uma vantagem desse algoritmo é a fácil compreensão das regras de decisão estabelecidas nos nós e galhos. Chamam-se **nós internos** os que geram galhos. A **profundidade** ou **altura** de uma árvore é o número de nós contados ao percorrer

11 *Classification and Regression Trees*

o maior caminho entre o nó raiz e uma folha. A profundidade da árvore da figura é igual a dois porque o nó raiz é considerado zero.

Figura 2 - Árvore de decisão para classificação de caractere de fim de frase



Apesar de sua boa interpretabilidade, as árvores de decisão são menos acuradas que os métodos anteriores para classificação e regressão [42]. Para melhorar o desempenho, há três tipos de aperfeiçoamentos baseados na aprendizagem **Ensemble**, a ser abordada na seção 3.2.5.

O primeiro aperfeiçoamento é o uso da técnica ***bagging*** (*bootstrap aggregation*) [55], que consiste em extrair B amostras com reposição (*bootstrap*) do mesmo conjunto de treinamento, para treinar B modelos e calcular a média de suas previsões $f_i(x)$, isto é, $\hat{f}_{bag}(x)=\frac{1}{B}\sum_{i=1}^{B}\hat{f}_i(x)$. A partir de uma amostra, cada árvore é construída sem poda, ou seja, com todas as ramificações possíveis e,

portanto, seu viés de predição é o mínimo possível. A média dos resultados das árvores possui variância menor que a das previsões individuais. Dessa forma, o *bagging* produz um equilíbrio satisfatório entre viés e variância. Para um problema de classificação, a classe mais escolhida pelas previsões individuais será considerada como a prevista pelo modelo consolidado. Outro critério para categorizar é utilizar a média das probabilidades condicionais (estimadas pelas árvores) de cada classe, no nó em que a classificação for realizada. Pode-se testar um número grande de amostras (valores altos do hiperparâmetro B) para buscar o modelo com os melhores resultados.

O algoritmo ***Random Forest*** [56] representa um aperfeiçoamento do *bagging* ao descorrelacionar as árvores de decisão componentes. Efetua-se cada ramificação de uma dada árvore a partir de um subconjunto extraído aleatoriamente, em vez de considerar todos esses atributos. Geralmente, são selecionados aleatoriamente dentre p atributos. Esse procedimento descorrelaciona os resultados das árvores à medida que impede que os atributos mais relevantes figurem no topo de todas elas.

Segundo Freund e Schapire, que propuseram o algoritmo AdaBoost em 1996 [57], o ***boosting*** é um método geral que pode ser aplicado para melhorar a performance de qualquer algoritmo de AM. O *boosting* produz diferentes modelos de aprendizagem ao aplicar um mesmo modelo base a versões modificadas do conjunto de treinamento, durante um número determinado de iterações. As modificações consistem em aplicar pesos distintos às observações do conjunto de treinamento. O erro calculado serve para balizar o grau de aumento dos pesos das observações classificadas erradamente e a redução dos pesos das instâncias com categorias corretas. O *boosting* é um tipo de modelo aditivo, ou seja, o modelo final f(x), preditor da observação de teste x, é um somatório de M modelos (ocasionalmente chamados de aprendizes fracos ou *weak learners*) e pode ser explicitado pela seguinte forma geral:

$$f(x) = \sum_{i=1}^{M} \beta_i \, b(x, \gamma_i) \tag{55}$$

Na fórmula, b é um modelo de AM que pode ser uma árvore de decisão, cujo conjunto de parâmetros γ_i é intrínseco ao modelo, como os atributos e os

valores empregados nas divisões de ramos das árvores. β_i são coeficientes que atribuem pesos aos resultados dos aprendizes fracos. Enquanto outros algoritmos otimizam seus parâmetros em conjunto, o *boosting* atua em etapas (*forward stagewise additive modeling*), isto é, ele otimiza os parâmetros de uma árvore por vez e adiciona as árvores anteriores sem ajustar os coeficientes e parâmetros das últimas. Isto simplifica o processo de minimização da função aplicável a (55), a qual precisaria otimizar simultaneamente uma quantidade considerável de parâmetros β_i e γ_i de todas as árvores. A estimação em etapas é mais lenta e assim o modelo consegue controlar melhor sua evolução, evitando o sobreajuste mais eficientemente. O algoritmo *forward stagewise additive modeling* é descrito como segue.

1. Inicialize com $f_0(x) = 0$.
2. De m = 1 até M, repita:

   2.1. Calcule $(\beta_m, \gamma_m) = \underset{\beta, \gamma}{argmin} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta\, b(x_i, \gamma))$

   2.2. Atribua $f_m = f_{m-1}(x) + \beta_m\, b(x, \gamma_m)$

No passo 2.2 é possível utilizar um fator de encolhimento ε tal que $f_m = f_{m-1}(x) + \epsilon\, \beta_m\, b(x, \gamma_m)$ para diminuir ainda mais a velocidade do algoritmo e, assim, incrementar o desempenho. ε também é chamado de taxa de aprendizagem.

O modelo de árvores *boosted* $f_M(x)$ é dado pela soma simples das M árvores constituintes T com seus parâmetros θ_i:

$$f_M(x) = \sum_{i=1}^{M} T(x, \theta_i) \tag{56}$$

A cada passo (item 2.1.) do processo *forward stagewise* anterior deve-se resolver o seguinte problema de otimização:

$$\hat{\theta}_m = \underset{\theta_m}{argmin} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i, \theta_m)) \tag{57}$$

Efetua-se a otimização de forma numérica e o algoritmo empregado é o *gradient boosting* proposto por Jerome Friedman em 2001 [58]. O *gradient*

boosting é um arcabouço estatístico que permite minimizar, de forma geral, uma função perda de um modelo aditivo composto por aprendizes fracos. O artigo [58] explica detalhadamente o algoritmo aplicado a métodos de regressão e classificação e a diversos tipos de funções perda.

Um modelo de árvores *boosted* possui numerosos hiperparâmetros para serem regulados. Além da taxa de aprendizagem, número de árvores construídas e de níveis delas, pode-se ajustar a quantidade pré-selecionada de atributos[12] a serem usados em cada nível da árvore (como na *Random Forest*), a taxa de amostragem das observações de treino (similarmente ao *bagging*), a intensidade da regularização a ser aplicada sobre a complexidade da árvore e outros parâmetros. A amostragem de observações e atributos provê natureza estocástica ao algoritmo de otimização (*stochastic gradient boosting*).

3.2.5. *Ensembles*

A aprendizagem *Ensemble* (ou por comitês de máquinas) se baseia na premissa de que a combinação de resultados de múltiplos modelos apresenta possibilidades de obter melhor desempenho conjunto que os dos modelos individuais. Os principais aspectos que caracterizam e determinam a performance de um comitê são:

i. os modelos individuais a serem combinados; e
ii. o método de combinação dos resultados dos modelos individuais.

Introduzindo a notação matemática aplicável aos comitês, sejam $\underline{M}$ modelos individuais $h_1, \ldots, h_M$, a serem combinados para predizer as classes de $\underline{N}$ observações, dentre as $\underline{K}$ classes possíveis $(c_1, \ldots, c_K)$. Para uma observação $\underline{x}$, cada modelo classificador h_i produzirá uma saída $h_i^j(x)$ para a classe c_j. Portanto, h_i gerará o vetor $(h_i^1(x), \ldots, h_i^K(x))^T$. A saída $h_i^j(x)$ pode tomar duas formas:

12 Os atributos são também denominados "colunas" na biblioteca xgboost, usada nesta dissertação.

i. classes preditas (*crisp labels*): $h_i^j(x) = 1$, se a classe predita for c_j; e $h_i^j(x) = 0$, caso contrário; ou

ii. probabilidades preditas de classes: $h_i^j(x) \in [0,1]$ e estima $P(c_j \mid x)$.

Segundo Zhou [59], o comitê de votação é a forma mais popular e fundamental de combinação de modelos para saídas categóricas. A votação utiliza as classes preditas. A modalidade padrão de votação é a plural, que elege a classe predita pelo maior número de modelos individuais. A classe H(x) predita pelo *ensemble* de votação plural, para a observação $\underline{x}$, será:

$$H(x) = c_k, \text{ onde } k = \underset{j}{argmax} \sum_{i=1}^{M} h_i^j(x) \tag{58}$$

O cálculo dos votos pode ser ponderado para, p. ex., conferir maior peso aos classificadores mais fortes. Atribuindo-se o peso w_i ao modelo h_i, tem-se:

$$H(x) = c_k, \text{ onde } k = \underset{j}{argmax} \sum_{i=1}^{M} w_i \cdot h_i^j(x) \tag{59}$$

Embora pouco frequente na literatura estatística sobre *ensembles*, o termo *hard voting* é utilizado como sinônimo de votação plural em AM, em oposição à modalidade *soft voting*[13], a qual lida com probabilidades preditas de classes. Na votação *soft* simples, o *ensemble* estimará a probabilidade $H^j(x)$ da classe c_j como a média das probabilidades preditas pelos modelos individuais para c_j:

$$H^j(x) = \frac{1}{M} \sum_{i=1}^{M} h_i^j(x) \tag{60}$$

A votação *soft* ponderada pode empregar pesos específicos por modelo (w_i) ou por classe e modelo (w_i^j). No primeiro caso, tem-se:

$$H^j(x) = \frac{1}{M} \sum_{i=1}^{M} w_i \cdot h_i^j(x) \tag{61}$$

O *stacking* é uma modalidade de *ensemble* em que um novo modelo, denominado metamodelo ou modelo nível 1, é treinado com os vetores de saída $(h_i^1(x), \ldots, h_i^K(x))^T$ relativos a todas as observações, classes e modelos

13 O termo *soft voting*, por sua vez, é corriqueiro na literatura estatística.

constituintes, os quais são chamados de modelos nível 0 e são usualmente gerados por algoritmos diferentes. Para problemas de classificação, Ting e Witten [60] recomendam o uso das probabilidades preditas de todas as classes como atributos do nível 1, em vez das classes preditas, por constituírem medidas de confiança das predições.

O *StackingC*, idealizado por Seewald [61], usa apenas as probabilidades preditas da classe c_j para treinar um metamodelo de classificação binária relativa a c_j. Assim, para $\underline{K}$ classes são treinados $\underline{K}$ metamodelos. A classe predita do nível 1 para uma observação é dada pelo metamodelo que retornar a maior probabilidade predita. Seewald defende que empregar as probabilidades preditas para todas as classes sujeita o metamodelo à "maldição da dimensionalidade", i.e., treina-o com uma quantidade excessiva de atributos. No artigo original, o *StackingC* proporcionou melhor desempenho em problemas de classificação multiclasse do que o método de *stacking* proposto por Ting e Witten.

3.2.6. Redes neurais profundas

As redes neurais artificiais, ou simplesmente **redes neurais**, são estruturas computacionais inspiradas no cérebro humano e dispostas em níveis ou **camadas** que processam sequencialmente os dados relativos às observações e realizam previsões ou inferências por generalização, tal qual os algoritmos clássicos de AM. As camadas são formadas por **unidades simples de processamento** ou **neurônios**, que conseguem armazenar conhecimento adquirido por experiência de processamentos anteriores, por meio de **pesos sinápticos** atuantes sobre os dados que fluem entre um neurônio e outro. Do mesmo modo que os algoritmos clássicos, as redes neurais capturam conhecimento por intermédio de um processo de aprendizagem sobre o conjunto de treinamento, em forma de algoritmo. Os neurônios e pesos sinápticos serão também aqui referidos como **unidades** e **pesos**, simples e respectivamente.

Apesar de autores como Hastie et al. [40] classificarem as redes neurais como simples "modelos estatísticos não lineares", elas diferem dos algoritmos clássicos em alguns aspectos [62]:

i. Capacidade de extrair de forma automatizada os atributos necessários para uma dada tarefa por meio do aprendizado de representação.
ii. Capacidade de lidar naturalmente com informação contextual, já que cada unidade é potencialmente afetada pelas atividades das outras na mesma rede.
iii. Tolerância a falhas pequenas ou graduais, pois a informação é distribuída pela rede e, portanto, seu desempenho se degrada aos poucos em vez de diminuir abruptamente.
iv. Uniformidade de design e análise, ou seja, redes neurais distintas compartilham teorias, algoritmos de aprendizagem e são modulares. Todas apresentam neurônios, embora possam ser de diferentes tipos.
v. Processamento de informações em paralelo, executado pelos neurônios.
vi. Conexão com a neurobiologia, tal que esta inspira novas soluções de engenharia de redes e vice-versa.

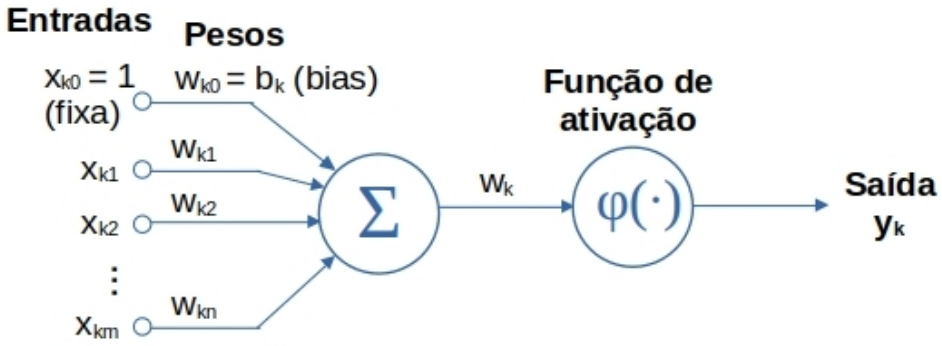
Para a resolução de problemas de classificação em PLN são usadas **redes neurais profundas**, isto é, que possuem mais de uma **camada escondida**. Toda rede neural possui uma entrada para receber os dados e emite resultados por meio da camada de saída. A entrada recebe palavras codificadas em vetores numéricos e a de saída gera probabilidades de o texto pertencer a cada uma das classes testadas ou de um ou mais tokens pertencerem à resposta da pergunta formulada. Com suas próprias respostas, a rede calcula o valor da função perda ou objetivo e, com base neste, rebalanceia os pesos para tentar melhorar seu desempenho na próxima rodada de alimentação dos dados de treinamento. As camadas escondidas são todas aquelas situadas entre a entrada e a saída, servindo para processar e representar os dados de forma mais complexa. A primeira camada escondida recebem os vetores processados pela entrada e, por sua vez, processa-os por meio de suas unidades e transmite-os para a camada seguinte, e daí em diante.

Cada unidade recebe valores x_i das unidades da camada anterior, atribui os pesos w_i correspondentes, consolida os valores e infere o total em uma função de ativação φ inerente à unidade e que é um hiperparâmetro. Um dos pesos (b), denominado viés (bias), sempre recebe valor unitário fora da camada anterior, sendo análogo ao intercepto de regressão linear. O resultado da função é

transmitido para a camada seguinte. A Figura 3 sintetiza esse processo. A saída y_k de uma unidade k com m entradas será dada por:

$$y_k = \phi_k\left(\sum_{i=1}^{m} x_{ki} w_{ki} + b_k\right) \tag{62}$$

Figura 3 - Estrutura de uma unidade (neurônio) k e suas conexões



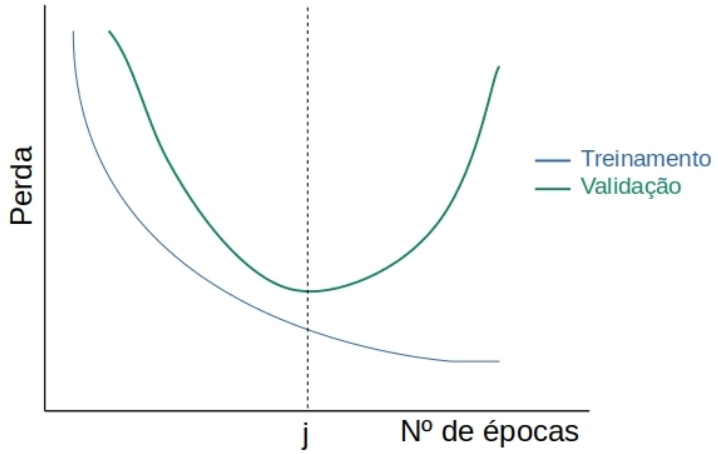
Fonte: [62]

As redes neurais são especificadas pelos tipos de unidades que as constituem, a disposição das unidades, a quantidade de camadas e as formas de conexão das unidades. Essas características compõem a **arquitetura** de uma rede. Cada arquitetura pode ser considerada como equivalente a um algoritmo clássico de AM, de forma que várias arquiteturas distintas são testadas durante a resolução de um problema.

O treinamento de uma rede neural é realizado sequencialmente, segundo o qual cada observação do conjunto de treinamento alimenta a entrada e é processada até a camada de saída. A rede usa um **algoritmo de aprendizado** para minimizar o erro das previsões da camada de saída e é comum que, para estabilizar a sua própria convergência, o algoritmo atualize os pesos somente após a processamento de um número determinado de observações chamado de lote (*mini-batch*). Desse modo, o **tamanho do lote** (*batch size*) é um hiperparâmetro do modelo de rede neural. Quando a totalidade das observações de teste passaram por todas as camadas da rede, diz-se que o algoritmo executou uma **época** e os lotes são montados de forma aleatória novamente para iniciar uma nova época. Um **treino** é, portanto, formado por uma quantidade pré-determinada de épocas.

As redes neurais são preferencialmente treinadas e avaliadas utilizando-se os três conjuntos de dados descritos na Seção 3.2.1 para viabilizar o uso da técnica ***early stopping***. A técnica serve para evitar o sobreajuste decorrente da conformação gradativa do modelo ao conjunto de treinamento com o decorrer das épocas. Ao final de cada época, o algoritmo estima o erro de generalização no conjunto de validação fixo e, caso ele deixe de diminuir por um dado número de épocas consecutivas (hiperparâmetro denominado **paciência**), o algoritmo interromperá o treino. Para que o *early stopping* funcione, é preciso que o treino compreenda uma quantidade de épocas suficientemente grande para que a interrupção sempre ocorra antes de se atingir a última época. Assim, além de evitar o sobreajuste, o *early stopping* é um método eficiente (segundo Bengio [63], "quase gratuito") de determinar o número ótimo de épocas de um treino. Finalmente, como o conjunto de validação é acessado constantemente pelo mecanismo de *early* stopping, um conjunto apartado de teste é necessário para avaliar o modelo final eficazmente. A Figura 4 mostra as típicas curvas de aprendizagem de uma rede neural, expressas pela evolução da função perda, ao longo das épocas de um treino. No início, as perdas relativas aos conjuntos de treinamento e validação decrescem mas, a partir da época j, a perda do conjunto de validação começa a oscilar e/ou crescer gradativamente. A fim de minimizar o risco de parar o treino em um mínimo local, pode-se aplicar o hiperparâmetro paciência de duas a dez épocas. Os pacotes modernos de AM gravam e resgatam os pesos do modelo correspondente à época em que a perda mínima foi verificada.

Figura 4 - Curvas de aprendizagem de uma rede neural



As principais funções de ativação e arquiteturas de redes profundas serão apresentadas nas próximas subseções.

3.2.6.1. Funções de ativação

As funções de ativação estão presentes em todas as unidades de uma rede neural.

3.2.6.1.1. Função Softmax

A função softmax converte um vetor de valores numéricos em um vetor de probabilidades, as quais são proporcionais à escala relativa de cada valor no vetor original. Por essa característica, a softmax é normalmente acoplada à camada de saída para resolver problemas de classificação. Sua expressão é:

$$f(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \tag{63}$$

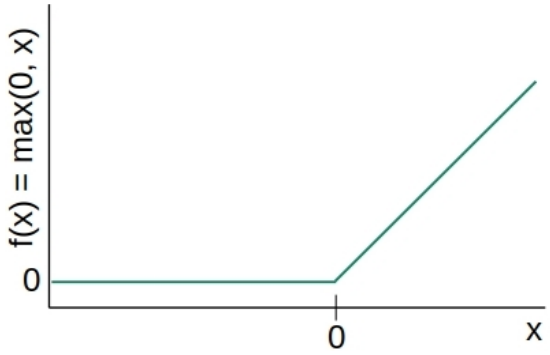
Na fórmula, z_i é o i-ézimo elemento do vetor $\vec{z}$, que pode reunir atributos ou valores provenientes de uma camada escondida, e K é o número de classes. O denominador normaliza o valor da função.

3.2.6.1.2. Função Unidade Linear Retificada (ReLU)

A função *Rectified Linear Unit*, mais comumente referida como **ReLU**, é simplesmente dada por f(x) = max(0, x). Por ser simples, fácil de calcular (assim como sua derivada, fora de x=0) e por não sujeitar a rede neural à perda de informação dos gradientes (*vanishing gradient*), que prejudica a convergência, ela é tida como a "função de ativação padrão recomendada para a maioria das redes de arquitetura *feedforward* [14]" [43]. Seu gráfico é mostrado na Figura 5.

14 As redes *feedforward* serão descritas brevemente na Seção 3.2.6.2.1.

Figura 5 - Gráfico da função ReLU



Fonte: [62]

3.2.6.1.3. Função Sigmoide

É denominada sigmoide qualquer função cujo gráfico exibe a forma de uma letra "S". Ela se caracteriza por ser diferenciável e monótona crescente, além de apresentar um equilíbrio suave entre o comportamento linear e não linear [62]. Um exemplo é a função logística dada por:

$$f(x)=\frac{1}{1+\exp(-\alpha x)} \tag{64}$$

Na equação, α é denominado parâmetro angular. É comum selecionar funções com imagem entre 0 e 1 (caso da função logística), assim como entre -1 e 1, como a função tangente hiperbólica tanh(x).

A função sigmoide também pode ser empregada na camada de saída. Nesse caso, as probabilidades determinadas por ela não necessariamente somam um, portanto seu uso é recomendado para categorias que não são mutuamente excludentes.

3.2.6.2. Arquiteturas de redes neurais profundas

Como mencionado anteriormente, as redes neurais profundas caracterizam-se por apresentar pelo menos duas camadas escondidas. Aumentar o número de camadas escondidas incrementa a profundidade da rede e a capacidade de efetuar

representações mais complexas, o que também ocorre ao preencher as camadas com mais unidades.

Nos problemas de PLN, cada documento é submetido à rede neural de modo sequencial, isto é, uma de suas palavras constituintes por vez. Para tanto, cada palavra precisa ser previamente convertida em um escalar ou, mais usualmente, um vetor. As redes mais complexas conseguem reter informações das palavras já processadas e combiná-las com as informações das próximas, replicando o comportamento do cérebro humano ao ouvir uma frase. Ao final do processamento da última palavra, a unidade emitirá um dado de saída, usado para efetuar a classificação.

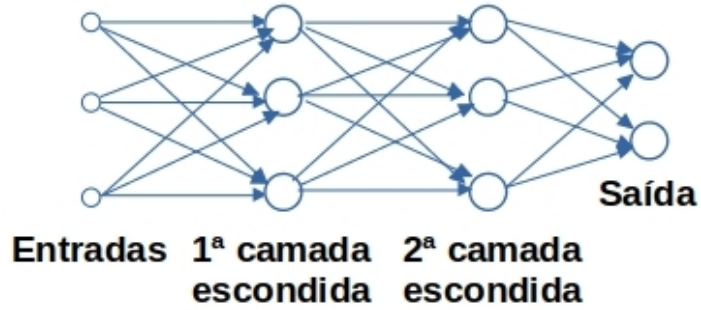
A seguir, serão apresentadas as arquiteturas de redes profundas mais empregadas na resolução de problemas de classificação de textos e extração de informações.

3.2.6.2.1. Redes Perceptron Multicamadas

As redes Perceptron Multicamadas (*Multilayer Perceptron* ou MLP) são as redes profundas mais simples existentes. Suas camadas possuem unidades com a mesma função de ativação, as quais recebem e transmitem informações de forma igual e em sentido único, à camada de saída constituída por uma função sigmoide ou softmax para efetuar a classificação. Esse fluxo unidirecional caracteriza a arquitetura *feedforward*. Todas as unidades de uma camada se conectam às da camada subsequente e, por isso, a rede também é chamada de densa ou plenamente conectada (*fully connected*). Cada unidade de camada escondida processa os dados conforme (62) e a Figura 3. A Figura 6 mostra o exemplo de uma rede profunda MLP com duas camadas escondidas com 3 unidades cada.

As redes MLP são capazes de resolver problemas de classificação de textos com muita rapidez devido à sua simplicidade e por receber cada documento convertido em um único vetor de palavras, diferentemente da sequência de palavras descrita anteriormente e tal como nos algoritmos clássicos.

Figura 6 - Rede MLP com duas camadas escondidas



Fonte: [62]

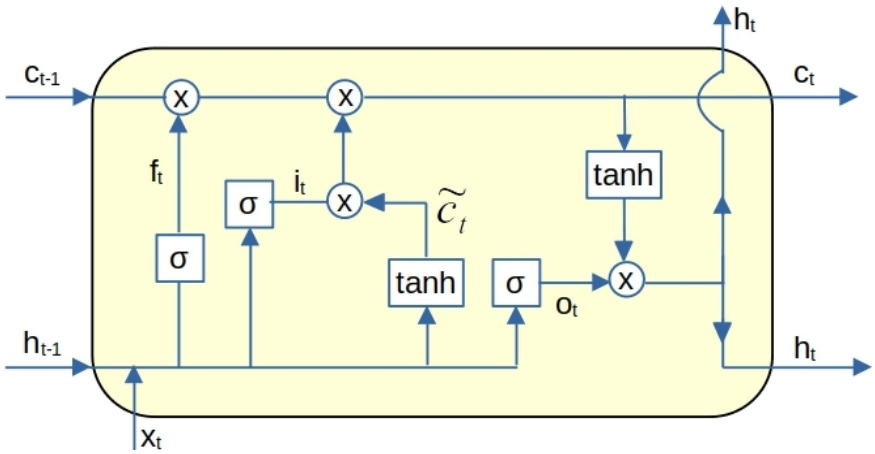
3.2.6.2.2. Redes *Long Short-Term Memory* (LSTM)

Proposta em 1997 [64], a rede LSTM possui conexões recorrentes que trazem saídas geradas nas iterações anteriores para complementar os dados de entrada da iteração corrente, ou seja, retroalimentam a própria rede. Dessa forma, a rede consegue capturar os aspectos temporais das observações do conjunto de treinamento. Isto é útil para aprender conhecimento advindo da linguagem humana, já que esta é formada primordialmente por frases, onde o sentido de uma palavra depende das palavras anteriores.

As unidades da rede LSTM são mais complexas que as da rede MLP. Possuem diversos componentes, mostrados na Figura 7, que possuem papéis distintos. O primeiro deles é o **estado da célula** (*cell state*), simbolizado pela linha horizontal na parte superior da figura, cuja função é armazenar a memória da rede. Além dos dados de entrada x_t da iteração corrente t, a unidade recebe duas informações da iteração anterior: o estado da célula c_{t-1} e a saída (ou **estado oculto**) h_{t-1}. A unidade recebe as informações de c_{t-1} e as filtra com o auxílio de uma camada sigmoide que recebe a soma dos vetores x_t e h_{t-1} ponderada pelos seus respectivos pesos. Esse filtro (ou portão) é denominado ***forget gate*** (f_t) por usar x_t e h_{t-1} para avaliar quais informações de c_{t-1} devem ser eliminadas por irrelevância, já que a camada sigmoide emite um vetor com elementos entre zero e um que é multiplicado pelos elementos de c_{t-1} (a multiplicação *element-wise* é simbolizada pelo "x" na figura). À direita do primeiro filtro há outro semelhante,

denominado ***input gate*** (i_t) que determina as informações de $x_t + h_{t-1}$ a serem armazenadas no estado da célula. A camada tangente hiperbólica processa a soma ponderada $x_t + h_{t-1}$ como uma rede neural composta de apenas uma camada. O vetor resultante da camada é chamado de **candidato** ($\tilde{c}_t$), pois representa o que seria adicionado ao estado da célula antes de passar pela *input gate*.

Figura 7 - Estrutura de uma unidade de rede LSTM



Fonte: [65]

Portanto, o estado da célula anterior é filtrado pelo *forget gate* e atualizado pelo *input gate* para ser transmitido para a próxima iteração (c_t).

O último filtro à direita da figura é o ***output gate*** (o_t), que estabelece a porção do estado da célula a ser transmitida à saída h_t, ou estado oculto corrente da unidade. O conteúdo atualizado c_t passa pela camada tangente hiperbólica e é filtrada pela camada sigmoide, que recebe a soma ponderada $x_t + h_{t-1}$.

As equações que regem o funcionamento descrito da unidade são explicitadas em (65). W_{in} e W representam as matrizes de pesos – viés incluído – das conexões da unidade com os dados de entrada e do estado oculto, respectivamente, enquanto os índices sobrescritos indicam o filtro ao qual cada matriz pertence. O sinal ○ indica o produto dos elementos de vetores ou matrizes.

$$
\begin{aligned}
f_t &= \sigma\left(x_t W_{\text{in}}^{f} + h_{t-1} W^{f}\right) \\
i_t &= \sigma\left(x_t W_{\text{in}}^{f} + h_{t-1} W^{h}\right) \\
o_t &= \sigma\left(x_t W_{\text{in}}^{o} + h_{t-1} W^{o}\right) \\
\widetilde{c}_t &= \tanh\left(x_t W_{\text{in}}^{c} + h_{t-1} W^{c}\right) \\
c_t &= c_{t-1} \circ f_t + \widetilde{c}_t \circ i_t \\
h_t &= \tanh\left(c_t\right) \circ o_t
\end{aligned}
\tag{65}
$$

Pode-se notar que a rede LSTM possui uma quantidade considerável de pesos W a serem calculados. Essa rede é capaz de aprender dependências de longo prazo, receber sequências de vetores na entrada e/ou emitir sequências na saída. Suas aplicações compreendem as séries temporais de forma geral, o reconhecimento de escrita manual, imagens e voz, geração de escrita, tradução automática, além de legendagem de imagens.

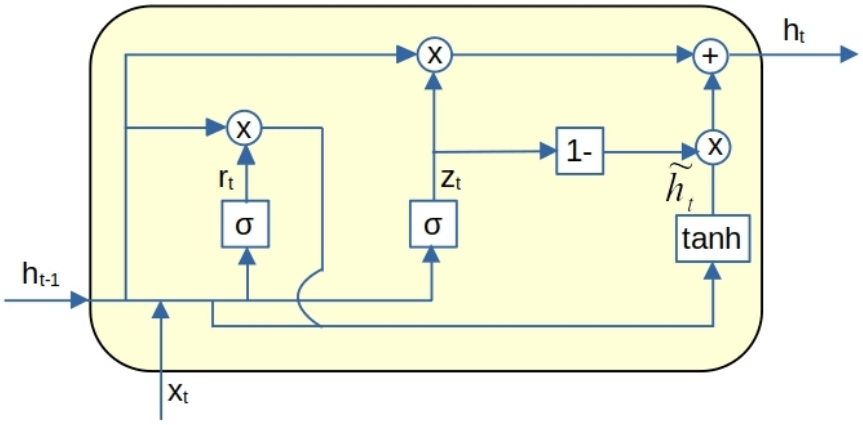
Reiterando explicação anterior, a resolução do problema de classificação é finalizada ao acoplar uma rede MLP à última camada da rede LSTM.

3.2.6.2.3. Redes *Gated Recurrent Unit* (GRU)

A rede GRU possui funcionamento similar ao da rede LSTM, sendo que o *forget gate* e o *input gate* são substituídos por um único filtro denominado *update gate* Z_t. Este controla a memória de longo prazo do estado oculto, o qual é única memória da rede. O *reset gate* (R_t) é responsável pela memória de curto prazo. O diagrama esquemático da rede está na Figura 8 e as equações correspondentes, listadas em (66).

$$
\begin{aligned}
Z_t &= \sigma\left(x_t W_{\text{in}}^{z} + H_{t-1} W^{z}\right) \\
R_t &= \sigma\left(x_t W_{\text{in}}^{r} + H_{t-1} W^{r}\right) \\
H_t^{*} &= \tanh\left(x_t W_{\text{in}} + \left(R_t \circ H_{t-1}\right) W\right) \\
H_t &= Z_t \circ H_{t-1} + \left(1 - Z_t\right) \circ H_t^{*}
\end{aligned}
\tag{66}
$$

Figura 8 - Diagrama de uma unidade de rede GRU



Fonte: [66]

A rede GRU foi apresentada em 2014 [67] como uma alternativa menos computacionalmente custosa que a LSTM. Chung et al. [68] e Greff et al. [69] concluíram que as duas arquiteturas possuem desempenho similares em tarefas como predição de sequências de músicas, modelagem de sinais de voz, bem como reconhecimento de voz, escrita manual e música.

3.2.6.2.4. Redes Convolucionais

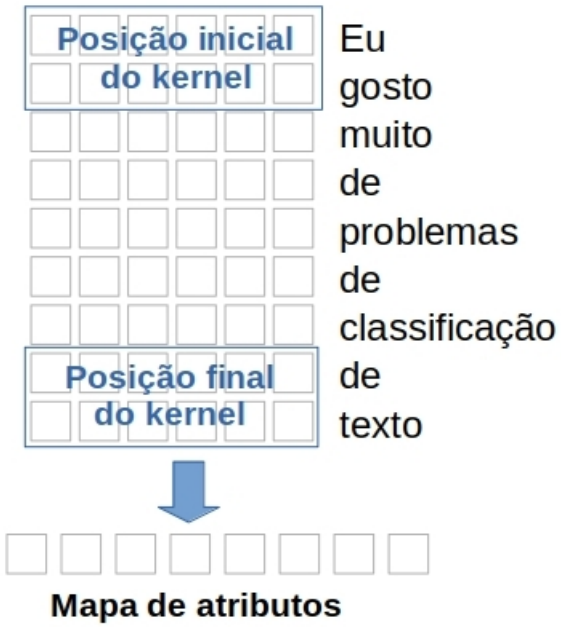
A rede convolucional (CNN) é formada por três tipos de camadas:

i. camadas convolucionais, que extraem características dos dados de entrada;
ii. camadas de amostragem (*pooling*), que reduzem a dimensionalidade dos dados provenientes das camadas convolucionais e sempre as acompanham; e
iii. camadas densas para realizar o processamento final e a classificação.

A camada convolucional usa filtros ou **kernels** que percorrem os dados para extrair características. A Figura 9 mostra um filtro unidimensional usado para dados textuais, enquanto entradas em forma de imagens requerem kernels 2D. Na figura, a frase "Eu gosto muito de problemas de classificação de texto" é filtrada por um kernel de tamanho igual a dois, ou seja, que se aplica a duas palavras ao mesmo tempo. O kernel se moverá 8 vezes até chegar no fim da frase e produzirá um vetor com 8 elementos denominado **mapa de atributos**. Cada palavra é representada por um vetor com 6 elementos. Nessa configuração o kernel possui

12 pesos dispostos na matriz W_k (dimensões 2 x 6) e uma função de ativação f. Um elemento e_i do mapa de atributos será igual a $f(W_k \cdot M_i)$, ou seja, resulta de uma função aplicada sobre o produto escalar entre W_k e a matriz M_i (2 x 6) correspondente à i-ézima passagem do kernel sobre duas palavras. No caso de uma imagem, um filtro pode destacar certas características ou cores no seu resultado, por isso se diz que o kernel é capaz de extrair atributos por si mesmo. Assim, a camada convolucional pode servir como seletora de atributos para outros algoritmos.

Figura 9 - Filtro unidimensional aplicado a texto



Fonte: [70]

A camada seguinte é a de ***pooling*** ou amostragem e agrega os atributos em apenas um deles. A forma mais popular de agregação é a *max pooling*, que seleciona o atributo de maior valor. A rede é finalizada por uma ou mais camadas densas que funcionam como redes MLP e mapeiam as características extraídas pelas camadas anteriores e a camada de saída.

3.2.6.2.5. Mecanismo de atenção

O mecanismo de atenção (*attention*) foi proposto por Bahdanau et al. em 2016 [71] como aperfeiçoamento de uma estrutura de codificador-decodificador

(*encoder-decoder*) formada por duas redes LSTM, para melhorar o seu desempenho em tradução automatizada. A atenção identifica relacionamentos entre palavras distantes em uma frase e, ao realizar uma predição ou tradução sobre um termo, ela ignora palavras próximas a ele que sejam irrelevantes para a tarefa. Dentre as diversas aplicações do mecanismo, será descrita uma forma simples, utilizada em problemas de classificação e adaptada do artigo original. O mecanismo de atenção normalmente recebe saídas h_j processadas por uma rede LSTM, uma rede de outra arquitetura ou um codificador, doravante chamada de rede originadora. As saídas são denominadas **anotações** no artigo original. Sua soma ponderada pelos **pesos de atenção** α_{ij} resulta no vetor de **contexto** c_i, dado por:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{67}$$

O total T_x representa o número de palavras vetorizadas e sequenciadas $x = (x_1, x_2, \ldots, x_n)$, que alimentaram a rede e podem ser uma frase, por exemplo. A saída h_j enfoca as palavras que estão ao redor de x_j, isto é, a j-ézima palavra da sequência de entrada. Em um problema de tradução, o contexto c_i serve para predizer a tradução y_i de um termo x_i, considerando também as palavras traduzidas anteriores $y_1, y_2, \ldots, y_{i-1}$. Os pesos α_{ij} são calculados em função dos modelos de alinhamento e_{ij} (dispostos numa matriz coluna E) que capturam o grau de concordância entre a entrada x_j e saída y_i, como segue.

$$\alpha_{ij} = \frac{\exp(\tanh e_{ij})}{\sum_{k=1}^{T_x} \exp(\tanh e_{ik})} = softmax[\tanh(E)] \tag{68}$$

No presente caso, o modelo de alinhamento é dado por uma camada com unidades simples e regidas por (62). Expressando então os modelos e_{ij} sob a forma matricial, tem-se:

$$E_{T_x x 1} = H_{T_x x d} W_{dx1} + B_{T_x x 1} \tag{69}$$

Na equação, H é a matriz das anotações, enquanto B e W são matrizes coluna dos vieses b e pesos da camada com unidades simples, respectivamente. A dimensão d é igual à quantidade de unidades da rede originadora.

Assim, a presente versão do mecanismo de atenção pode ser resumida em uma pequena rede MLP, formada por uma camada de neurônios ativada por uma função tangente hiperbólica e seguida por uma camada softmax, que produz vetores de contexto. A diferença fundamental entre a atenção e uma rede MLP que comumente realiza a etapa final de classificação (após a passagem dos dados pela rede originadora) é o fato de a primeira receber as saídas referentes a todas as palavras que alimentaram a rede originadora, e não apenas a correspondente ao último termo. Portanto, os vetores de contexto atribuem relevância aos conteúdos das palavras, com pesos calculados pela rede MLP.

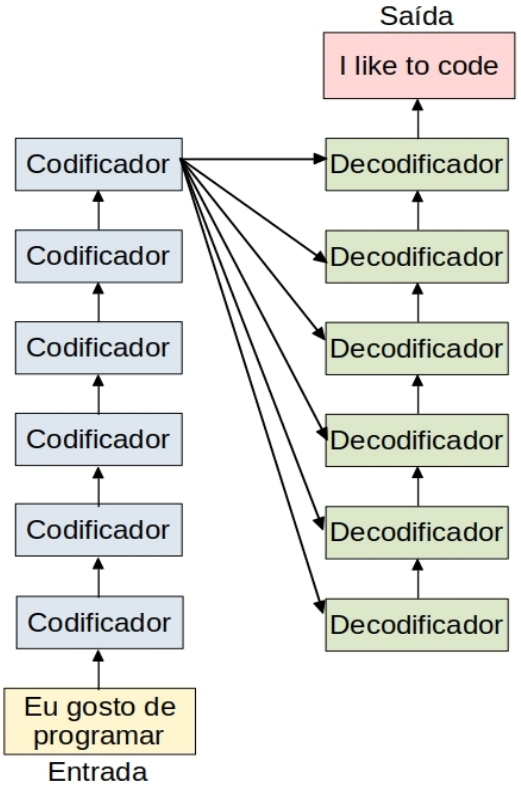
3.2.6.2.6. *Transformer*

A arquitetura de rede *Transformer* foi lançada em 2017 [72] e faz uso intenso de mecanismos de atenção e redes densas, dentro de uma estrutura de codificador-decodificador, para examinar as dependências globais entre os dados de entrada e saída. O codificador transforma uma sequência de palavras $x = (x_1, x_2, \ldots, x_n)$ em outra sequência de representações contínuas $z = (z_1, z_2, \ldots, z_n)$, a qual serve de entrada para o decodificador, que por sua vez gera uma saída $y = (y_1, y_2, \ldots, y_n)$ de forma sequencial, i.e., um y_i a cada instante de tempo. Essa técnica é denominada *Sequence-to-Sequence* (Seq2Seq) e aplica-se a tarefas de PLN que exigem uma saída formada por uma sequência de palavras ou frases, como tradução, resumo de textos, sistemas de respostas a perguntas (*question answering*), reconhecimento de fala e texto manuscrito.

O codificador e o decodificador do *Transformer* são compostos por seis camadas iguais e sequenciais, sendo que a saída do codificador (emitida por sua última camada) alimenta todas as camadas do decodificador (Figura 10). Esse mecanismo enfatiza a saída codificada em cada etapa de decodificação. Cada camada do codificador é formada por duas subcamadas: a primeira contém um

mecanismo de autoatenção e a segunda é uma rede densa. O mecanismo de **autoatenção** (*self-attention*) ou atenção interna procura calcular, em relação a uma dada palavra, as pontuações indicativas das importâncias de todas as outras palavras da sequência, mesmo que estejam separadas por várias posições. As subcamadas de um codificador (parte esquerda) e um decodificador (parte direita) são mostradas na Figura 11.

Figura 10 - Estrutura macro da rede Transformer



Fonte: [73]

Nos experimentos do artigo original, os autores definiram que cada palavra seria representada por um vetor com 512 posições e esta dimensão d_m (*dimension model*) = 512 será mantida ao longo de toda a rede. Portanto, representa-se um documento ou uma frase com n palavras por uma matriz X com dimensões n x d_m.

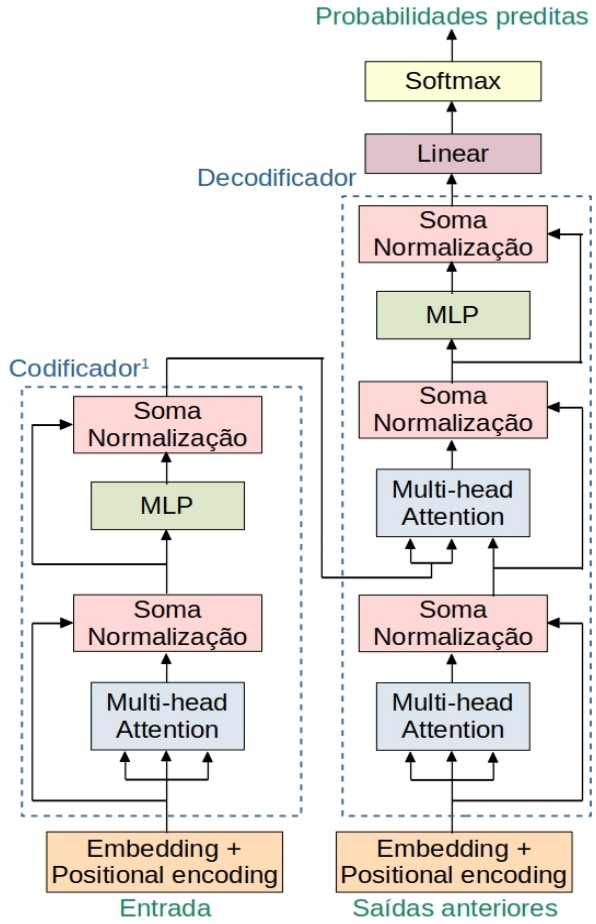
O processamento de uma sequência de palavras pela rede *Transformer* se dá em seis etapas, mostradas na Figura 11:

i. Codificação posicional (*positional encoding*). Nas entradas do codificador e do decodificador, o vetor 1 x d_m representativo de uma palavra (*embedding*) é somado a um vetor 1 x d_m de codificação

posicional PE, relativa à posição pos da palavra na frase. As posições i do vetor PE são geradas de forma intercalada por uma senoide (posições pares) e uma cossenoide (posições ímpares):

$$PE(pos,\ 2i) = sen(pos/10000^{2i/d_m})$$
$$PE(pos,\ 2i+1) = \cos(pos/10000^{2i/d_m}) \quad (70)$$

Figura 11 - Subcamadas da rede *Transformer*



Fonte: [72]

A PE permite à rede guardar a ordem das palavras no documento de entrada, já que os vetores de palavras percorrem as camadas da rede em paralelo.

ii. Mecanismo de autoatenção (*multi-head self-attention*). A matriz de entrada X é ponderada por três matrizes de pesos W^Q, W^K e W^V, aprendidas pela rede e com d_q, d_k e d_v linhas respectivamente e n

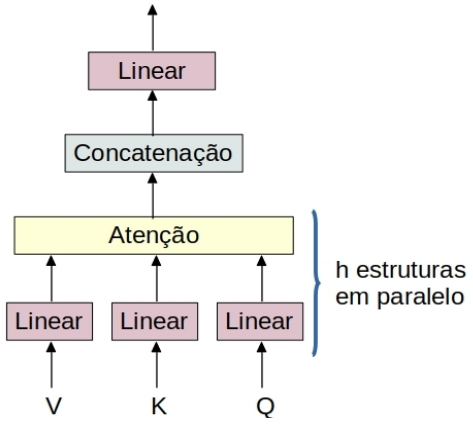
colunas. No artigo original, $d_k = d_v = 64$ e, para viabilizar os produtos, X terá suas 512 colunas particionadas de acordo com d_k e d_v. Assim, os produtos XW^Q, XW^K e XW^V geram três representações diferentes de X: as matrizes Q, K e V (n x d_k), respectivamente. Cada linha de Q (*query*) corresponde a uma palavra a ser analisada. K (*key*) representa as informações existentes em todas as palavras da frase, inclusive as analisadas, e V (*value*), as informações desejadas de todas as palavras. O produto QK^T equivale aos pesos de atenção, ou seja, a importância de cada palavra relativa às palavras analisadas. A matriz de atenção de um documento ou frase é dada por:

$$Atenção(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{71}$$

A função softmax e o denominador $\sqrt{d_k}$ servem para normalizar o resultado parcial e estabilizar gradientes durante o treinamento, respectivamente. O nome autoatenção vem do fato de a própria entrada X produzir todos os elementos Q, K e V suficientes para a rede aprender sobre o que merece atenção.

As h partições de X são ponderadas por h diferentes matrizes W^Q, W^K e W^V, cujo processamento se dá em paralelo. São geradas portanto múltiplas matrizes ou **cabeças de atenção** (*multi-head*) (Figura 12), que são concatenadas e ponderadas por uma matriz de pesos W^O, resultando em uma matriz de atenção múltipla com dimensão n x d_m.

Figura 12 - Atenção com múltiplas cabeças



Fonte: [72]

iii. Conexão residual. Cada subcamada soma a sua saída com a sua entrada e normaliza o resultado (caixas "Soma Normalização" da Figura 11). A conexão residual objetiva combater o fenômeno de degradação da acurácia do modelo [74], que se agrava à medida que ele se torna mais profundo, i.e., possui mais camadas.

iv. Rede densa *feedforward*. As saídas da subcamada de autoatenção alimentam uma subcamada formada por uma rede MLP ou densa, que realiza duas transformações lineares mediadas por uma função de ativação ReLU. A saída de uma dada unidade, para uma posição i, será função de duas matrizes de pesos W_1 e W_2, além de dois vetores de vieses b_1 e b_2:

$$f(x_i) = max(0, x_1 W_1 + b_1) W_2 + b_2 \tag{72}$$

v. Decodificação. O decodificador recebe as saídas do codificador e as suas próprias saídas finais anteriores (se for o primeiro decodificador) ou a saída da camada de decodificação precedente. Ele possui as mesmas duas subcamadas que o codificador, além de uma subcamada de autoatenção extra que recebe posições subsequentes mascaradas para considerar apenas as posições anteriores são usadas na predição da posição em análise. A subcamada de atenção recebe, além do vetor *query* da camada de decodificação anterior, os vetores *key* e *value* da saída do codificador, portanto a decodificação de qualquer trecho observa todas as posições da sequência de entrada.

vi. Camada linear e softmax. A saída final do decodificador é um vetor de números reais com dimensão 1 x 512 que passa por uma camada de rede densa (caixa "Linear" da figura) para transformá-lo em um vetor 1 x tamanho do vocabulário. A camada softmax se encarrega de converter o último em um vetor de probabilidades para se selecionar a palavra mais provável.

Os pesquisadores do artigo original treinaram uma rede por 3,5 dias, a qual atingiu resultados de estado-da-arte para traduções textuais inglês-alemão e inglês-francês. O modelo demonstrou seu poder de generalização ao conseguir

boas pontuações em tarefas de análise sintagmática em inglês (*constituency parsing*).

3.2.6.2.7. *Bidirectional Encoder Representations from Transformers* (BERT)

Lançado por pesquisadores do Google em 2018 [75], o BERT é um modelo de representação de linguagem destinado à execução de diversas tarefas de PLN, como classificação de textos, extração de informações e inferência de idiomas. Sua engrenagem fundamental é o codificador da rede *Transformer*, analisada anteriormente.

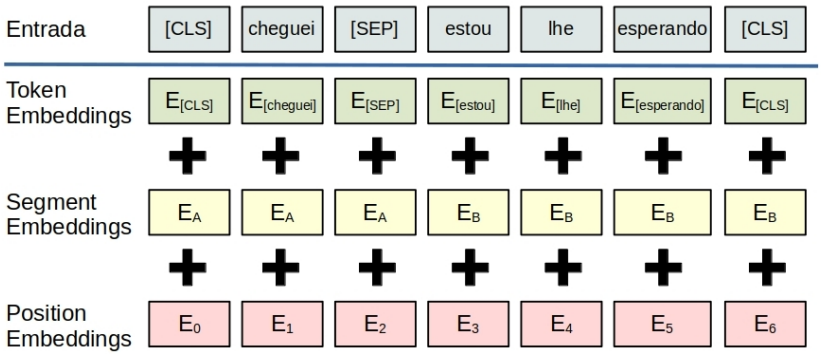
As características principais do BERT são:

a) Contextualidade. A representação por ele produzida é contextual, ou seja, cada palavra é representada com base nas outras palavras da sentença. Dessa forma, homônimos e parônimos como a palavra banco, que possui os sentidos de instituição financeira e elevação do fundo do mar, conseguem ser diferenciados. Isto é uma evolução em relação a modelos sem contexto, como o word2vec discutido adiante.

b) Bidirecionalidade. O BERT dispõe os codificadores da rede *Transformer* de forma bidirecional, isto é, as subcamadas de autoatenção trabalham com as palavras anteriores e posteriores à palavra em análise. Essa funcionalidade é importante para a compreensão do texto para realizar tarefas como a busca automática de respostas a perguntas.

c) Representação particular. O BERT adota representações peculiares que possibilitam identificar um par de frases e as suas palavras constituintes dentro de uma sequência de palavras que lhe é apresentada. A Figura 13 mostra os três tipos de representação que o modelo atribui a uma palavra: uma intrínseca (*token embedding*) ao seu significado e ao contexto e outras duas indicadoras da frase à qual pertence (*sentence embedding*) de sua posição na frase (*position embedding*). O modelo também usa palavras especiais para marcar o começo de uma sequência e de uma sentença.

Figura 13 - Representações de entrada do BERT



Fonte: [75]

d) Abordagem de **sintonia fina**. Os modelos modernos trabalham com o treinamento prévio em corpora de grande escala para aprender as representações. A abordagem de sintonia fina prevê o ajuste fino de todos os parâmetros pré-treinados e o uso de poucos parâmetros específicos para a tarefa a ser realizada. Já a abordagem baseada em atributos emprega arquiteturas específicas para a tarefa e as representações pré-treinadas são considerados atributos adicionais.

e) Transferência de aprendizado (*transfer learning*) de alto desempenho. A dificuldade de obter conjuntos de dados rotulados é contornada quando se dispõe de um modelo pré-treinado com corpora volumosos, dotado de grande conhecimento da linguagem e sua estrutura. As tarefas realizadas no pré-treinamento e na efetiva utilização do modelo em geral são diferentes, mas relacionadas. O alto desempenho provém do uso de mecanismos de autoatenção e atenção da rede *Transformer*, os quais permitem que o conhecimento prévio seja comparado repetidamente com as palavras da tarefa desempenhada.

Os modelos pré-treinados BERT oferecidos[15] ao público dividem-se em dois grupos. Os modelos grandes foram treinados com 24 camadas codificadoras da rede *Transformer* e 16 cabeças de autoatenção, enquanto os modelos-base, com 12 camadas codificadoras e 12 cabeças de autoatenção. É possível usar os modelos pré-treinados diretamente na tarefa desejada sem treinamento algum, procedimento chamado de *zero-shot learning*. No entanto, é preferível realizar a

15 https://github.com/google-research/bert

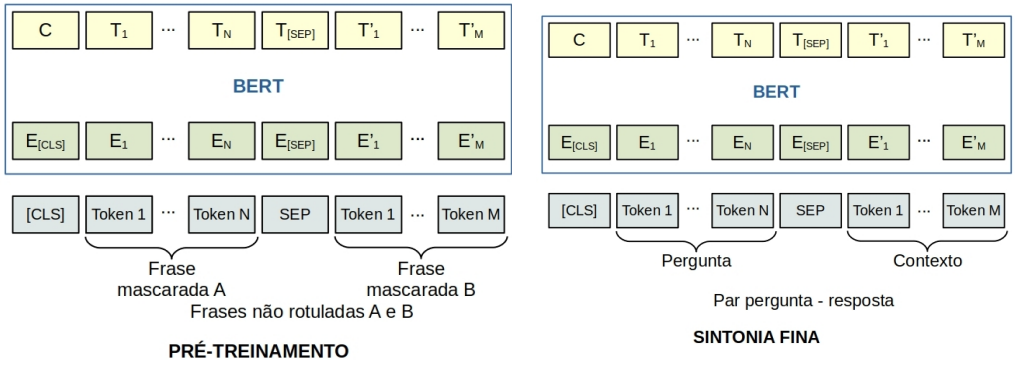
sintonia fina sob a forma de um pequeno treinamento supervisionado e direcionado à tarefa a desempenhar.

Os pesquisadores do Google efetuaram o pré-treinamento dos modelos em duas tarefas supervisionadas. A primeira é a *Masked Language Model* (MLM) e consiste em predizer algumas palavras aleatoriamente mascaradas. A segunda é a predição da sentença subsequente (*Next Sentence Prediction* - NSP), na qual o modelo foi submetido a baterias de testes compostos sempre por duas frases. Em metade dos testes as duas frases eram adjacentes em algum texto e na outra metade as frases não tinham relação entre si. Essa tarefa é importante para o modelo capturar as relações entre frases. Os autores classificam os pré-treinamentos como não supervisionados, porém cabe observar os próprios textos empregados são a fonte das respostas que orientaram o cálculo das métricas de desempenho, da função objetivo etc. Dessa forma, é possível entender que houve supervisão com rótulos que prescindiram de esforço humano. O pré-treinamento utilizou o BooksCorpus, com 800 milhões de palavras, e a Wikipedia em inglês, com 2,5 bilhões de palavras.

A Figura 14 ilustra a semelhança entre as representações adotadas pelo BERT nas etapas de pré-treinamento (tarefa MLM) e sintonia fina (perguntas e respostas ou *question answering*). A primeira representação divide e mascara uma dupla de frases para predizer as palavras faltantes de ambas. Na tarefa de *question answering* em sintonia fina, o BERT recebe simultaneamente a pergunta e o parágrafo que contém a resposta (chamado tecnicamente de **contexto**), com vistas a identificar o início e o final do trecho com a solução. A arquitetura interna da rede permanece inalterada e somente a camada de saída é adaptada à tarefa. Pode-se também aproveitar a representação gerada pelo BERT em algoritmos clássicos.

O artigo original descreve os resultados da aplicação do BERT em 11 tarefas de PLN, como sistema de perguntas e respostas, previsão da próxima frase e classificação da relação entre duas frases. Em geral, os modelos grandes alcançaram desempenho superior aos dos modelos unidirecionais e com abordagem baseada em atributos. O BERT é indicado para tarefas que envolvem a compreensão da linguagem natural (*Natural Language Understanding* ou NLU), enquanto outros modelos são mais recomendados para a geração de textos.

Figura 14 - Representações adotadas pelo BERT no pré-treinamento e na sintonia fina



Fonte: [75]

3.3. Técnicas de subamostragem para equilibrar a representação das classes

O problema de classificação trabalhou com uma amostra de treinamento composta de mais de 33 mil documentos, dos quais 85% pertenciam à classe dominante. As outras duas classes minoritárias compreendem os documentos que abordam os dois tipos de operações societárias de interesse. Segundo Fernández et al. [76], os algoritmos de AM em sua forma padrão tendem a favorecer a classe majoritária, porque classificar as suas instâncias corretamente exerce maior impacto na função objetivo e/ou possibilita obter uma maior acurácia global. Assim, os resultados dos modelos para as classes de interesse serão geralmente inferiores aos da classe dominante, caso não se realizem ajustes.

Uma das soluções para o problema de desbalanceamento entre classes consiste em atuar sobre os dados disponíveis, antes de seu processamento pelos algoritmos de AM. As abordagens de pré-processamento de dados se dividem em super e subamostragem. Como o caso concreto lidou com um número grande de documentos, deu-se preferência aos métodos de subamostragem da classe dominante, descritos a seguir, para possibilitar a execução mais rápida dos algoritmos, além de explorar uma gama maior de hiperparâmetros na validação cruzada.

3.3.1. *NearMiss*

Proposta por Zhang e Mani [77] em 2003, trata-se de uma família de métodos que seleciona observações da classe majoritária (ou negativa) com base na menor distância média entre elas e os exemplos (positivos) das classes minoritárias. As observações majoritárias mais distantes das observações positivas são retiradas da amostra, por serem menos relevantes para a classificação. O NearMiss-1 escolhe retirar as instâncias dominantes cujas distâncias médias a n observações positivas são mínimas. O NearMiss-2 elimina as observações majoritárias mais próximas de todas os exemplos minoritários. O NearMiss-3 trabalha em duas etapas: primeiro, ele identifica as m observações dominantes mais próximas de cada uma das observações minoritárias e depois retira da amostra as observações dominantes com as maiores distâncias médias medidas até as n observações positivas mais próximas (*n-nearest-neigbors*). Os hiperparâmetros dos algoritmos são n e m.

As três versões da família NearMiss permitem fixar de antemão o número desejado de observações majoritárias ao final da subamostragem. Em geral, o seu tempo de execução é curto.

3.3.2. *Tomek Links*

Este algoritmo identifica os links de Tomek [78], que são os pares de observações O_i e O_j de classes distintas cuja distância entre elas é $d(O_i, O_j)$, tal que uma delas é a observação mais próxima da outra e vice-versa. Matematicamente:

$$(O_i, O_j) \text{ é link de Tomek} \Leftrightarrow \nexists O_k \mid d(O_i, O_k) < d(O_i, O_j) \vee d(O_j, O_k) < d(O_i, O_j) \quad (73)$$

Portanto, os links de Tomek definem as fronteiras entre classes. A subamostragem consiste em eliminar as observações da classe majoritária que formam links de Tomek, por se tratarem de observações com maior chance de serem ruidosas ou para tornar a fronteira mais clara. O algoritmo não permite selecionar a quantidade de observações a serem removidas.

3.3.3. *Condensed Nearest Neighbor* (CNN)

O CNN [79] procura identificar um subconjunto consistente T' de observações, definido como um subconjunto da amostra total T que classifica corretamente todas as observações de T pelo algoritmo *k-nearest neighbors* (KNN, vide 3.2.4.4). O resultado da subamostragem será T'.

Primeiramente, o algoritmo seleciona aleatoriamente s exemplos majoritários (ou sementes) e todas as observações minoritárias para formar a primeira estimativa de T'. O KNN é aplicado para classificar todas as observações de T - T' por meio de T'. Cada elemento de T - T' classificado erradamente é transferido para T'. A classificação pelo KNN e a transferência de elementos para T' são reexecutadas até que não se consiga mais aumentar T'.

Segundo Lemâitre et al. [80], o CNN tende a adicionar observações ruidosas no subconjunto consistente. Além disso, sua execução é custosa para valores grandes de k.

3.3.4. *One-Sided Selection* (OSS)

Proposto em 1997 [81], o OSS combina os algoritmos Tomek Links, usado para remover previamente as observações majoritárias ruidosas e as próximas da fronteira de decisão, e CNN, o qual complementa a subamostragem. A etapa representada pelo CNN é executada de forma distinta da do algoritmo original: após a seleção de sementes, aplica-se o algoritmo KNN apenas uma vez a todas as observações restantes e as observações classificadas com erro são transferidas para T'. Não há iterações para adicionar mais elementos a T'. Tal como o CNN, os hiperparâmetros do algoritmo são s e k.

4
Exercícios de aplicação do processamento de linguagem natural para classificação de documentos financeiros

4.1.
Os dados

As resoluções dos dois problemas propostos utilizaram partições do mesmo conjunto de dados. A CVM recebe documentos divulgados pelas companhias abertas pelo sistema receptor Empresas.NET. Todos os documentos são públicos e podem ser acessados pelo portal da CVM[16], porém a consulta atual não retorna simultaneamente dados de mais de uma empresa. Por esse motivo, o doravante denominado **conjunto inicial** de documentos foi formado a partir de uma extração de documentos da base corporativa da CVM. O órgão disponibilizará os documentos do Empresas.NET em seu portal Dados Abertos[17] em 2021 para que o cidadão possa baixá-los em lote. Foram também extraídos os metadados dos documentos, descritos na seção 6.3.

O conjunto inicial é formado por todos os documentos submetidos por todas as companhias abertas à CVM entre 01/01/2009 e 31/12/2019 e selecionados segundo os ajustes e critérios a seguir. Escolheram-se os documentos enquadrados em oito categorias do sistema Empresas.NET, selecionadas por sua relevância para os problemas. Foram excluídos documentos compostos por mapas de votações e apresentações, por conterem muitos elementos gráficos e texto de difícil tokenização, além de documentos que sabidamente não guardavam relação com as operações societárias em questão. A exclusão foi realizada por meio dos metadados detalhados adiante. Os 202.524 documentos restantes compõem o

16 https://cvmweb.cvm.gov.br/SWB/Sistemas/SCW/CPublica/CiaAb/FormBuscaCiaAb.aspx?TipoConsult=c

17 http://dados.cvm.gov.br/

conjunto inicial. A Tabela 1 mostra a divisão dos documentos do conjunto inicial entre as categorias.

Tabela 1 - Distribuição de documentos do conjunto inicial por categoria do sistema Empresas.NET

Categoria	Nº documentos	%
Assembleia	56.766	28,0
Aviso aos Acionistas	11.928	5,9
Aviso aos Debenturistas	2.767	1,4
Comunicação sobre Transação entre Partes Relacionadas	1.112	0,5
Comunicado ao Mercado	49.675	24,5
Contratos com Partes Relacionadas	1.352	0,7
Fato Relevante	17.827	8,8
Reunião da Administração	61.097	30,2
Total	202.524	100,0

O conjunto inicial constituiu a base do CVMCorpus, corpus especializado no domínio de finanças, que será descrito em detalhe adiante no Capítulo 6. O conjunto inicial também foi o ponto de partida para a montagem da base que serviu para a análise de bigramas e determinação dos *word embeddings* pelo algoritmo word2vec.

Para a rotulação, inicialmente foram selecionados os documentos do conjunto inicial com data de recebimento entre 01/01/2018 e 16/09/2019 inclusive. Os documentos foram divididos em três classes:

- Classe "**a**": documentos com informações sobre operações de reorganização societária.
- Classe "**b**": documentos com informações sobre operações de aumento de capital.
- Classe "**n**": demais documentos.

Foi observado que as classes eram bastante desbalanceadas, com 95% dos documentos pertencentes à categoria n. Para atenuar o desbalanceamento, rotularam-se 4.371 documentos adicionais do conjunto inicial pertencentes às classes a e b, recebidos entre 2009 e 2019, para formar a **base de análise** junto com os documentos anteriores rotulados. A distribuição da base de análise entre as classes está mostrada na Tabela 2.

Tabela 2 - Composição da base de análise

Classe	Nº de documentos	%
a	2.569	6,1
b	3.593	8,5
n	35.970	85,4
Total	42.132	100,0

Os casos que ensejariam mais de um rótulo para um mesmo documento foram resolvidos com a adoção de apenas uma das possíveis classes. O primeiro caso são as operações de reorganização societária que preveem aumentos de capital como sendo uma de suas etapas, principalmente quando a companhia incorporadora emite novas ações para pagar os acionistas da empresa incorporada. Os documentos sobre tais operações foram considerados como pertencentes à classe a, já que a operação principal é a reorganização societária. O segundo caso ocorre quando um documento referencia operações distintas de reorganização societária e aumento de capital. Os 34 documentos que se enquadram nesse caso foram classificados aleatoriamente entre as classes a e b. Dessa forma evitou-se enunciar um problema de classificação multirrótulo sem perda de qualidade, já que o objetivo é destacar os documentos das duas classes relevantes para análise posterior do analista e os 34 documentos que poderiam efetivamente ser classificados com dois rótulos são raros.

A base de análise serviu de ponto de partida para a criação de seis variantes, de acordo com a forma de aproveitamento do metadado assunto do documento e adoção de bigramas:

i. Tokens do assunto agregados ao início do teor do documento, sem distinção entre as duas partes (doravante: base com assuntos ou "**as**").
ii. Tokens do assunto agregados ao início do teor do documento, porém modificados (adicionou-se o sufixo ".tt") para se diferenciarem e ganharem parâmetros particulares nos algoritmos de AM (doravante: base com assuntos modificados ou "**as_mod**").
iii. Somente tokens do teor do documento (doravante: base "**teor**").

As três bases foram divididas entre o conjunto de treinamento (33.705 documentos ou observações) e o conjunto de teste (8.427 documentos), conforme

a proporção 80%/20% e com amostragem estratificada proporcional entre as classes.

A base de assuntos modificados possui mais de 260.000 tokens únicos. Após reduzi-los para 20 dimensões por Análise Probabilística de Semântica Latente (*Latent Semantic Analysis* – LSA), estas responderam por apenas 12% da variância total da matriz tf-idf de entrada. A Figura 15 mostra uma representação em duas dimensões de todos os documentos da base obtida pelo algoritmo t-SNE (*t-Distributed Stochastic Neighbor Embedding*) após uma primeira redução para 50 dimensões por LSA. As classes "a" e "b" (representadas respectivamente pelos algarismos 1 e 2 posicionados sobre os centroides dos pontos verdes e azuis) conseguem se destacar de forma razoável da classe majoritária "n" (pontos vermelhos centrados em 0). A proximidade entre os centroides das classes "a" e "b" sugere que distingui-las é mais desafiador.

Figura 15 - Representação gráfica dos documentos da base as_mod em duas dimensões

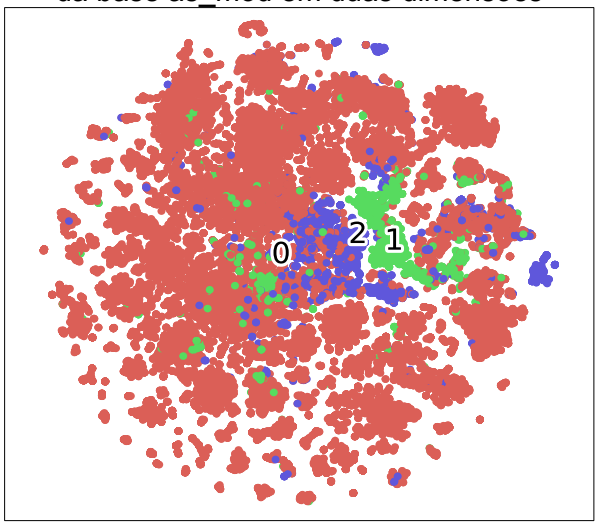
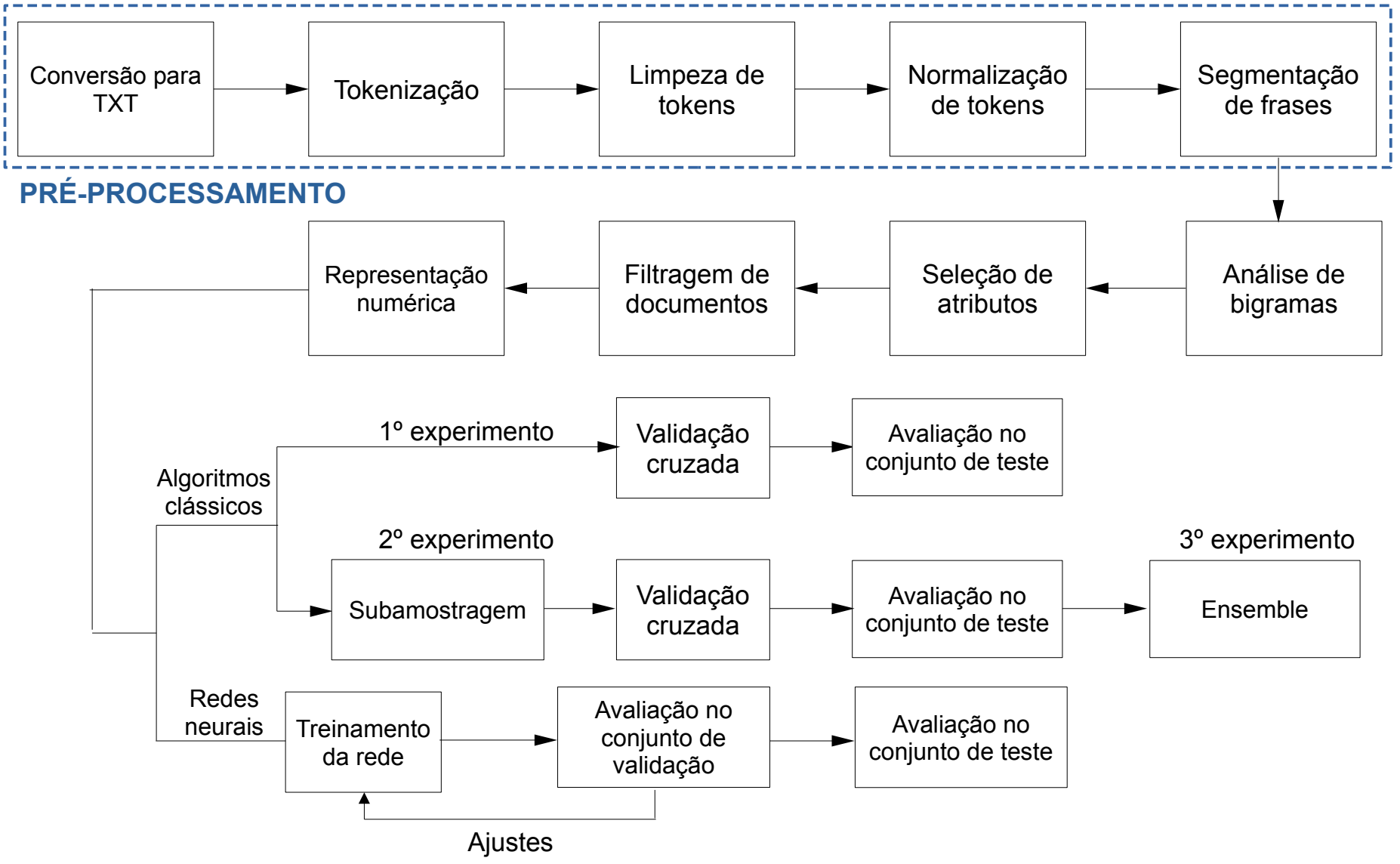


Figura 16 - Diagrama esquemático das etapas de resolução do problema de classificação



4.2. Metodologia

Como discutido anteriormente, a resolução de um problema de PLN com técnicas de AM requer a execução de uma série de procedimentos em etapas. A representação esquemática das etapas percorridas para o caso específico é mostrada na Figura 16 e detalhada nas próximas seções.

4.2.1. Pré-processamento

A etapa de pré-processamento iniciou-se com a conversão dos documentos em formato legível por linguagem de programação, tendo sido eleito o TXT. Os documentos foram submetidos pelas companhias abertas à CVM em formatos diversos como PDF, Excel, Word e Powerpoint. A sua extração da base corporativa da CVM converteu-os todos automaticamente para PDF, o que simplificou o processo de conversão, já que uma biblioteca (Apache Tika[18]) foi suficiente para trabalhar com o formato único, quando pesquisável. Por outro lado, a conversão de outros formatos para PDF deixou marcas de nomes de arquivo e de software, compostas geralmente por até dez tokens no início do documento convertido.

As companhias também forneceram arquivos PDF não pesquisáveis, provenientes de digitalização de imagens, que foram lidos pela biblioteca pytesseract[19]. Algumas imagens de baixa qualidade ocasionaram erros de leitura não sinalizados, os quais geraram textos vazios ou com caracteres ilegíveis.

A tokenização da base de análise constituiu a etapa seguinte. Tentou-se ao máximo adotar as convenções de tokenização do projeto AC/DC (Acesso a corpus/Disponibilização de corpus), integrante do portal Linguateca[20], para aproveitar os racionais definidos pelos linguistas participantes do projeto. Desse modo, foi possível construir um corpus especializado no domínio financeiro comparável, o quanto possível, aos corpora disponibilizados pelo AC/DC e útil para estudos linguísticos. Expostas no Apêndice 1, as convenções dizem respeito a

18 https://tika.apache.org/
19 https://pypi.org/project/pytesseract/
20 https://www.linguateca.pt/ACDC/

quais elementos gráficos ou caracteres foram considerados como um token, assim como separados ou unidos para formar um token.

As bases passaram por uma etapa de limpeza de tokens irrelevantes para o problema de classificação. Foram excluídos tokens formados por sinais de pontuação e numerais, além dos pertencentes a uma lista de 287 *stopwords* em português e 179 em inglês, definida por pronomes, preposições, verbos auxiliares e palavras costumeiras em documentos financeiros, como "CNPJ", "página", "nº" etc. Tokens que iniciavam em cedilha, "ã" e "õ", além de suas versões em maiúsculas, também foram eliminados da base por não representarem palavras em português. Por meio de expressões regulares, restringiram-se os tokens remanescentes àqueles que eram necessariamente iniciados por uma letra, podendo ou não apresentar sinais como hífen, cifrão, "&" e outros presentes em nomes próprios ou palavras em inglês.

A etapa de normalização consistiu em converter todos os tokens para caixa baixa e retirar todos os pontos para uniformizar as abreviaturas.

Para cada base de documentos convertidos em tokens, construiu-se outro conjunto com textos segmentados em frases com tokens limpos. O detalhamento do processo de identificação das frases está detalhado na Seção 6.2.1.

Optou-se por não aplicar lematização nem *stemming* aos tokens devido à incerteza quanto à precisão, do ponto de vista linguístico, das saídas oferecidas pelas ferramentas disponíveis para o português. Muzart [82], pesquisador do Tribunal de Contas da União ativo em PLN, classifica o *stemming* como um método de "baixo custo e baixo retorno", porque a definição do radical de uma palavra não é precisa. Um exemplo é a palavra inglesa *caring*, a qual deveria ser convertida para *care* e não *car*. Assim, o algoritmo deveria ser sintonizado para adicionar a letra "e" ao final do radical, complicando o processo. Viera e Virgil [83] analisaram três algoritmos de *stemming* para o português e apontaram deficiências em todos. Já os lematizadores aparentam ter sido menos avaliados que os *stemmers* na literatura científica. Nieradzik [84] encontrou erros evidentes nas cinco ferramentas que analisou. Portanto, na falta de uma visão clara dos benefícios decorrentes evitou-se aplicar as duas técnicas de normalização, tendo

em conta ainda que elas potencialmente aumentariam em duas vezes a quantidade de variantes de modelos para testar.

Os assuntos também passaram pelo pré-processamento anterior.

4.2.2. Análise de bigramas

Na tentativa de obter melhores resultados com variantes adicionais de bases de tokens, procedeu-se à determinação dos bigramas relevantes. A biblioteca gensim [85] foi escolhida para efetuar a análise com base em contagem de colocações (i.e., ocorrências conjuntas de duas palavras). O gensim treinou um modelo fraseador sobre o corpus formado pelo conjunto inicial de tokens sem o conjunto de teste. Foram considerados bigramas válidos todas as colocações que satisfizeram as seguintes condições simultaneamente:

i. frequência absoluta mínima de 250; e
ii. informação mútua normalizada, calculada por (74), mínima de 0,6.

$$\frac{\ln \frac{p(w_1, w_2)}{p(w_1)\,p(w_2)}}{-\ln p(w_1, w_2)}, \text{ onde } p(w_i) = \frac{\textit{frequência absoluta de } w_i}{n^o \textit{ total de tokens no corpus}} \tag{74}$$

Aplicou-se o modelo treinado nas três bases de análise, gerando três novas bases de tokens/bigramas, nomeadas com o sufixo "-bi". As Tabelas 3 e 4 mostram a estratificação dos documentos por quantidade de tokens/bigramas que os compõem, para as três bases de tokens originais e as três bases de tokens/bigramas, respectivamente. A adição dos assuntos é mais relevante em termos de número de tokens/bigramas para documentos com até 100 tokens/bigramas.

Tabela 3 - Divisão dos documentos das bases de análise por número de tokens

	Base teor		Bases as e as_mod	
Nº de tokens	Nº de documentos	%	Nº de documentos	%
[0, 5)	111	0,26	49	0,12
[5, 10)	62	0,15	48	0,11
[10, 50)	663	1,57	553	1,31
[50, 100)	4.254	10,10	3.990	9,47
[100, 500)	23.641	56,11	23.748	56,37
[500, 1000)	5.846	13,88	6.104	14,49
[1.000, 1.500)	1.756	4,17	1.790	4,25
[1.500, 5.000)	3.026	7,18	3.069	7,28
[5.000, 10.000)	1.168	2,77	1.174	2,79
[10.000, 15.000)	498	1,18	498	1,18
[15.000, 20.000)	339	0,80	339	0,80
[20.000, 50.000)	676	1,60	678	1,61
[50.000, 75.000)	62	0,15	62	0,15
[75.000, 100.000)	21	0,05	21	0,05
[100.000, 200.000)	8	0,02	8	0,02
[200.000, 500.000)	1	0,00	1	0,00
Total	42.132	100,00	42.132	100,00

Tabela 4 - Divisão dos documentos das bases de análise com bigramas por número de tokens/bigramas

	Base teor-bi		Base as-bi		Base as_mod-bi	
Nº de tokens/bigramas	Nº de documentos	%	Nº de documentos	%	Nº de documentos	%
[0, 5)	121	0,29	55	0,13	61	0,14
[5, 10)	54	0,13	46	0,11	48	0,11
[10, 50)	996	2,36	872	2,07	899	2,13
[50, 100)	5.146	12,21	4.947	11,74	4.983	11,83
[100, 500)	23.854	56,62	24.120	57,25	24.075	57,14
[500, 1000)	5.056	12,00	5.175	12,28	5.158	12,24
[1.000, 1.500)	1.569	3,72	1.570	3,73	1.563	3,71
[1.500, 5.000)	2.811	6,67	2.820	6,69	2.821	6,70
[5.000, 10.000)	1.091	2,59	1.093	2,59	1.093	2,59
[10.000, 15.000)	473	1,12	473	1,12	471	1,12
[15.000, 20.000)	362	0,86	365	0,87	364	0,86
[20.000, 50.000)	536	1,27	533	1,27	533	1,27
[50.000, 75.000)	42	0,10	42	0,10	42	0,10
[75.000, 100.000)	14	0,03	14	0,03	14	0,03
[100.000, 200.000)	6	0,01	6	0,01	6	0,01
[200.000, 500.000)	1	0,00	1	0,00	1	0,00
Total	42.132	100,00	42.132	100,00	42.132	100,00

4.2.3. Seleção de atributos

Considerando cada token da base teor como um atributo, o problema apresentaria uma matriz de observações x atributos com dimensão 33.705 x 262.905. Esse tipo de problema de alta dimensionalidade, em que o número de preditores é muito maior que o de observações, está sujeito à alta variância e sobreajuste [40]. Nesse cenário, existem duas saídas recomendáveis: empregar algoritmos de AM modificados para lidar com a alta dimensionalidade ou selecionar atributos. Como nem todos os algoritmos modificados de AM estão implementados em código aberto, optou-se pela segunda alternativa.

Procedeu-se à seleção de atributos combinando os critérios de IM e teste do χ^2. Para se ter máxima robustez, os critérios foram empregados de forma combinada. Primeiramente, foi apurada a matriz de atributos e suas frequências absolutas (i.e., representação BOW) e identificaram-se os tokens/bigramas dos conjuntos de treinamento que apresentaram os 5.000 maiores escores dos dois índices foram reunidos. Selecionaram-se os 2.000 atributos mais bem colocados na soma das posições dos dois *rankings* para que a seleção não fosse agressiva a ponto de deixar muitos documentos com menos de cinco tokens/bigramas. Doravante o critério combinado será referido como IM+ χ^2.

Os tokens e bigramas que integram as vinte primeiras posições para as bases teor, as e suas variantes com bigramas estão elencados na Tabela 5. As diferenças entre cada dupla de listagens são mínimas. Formaram-se bigramas com tokens presentes nas listas das bases anteriores à análise de bigramas, abrindo espaço para novos tokens entrarem no rol de 2.000 atributos selecionados. Os bigramas da tabela são efetivamente expressões corriqueiras no domínio financeiro, porém não possuem significado distinto dos de seus tokens constituintes.

Foram realizadas também seleções de atributos das seis bases de análise por meio da regressão Lasso. A biblioteca glmnet [49], com API Python escrita pela Civis Analytics[21], foi escolhida por ser especializada em modelos lineares generalizados. A regressão Lasso foi executada com a matriz BOW de atributos para tentar resolver o problema de classificação em cada base, com validação

21 https://github.com/civisanalytics/python-glmnet

cruzada *k-fold*, sendo k=5. Foram considerados como selecionados os atributos cujos coeficientes determinados pela regressão eram diferentes de zero. Testaram-se entre 3.000 e 8.000 valores do parâmetro λ, em baterias de 800 a 1.000 valores de λ. A biblioteca gerou sequências lineares de valores testados de λ em escala logarítmica, obedecendo ao hiperparâmetro definidor da razão entre os valores mínimo e máximo igual a 10^{-5} ou 10^{-6}.

Tabela 5 - Vinte tokens/bigramas mais bem colocados no critério combinado de informação mútua e qui-quadrado

Base teor	Base as	Base teor (bigramas)	Base as (bigramas)
incorporação	incorporação	incorporação	incorporação
justificação	protocolo	protocolo_justificação	aumento_capital
protocolo	justificação	capital	protocolo_justificação
laudo	laudo	subscrição	subscrição
ações	ações	laudo_avaliação	laudo_avaliação
subscrição	capital	aumento	ações
avaliação	subscrição	ações	protocolo
capital	avaliação	contábil	patrimônio_líquido
aumento	aumento	patrimônio_líquido	contábil
valor	valor	protocolo	social
patrimônio	patrimônio	social	capital
contábil	contábil	avaliação	avaliação
preferência	preferência	direito_preferência	direito_preferência
sobras	sobras	valor	valor
subscritas	subscritas	sobras	sobras
companhia	companhia	incorporada	subscritas
incorporada	incorporada	subscritas	incorporada
acervo	acervo	ações_ordinárias	companhia
novas	novas	novas	ações_ordinárias
ordinárias	ltda	companhia	novas

Os valores encontrados de λ foram analisados sob duas óticas:

i. a macromédia dos valores da métrica F1[22], para todas as classes, obtidos nos *folds* de validação cruzada sobre o conjunto de validação – referida nesta seção como escore F1; e
ii. o número de documentos que conteriam pelo menos 5 tokens/bigramas se a seleção fosse realizada com o λ em questão.

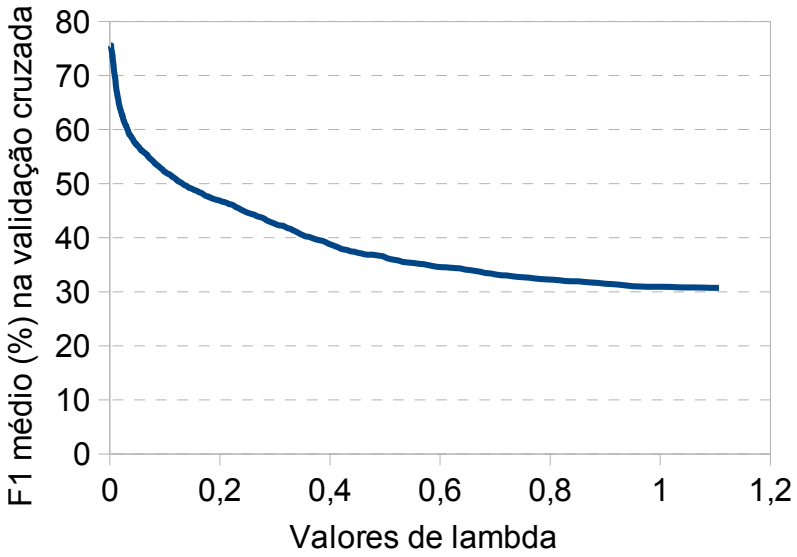
Estabeleceu-se que documentos com menos de cinco tokens/bigramas deveriam eliminados da base de dados antes de rodar os algoritmos, por serem

22 Não se utilizou a métrica preferencial microF2 devido à aparente incompatibilidade da biblioteca com métricas definidas pelo usuário.

curtos demais. Portanto, simular a seleção foi importante para evitar a escolha de um valor de λ que gerasse descarte grande de documentos.

O Gráfico 1 mostra o resultado de uma rodada de validação cruzada com 514 valores testados de λ para a base teor. Como as sequências de valores testados eram pseudoaleatórias e a melhoria no escore F1 era significativa quanto menor fosse λ, foi interessante realizar múltiplas rodadas em busca do λ que proporcionasse o melhor escore e uma quantidade razoável de documentos com no mínimo cinco atributos ao final da seleção. Geralmente o menor λ de convergência do algoritmo satisfazia as duas condições. Rodadas que testaram maior quantidade de valores de λ (isto é, com sequências mais longas) tenderam a produzir a valores de F1 mais elevados, porém consumiam mais memória RAM (32 GB permitiram testar 300 a 1.000 valores de λ).

Gráfico 1 - Evolução do escore médio F1 (macro) em função do lambda testado na seleção de atributos por regressão Lasso (base teor)



A regressão Lasso provou ser bem mais rigorosa que o critério anterior. A regressão selecionou entre 163 e 553 atributos das bases de análise testadas, como se nota na Tabela 6. O descarte de documentos foi inferior a 1%, como se vê na linha "% docs ≥ 5 atributos", a qual informa o percentual de documentos que possuíam pelo menos cinco atributos selecionados. A tabela também traz as médias dos escores F1 dos *folds* de validação cruzada, calculados sobre o conjunto de validação ao final do treinamento do modelo de regressão.

Tabela 6 - Análise da seleção de atributos por regressão Lasso

	Bases					
	teor	as	as_mod	teor-bi	as-bi	as_mod-bi
λ	0,00194	0,00198	0,00692	0,00188	0,00340	0,00190
Nº atributos selecionados	551	533	166	538	343	438
% docs ≥ 5 atributos	99,6	99,8	99,3	99,5	99,7	99,5
F1 médio validação(%)	75,6	76,2	76,9	75,7	77,5	81,7
Desvio-padrão do F1 validação	2,2	1,8	0,81	1,4	0,4	0,36

Aplicar os algoritmos de AM sobre os atributos selecionados pelos critérios IM+χ^2 e Lasso permitirá comparar os desempenhos de dois tipos distintos de seleção de variáveis (*filter* versus *embedded*). Além disso e seguindo o princípio "aposte na esparsidade[23]" [40], a seleção por regressão Lasso permitirá avaliar se modelos estatísticos esparsos (i.e., aqueles onde um número reduzido de atributos são relevantes) são adequados ao problema de classificação.

A seleção de atributos pelo critério IM+χ^2 também acarretou descarte menor que 1% de todas as bases. No entanto, a regressão Lasso se destaca por determinar automaticamente e, portanto, com menor grau de arbitrariedade o número de atributos a serem selecionados.

4.2.4. Filtragem de documentos

Como já descrito, selecionaram-se os documentos das bases de análise que continham no mínimo cinco tokens selecionados pelo procedimento anterior. Bigramas foram contados como dois tokens. Este procedimento gerou pequenas diferenças nas quantidades de documentos remanescentes nas bases (p.ex., em torno de 20 documentos, ou 0,2%, na base de teste).

23 "Use a procedure that does well in sparse problems, since no procedure does well in dense problems." [40]

4.2.5. Representação numérica

Aplicaram-se três tipos de representação numérica sobre as bases de documentos pós-descarte:

1. **Tf-idf** calculada segundo (4), por ser simples e amplamente empregada na literatura com bons resultados.
2. ***Bag-of-words***, para verificar se o uso de frequências de termos por si só é capaz para gerar melhores resultados.
3. **Word2vec**, pelo seu conteúdo semântico e fácil utilização em redes neurais.

A famosa biblioteca Python de AM Scikit-learn [35] efetuou a vetorização dos atributos nas representações tf-idf e *bag-of-words*.

A representação word2vec foi obtida com a biblioteca gensim [85] versão 3.8.3, devido à sua rapidez e gerenciamento eficiente de memória. Deu-se preferência ao método *skip-gram* por apresentar resultados semânticos melhores no artigo original [36]. Primeiramente, o modelo de vetorização foi treinado sobre todas as palavras presentes no corpus composto pelo conjunto inicial sem o conjunto de teste (parâmetro *min_count* igual a um). Os padrões da biblioteca foram mantidos para os demais parâmetros do modelo[24]. Dessa forma, produziram-se:

- três modelos treinados, capazes de representar tokens em vetores com dimensões 100, 200 e 300; e
- outros três modelos para representar bigramas e tokens presentes no mesmo documento, nas três dimensões anteriores.

O passo seguinte consistiu na conversão dos tokens/bigramas dos conjuntos de treinamento e teste em vetores word2vec pelos modelos treinados. Como os algoritmos clássicos necessitam que o documento se torne um único vetor, este foi obtido pela média dos vetores dos tokens constituintes, ponderada pelos seus valores tf-idf.

24 Janela de predição do *skip-gram* = 5 tokens, taxa de aprendizado inicial = 0,025, taxa de aprendizado final = 10^{-4}, *negative sampling* de 5 tokens a terem seus pesos atualizados, expoente de *negative sampling* = 0,75 e nº de épocas de treinamento = 5.

4.3. Classificação via algoritmos clássicos

Realizaram-se três experimentos para testar os 12 algoritmos clássicos de AM, descritos na Seção 3.2.4 e listados na Tabela 7, na tarefa de classificação dos documentos das bases de análise.

4.3.1. Desenho do 1º experimento: aplicação individual de algoritmos sem subamostragem

O primeiro experimento estudou a aplicação dos 12 algoritmos clássicos da Tabela 7 de forma **individual**, para classificar bases de documentos **sem subamostragem**.

Tabela 7 - Algoritmos testados no 1º experimento de classificação de documentos

Tipo	Algoritmo
SVM	SVM kernel radial
	SVM kernel polinomial
Naive Bayes	Naive Bayes multinomial
	Naive Bayes complementar
Baseados em distância	*K-Nearest Neighbors*
	Nearest Shrunken Centroids
Análises discriminantes	Análise Discriminante Linear (LDA)
	Análise Discriminante Quadrática (QDA)
Baseados em árvores de decisão	*Random Forest*
	Árvores *boosted*
Regressões	Regressão Logística[25]
	Regressão Elastic Net

O teste dos algoritmos foi realizado conforme as seguintes características:

- Para a regressão Elastic net, testaram-se valores do parâmetro α entre 0 (Lasso) e 1 (Ridge), sendo α = 0; 0,1; 0,2;…;1. A regressão foi executada sobre bases sem seleção prévia de variáveis, para se testar a efetividade de um algoritmo de forte regularização em uma matriz esparsa de 260 mil atributos. Optou-se pela biblioteca glmnet [49] para treinar o algoritmo.

25 Sem regularização.

- As árvores *boosted* foram testadas com o uso da biblioteca XGBoost [86] (*Extreme Gradient Boosting*), devido à sua rápida execução e suporte a unidades de processamento gráfico (GPU).
- Os demais algoritmos foram executados com a biblioteca Scikit-learn [35].
- Para todos os algoritmos, o método de validação cruzada *k-fold* com k=5 determinou a melhor configuração de hiperparâmetros. O modelo de maior escore microF2 (seção 4.3.1.1) no conjunto de validação foi retreinado no conjunto de treinamento inteiro (mantidos os hiperparâmetros), para efetuar a predição no conjunto de teste que mensurou o desempenho do algoritmo.
- A validação cruzada utilizou o método de pesquisa exaustiva em grade (*grid search*), que testou todas as combinações de valores pré-definidos de hiperparâmetros em cada *fold* do conjunto de validação. Devido à grande quantidade de hiperparâmetros disponíveis, efetuou-se por etapas a pesquisa realizada para os algoritmos baseados em árvores. Um grupo de 3 a 5 hiperparâmetros foi testado em cada etapa. No caso das árvores *boosted*, a pesquisa englobou 8 etapas de *grid search* e 2 etapas de pesquisa aleatória (*random search*), necessárias para limitar a quantidade de iterações e manter o tempo de execução dentro de um limite razoável.
- Foram experimentados os dois critérios de seleção de atributos e os três tipos de representação numérica citados na seção anterior.
- A representação Word2vec também foi aplicada em bases sem seleção de atributos, para testar sua eficácia em reduzir alta dimensionalidade.
- Estabeleceu-se um *baseline* de 26,9% para microF2 por uma alocação simples de todos documentos na classe “b” (mais numerosa que a “a”).
- Estipulou-se outro *baseline* de 85,4% para a acurácia global, obtido ao classificar todos os documentos na categoria “n”.

O Apêndice 2 descreve os hiperparâmetros e as faixas de valores experimentados em cada algoritmo, tendo sido utilizados os valores padrões das bibliotecas para os demais hiperparâmetros.

A Tabela 8 traz todas as combinações testadas de algoritmos, métodos de seleção de atributos e de representação. As combinações dividem-se em 7 blocos, que diferem quanto às bases utilizadas ou forma de cálculo do Word2vec.

Tabela 8 - Combinações testadas no 1º experimento de classificação de documentos

Algoritmo	Seleção de atributos	Representação
SVM kernel radial	IM + χ^2	Tf-idf
SVM kernel polinomial	IM + χ2	Tf-idf
Naive Bayes multinomial	IM + χ2	Tf-idf
Naive Bayes complementar	IM + χ2	Tf-idf
K-Nearest Neighbors	IM + χ2	Tf-idf
Nearest Shrunken Centroids	IM + χ2	Tf-idf
LDA	IM + χ2	Tf-idf
QDA	IM + χ2	Tf-idf
Random Forest	IM + χ2	Tf-idf
Árvores boosted	IM + χ2	Tf-idf
Regressão Logística[26]	IM + χ2	Tf-idf
Regressão Elastic net	IM + χ2	Tf-idf
Regressão Elastic net	-	Tf-idf
Regressão Elastic net	-	BOW
SVM kernel radial	IM + χ2	Word2vec 300
SVM kernel polinomial	IM + χ2	Word2vec 300
Naive Bayes multinomial	IM + χ2	Word2vec 300
Naive Bayes complementar	IM + χ2	Word2vec 300
K-Nearest Neighbors	IM + χ2	Word2vec 300
Nearest Shrunken Centroids	IM + χ2	Word2vec 300
LDA	IM + χ2	Word2vec 300
QDA	IM + χ2	Word2vec 300
Random Forest	IM + χ2	Word2vec 300
Árvores boosted	IM + χ2	Word2vec 300
Regressão Logística[26]	IM + χ2	Word2vec 300
Regressão Elastic net	-	Word2vec 300
SVM kernel radial*	-	Word2vec 100
SVM kernel polinomial*	-	Word2vec 100
Naive Bayes multinomial*	-	Word2vec 100
Naive Bayes complementar*	-	Word2vec 100
K-Nearest Neighbors*	-	Word2vec 100
Nearest Shrunken Centroids*	-	Word2vec 100
LDA*	-	Word2vec 100
QDA*	-	Word2vec 100
Random Forest*	-	Word2vec 100
Árvores boosted*	-	Word2vec 100
Regressão Logística[26]*	-	Word2vec 100
Regressão Elastic net*	-	Word2vec 100

* Aplicados somente sobre as bases as_mod e as_mod*.

26 Sem regularização.

Tabela 8 - Combinações testadas no 1º experimento realizado de documentos (cont.)

Algoritmo	Seleção de atributos	Representação
SVM kernel radial*	-	Word2vec 200
SVM kernel polinomial*	-	Word2vec 200
Naive Bayes multinomial*	-	Word2vec 200
Naive Bayes complementar*	-	Word2vec 200
K-Nearest Neighbors*	-	Word2vec 200
Nearest Shrunken Centroids*	-	Word2vec 200
LDA*	-	Word2vec 200
QDA*	-	Word2vec 200
Random Forest*	-	Word2vec 200
Árvores boosted*	-	Word2vec 200
Regressão Logística[26]*	-	Word2vec 200
Regressão Elastic net*	-	Word2vec 200
SVM kernel radial*	-	Word2vec 300
SVM kernel polinomial*	-	Word2vec 300
Naive Bayes multinomial*	-	Word2vec 300
Naive Bayes complementar*	-	Word2vec 300
K-Nearest Neighbors*	-	Word2vec 300
Nearest Shrunken Centroids*	-	Word2vec 300
LDA*	-	Word2vec 300
QDA*	-	Word2vec 300
Random Forest*	-	Word2vec 300
Árvores boosted*	-	Word2vec 300
Regressão Logística[26]*	-	Word2vec 300
Regressão Elastic net*	-	Word2vec 300
SVM kernel radial*	-	Word2vec 300**
SVM kernel polinomial*	-	Word2vec 300**
Naive Bayes multinomial*	-	Word2vec 300**
Naive Bayes complementar*	-	Word2vec 300**
K-Nearest Neighbors*	-	Word2vec 300**
Nearest Shrunken Centroids*	-	Word2vec 300**
LDA*	-	Word2vec 300**
QDA*	-	Word2vec 300**
Random Forest*	-	Word2vec 300**
Árvores boosted*	-	Word2vec 300**
Regressão Logística[26]*	-	Word2vec 300**
Regressão Elastic net*	-	Word2vec 300**

* Aplicados somente sobre as bases as_mod e as_mod*.

** Word2vec determinado pelo produto da matriz de *embeddings* pela matriz de todos os atributos representados por BOW. Nos quatro blocos anteriores de combinações calculou-se o Word2vec por meio do produto da matriz de *embeddings* pela de todos os atributos representados por Tf-Idf.

Tabela 8 - Combinações testadas no 1º experimento de classificação de documentos (cont.)

Algoritmo	Seleção de atributos	Representação
SVM kernel radial	Lasso	Tf-idf
SVM kernel polinomial	Lasso	Tf-idf
Naive Bayes multinomial	Lasso	Tf-idf
Naive Bayes complementar	Lasso	Tf-idf
K-Nearest Neighbors	Lasso	Tf-idf
Nearest Shrunken Centroids	Lasso	Tf-idf
LDA	Lasso	Tf-idf
QDA	Lasso	Tf-idf
Random Forest	Lasso	Tf-idf
Árvores boosted	Lasso	Tf-idf
Regressão Logística[26]	Lasso	Tf-idf
Regressão Elastic net	Lasso	Tf-idf
Regressão Elastic net	Lasso	BOW

O experimento aplicou cada combinação da Tabela 8 em oito bases de análise montadas a partir das três bases de tokens apresentadas na seção 4.2.1. A exceção fica por conta de quatro blocos (3º ao 6º) de combinações executadas sem seleção de atributos e com representação Word2vec, as quais foram aplicadas somente em duas bases[27]. As composições das oito bases de análise são sintetizadas na Tabela 9 e são produto da combinação de três características:

1. adoção de bigramas;
2. incorporação de assuntos originais ou modificados (vide seção 4.1); e
3. incorporação de três metadados relativos às categorias, tipos e espécies dos documentos, representados por três atributos categóricos convertidos em 47 variáveis binárias. Devido ao seu reduzido número, os atributos categóricos não passaram por processo de seleção.

Tabela 9 - Bases testadas no primeiro experimento

Nome da base	Unigramas/Bigramas	Assuntos	Metadados
teor	Unigramas	-	-
teor-bi	Bigramas	-	-
as*	Unigramas	Originais	-
as	Unigramas	Originais	Sim
as-bi	Bigramas	Originais	Sim
as_mod*	Unigramas	Modificados	-
as_mod	Unigramas	Modificados	Sim
as_mod-bi	Bigramas	Modificados	Sim

27 Bases as_mod* e as_mod listadas na Tabela 9.

4.3.1.1.
Métricas de desempenho de classificação

A definição das métricas para medir o desempenho dos algoritmos no três experimentos de classificação seguiu o seguinte racional:

- Embora seja de mensuração complexa, sabe-se que o custo de não detectar um documento com informações sobre operações societárias ou aumento de capital (classes minoritárias a e b) é mais alto que o custo de recuperar um falso positivo (documento irrelevante), pois a falta de um documento importante prejudica a supervisão desses eventos. Portanto, o *recall* relativo às duas classes seria uma possível escolha de métrica.
- Ao mesmo tempo, é preciso também levar em conta a precisão do modelo, já que recuperar muitos falsos positivos elevará o custo da supervisão ao desperdiçar tempo dos analistas da CVM com documentos irrelevantes.
- A métrica F2 captura apropriadamente esse compromisso, dado que ela confere peso 2 ao *recall* na média harmônica entre este e a precisão. Selecionou-se então o escore F2 formado pelas micromédias da precisão e *recall* (doravante **microF2**) relativas às duas classes a e b como métrica principal para aferir o desempenho dos algoritmos de AM no problema de classificação, por considerar as proporções de documentos de cada classe. MicroF2 é computado conforme (18).
- Adotar escore F2 único que reúne as classes a e b facilita a comparação de algoritmos e prioriza a recuperação de documentos dessas classes, dando menor ênfase a acertar a classe em si.
- As métricas secundárias e subsidiariamente discutidas são a microF1, que confere peso igual ao *recall* e à precisão, e a acurácia global.

4.3.2.
1º experimento: resultados, análise e discussão

A Tabela 10 traz os algoritmos com os maiores escores microF2 no conjunto de teste, para cada bloco de combinações da Tabela 8 e base de análise da Tabela 9. Por exemplo, o SVM radial da 1ª linha da Tabela 10 é o algoritmo de maior

microF2 no conjunto de teste, dentre todos aqueles aplicados na base teor com atributos selecionados pelo critério IM + χ^2 e com representação Tf-idf, incluindo duas regressões Elastic Net com todos os atributos (1º bloco da Tabela 8).

Pela tabela, observa-se que:

- As árvores *boosted* e o SVM predominaram (ambos modelos não lineares), além da regressão Elastic net aplicada nas duas bases as_mod e as_mod*.
- A representação Tf-idf proporcionou os melhores resultados, enquanto os vetores de dimensão 300 calculados pelo Word2vec possivelmente não possuem conteúdo semântico equivalente para classificar as duas classes de interesse com a mesma eficácia, visto que a piora dos valores de *recall* relativos ao Word2vec é maior que a diminuição da acurácia. Reduzir as dimensões para 200 ou 100 também implicou queda de desempenho.
- A regressão Lasso pode oferecer um bom compromisso entre performance e custo computacional para a seleção de atributos, quando comparada ao critério combinado IM+χ^2: a quantidade menor de atributos selecionados diminui consideravelmente o tempo gasto na pesquisa de hiperparâmetros ao custo de resultados levemente inferiores.
- Todos os modelos ofereceram performance superior aos *baselines* propostos.
- O melhor escore microF2 obtido de 94,1% já seria aceitável para motivar a implementação em regime de produção do modelo SVM com kernel radial, aplicado à base de assuntos modificados sem metadados (as_mod*). O escore microF1 desse melhor modelo é praticamente igual ao microF2, demostrando que há equilíbrio entre precisão e *recall*, assim como uma quantidade aceitável de falsos positivos.
- A regressão Elastic net aplicada à base de assuntos modificados com metadados (as_mod) é uma segunda opção para implementação, com microF2 muito próximo ao melhor escore.

Os escores microF2 de todos os algoritmos executados no primeiro experimento estão mostrados graficamente no Apêndice 4.

Tabela 10 - Resultados dos modelos clássicos individuais de maior escore microF2 (%) – 1º experimento

Base	Melhor modelo	Atribs.	Emb.	MicroF2	Ac.	MicroF1	MicroR
Seleção de atributos pelo critério combinado Informação Mútua + χ^2							
teor	SVM radial	2000	Tf-idf	92,7	98,0	93,1	92,4
teor-bi	Árvores boosted	2000	Tf-idf	92,0	97,8	92,4	92,4
as*	SVM radial	2000	Tf-idf	93,9	98,3	94,0	93,4
as	Árvores boosted	2047	Tf-idf	93,6	89,2	93,7	93,8
as-bi	Árvores boosted	2047	Tf-idf	93,3	98,1	93,4	93,2
as_mod*	SVM radial	2000	Tf-idf	**94,1**	**98,4**	94,3	**94,0**
as_mod	Elastic net	Todos**	Tf-idf	94,0	**98,4**	**94,4**	93,8
as_mod-bi	Elastic net	Todos**	Tf-idf	93,7	98,3	94,0	93,5
teor	SVM radial	2000	w2v300	89,4	97,3	90,6	88,5
teor-bi	SVM radial	2000	w2v300	86,2	96,9	88,7	84,6
as*	SVM radial	2000	w2v300	90,5	97,5	91,2	90,0
as	SVM polinomial	2047	w2v300	87,4	96,8	88,7	86,6
as-bi	SVM polinomial	2047	w2v300	87,8	96,8	88,8	87,1
as_mod*	SVM radial	2000	w2v300	90,7	97,5	91,3	90,3
as_mod	SVM radial	2047	w2v300	89,0	97,3	90,4	88,2
as_mod-bi	SVM radial	2047	w2v300	89,3	97,3	90,4	88,7
Word2vec como redutor de dimensionalidade sobre todos os atributos originais							
as_mod	SVM radial	Todos***	w2v300	78,5	95,3	82,2	76,1
as_mod*	SVM radial	Todos***	w2v300	78,6	95,4	82,8	76,1
as_mod	SVM radial	Todos***	w2v200	76,3	94,8	80,1	73,9
as_mod*	SVM radial	Todos***	w2v200	76,0	94,7	80,0	73,6
as_mod	SVM radial	Todos***	w2v100	65,3	92,5	70,0	62,6
as_mod*	QDA	Todos***	w2v100	66,5	88,5	62,6	57,2
as_mod	Árvores boosted	Todos****	w2v300	81,5	95,8	84,7	79,5
as_mod*	Árvores boosted	Todos****	w2v300	80,4	95,6	83,7	78,4
Seleção de atributos por regressão Lasso							
teor	SVM radial	551	Tf-idf	92,9	98,0	92,9	92,8
teor-bi	SVM polinomial	538	Tf-idf	93,2	98,1	93,3	93,7
as	Árvores boosted	533	Tf-idf	93,0	98,0	93,1	93,0
as-bi	SVM radial	343	Tf-idf	92,6	97,9	92,8	91,7
as_mod*	SVM radial	166	Tf-idf	93,1	98,0	92,9	93,3
as_mod	Árvores boosted	166	Tf-idf	92,9	98,0	93,1	92,8
as_mod-bi	SVM radial	438	Tf-idf	92,6	97,9	92,8	92,5
Baselines				26,9	85,3		

* Bases sem metadados de documentos.
** As bases as_mod e as_mod-bi possuem 273.987 e 285.521 atributos no total, respectivamente.
*** Vetores dos documentos calculados por Word2vec ponderado pelos índices Tf-idf.
**** Vetores dos documentos calculados por Word2vec ponderado por *bag-of-words*.
Atribs.: Nº de atributos do modelo
Emb.: Tipo de representação (*embedding*)
Ac.: Acurácia global
MicroR: *Recall* micromédio
Escores microF2, microF1 e microR calculados para as classes “a” e “b”.

Para examinar o efeito do emprego de bigramas e da adição dos assuntos e metadados dos documentos, foram reunidos os escores microF2 dos algoritmos SVM radial, SVM polinomial e árvores *boosted* no experimento com o critério IM+χ^2 e representação Tf-idf. Selecionaram-se esses algoritmos por proverem os escores mais elevados e sem *outliers*. As médias das variações de microF2 dos algoritmos, atribuídas às alterações de composição da base de documentos e uso de bigramas, são mostradas na Tabela 11.

Tabela 11 - Variação média de microF2 atribuída a alterações de bases de dados no 1º experimento

Alteração	Variação média de microF2 (%)
Adoção de bigramas na base teor	-1,49
Adoção de bigramas na base as	-0,19
Adoção de bigramas na base as_mod	-0,06
Inclusão de assuntos originais	0,78
Inclusão de assuntos originais e metadados	-0,87
Inclusão de assuntos modificados	0,90
Inclusão de assuntos modificados e metadados	-0,83

Pode-se tecer as seguintes observações:

- Os bigramas não contribuíram para classificar melhor os documentos das classes de interesse e, mais ainda, o seu emprego acarretou redução média de microF2 entre 0,06 e 1,5 p.p. Uma hipótese plausível é ter ocorrido perda de poder de predição de uma palavra quando as suas ocorrências foram substituídas pelos diversos bigramas gerados a partir dela.
- No tocante às bases de dados, agregar os assuntos dos documentos aos seus conteúdos proporcionou aumento médio de 0,8 p.p. no escore microF2 e vai ao encontro da intuição de que os assuntos sintetizam as informações mais relevantes dos documentos.
- A inclusão de metadados dos documentos conduziu a uma variação negativa de monta similar.

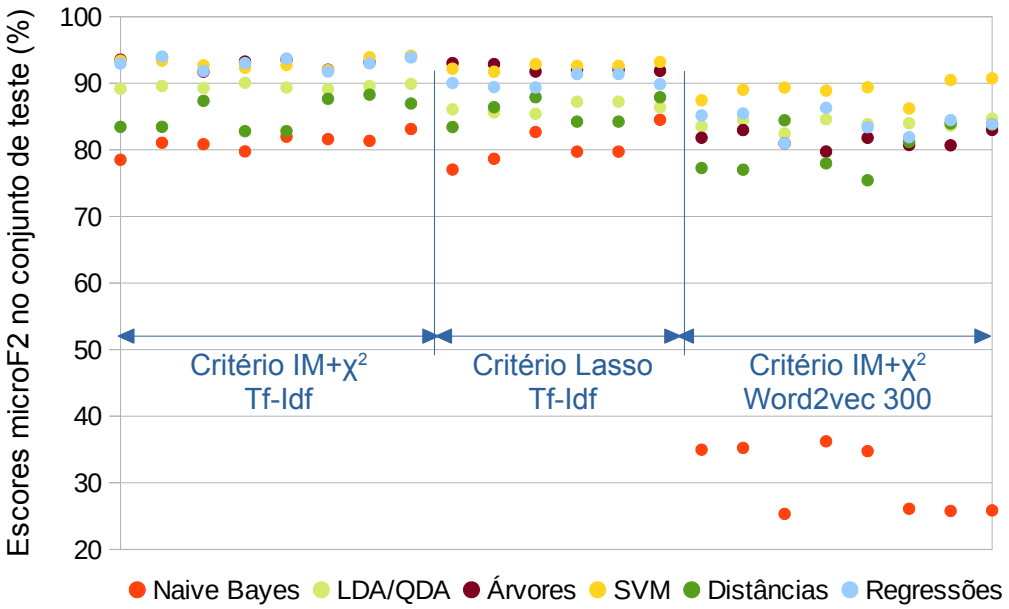
O Gráfico 2 compara os resultados alcançados pelos diferentes tipos de algoritmos, variando-se o critério de seleção e o método de representação de atributos. Foram selecionados os algoritmos com maior escore microF2 de cada tipo. O gráfico reúne os algoritmos executados sobre:

i. bases com atributos selecionados pelo critério IM+χ^2, representados por índices Tf-idf;
ii. bases com atributos selecionados pelo critério IM+χ^2 e representados por vetores Word2vec com 300 dimensões; e
iii. bases com atributos selecionados pela regressão Lasso e com representação Tf-idf.

Observa-se, pelo gráfico, que:

- O SVM está consistentemente entre os algoritmos de melhor desempenho.
- As regressões lineares vêm em seguida, cabendo ressaltar que elas foram mais sensíveis à redução de dimensionalidade imposta pelo Lasso e Word2vec.
- Os algoritmos baseados em árvores e no Naive Bayes também sofreram com o Word2vec de 300 dimensões.

Gráfico 2 - Maiores escores microF2 do 1º experimento por tipo de algoritmo



O tipo árvores compreende *Random Forest* e árvores boosted.
O tipo distâncias compreende KNN e *Nearest Shrunken Centroids*.

Os resultados de todos os testes do primeiro experimento estão apresentados no Apêndice 5.

4.3.3.
Desenho do 2º experimento: aplicação individual de algoritmos com subamostragem

O segundo experimento buscou melhorar os resultados obtidos em duas bases selecionadas do primeiro experimento, com o emprego de seis técnicas de subamostragem. As características do experimento são:

- Todas as técnicas de subamostragem eliminaram observações da classe dominante n para balancear o conjunto de treinamento.
- Treinaram-se os mesmos algoritmos do primeiro experimento sobre as bases subamostradas, repetindo-se os procedimentos anteriores de pesquisa de hiperparâmetros e validação cruzada.
- A Tabela 12 lista as combinações testadas de algoritmos, métodos de seleção de atributos e representação numérica.
- As combinações foram experimentadas sobre as duas bases de tokens com assuntos modificados as_mod e as_mod*, escolhidas por proporcionarem os melhores escores microF2 no primeiro experimento.

Tabela 12 - Combinações testadas no 2º experimento de classificação de documentos

Algoritmo	Seleção de atributos	Representação
SVM kernel radial	IM + χ^2	Tf-idf
SVM kernel polinomial	IM + χ2	Tf-idf
Naive Bayes multinomial	IM + χ2	Tf-idf
Naive Bayes complementar	IM + χ2	Tf-idf
K-Nearest Neighbors	IM + χ2	Tf-idf
Nearest Shrunken Centroids	IM + χ2	Tf-idf
LDA	IM + χ2	Tf-idf
QDA	IM + χ2	Tf-idf
Random Forest	IM + χ2	Tf-idf
Árvores boosted	IM + χ2	Tf-idf
Regressão Logística[28]	IM + χ2	Tf-idf
Regressão Elastic net	IM + χ2	Tf-idf
Regressão Elastic net	-	Tf-idf
Regressão Elastic net	-	BOW

28 Sem regularização.

4.3.4.
2º experimento: resultados, análise e discussão

A Tabela 13 apresenta os resultados correspondentes aos modelos de maior escore microF2 por técnica de subamostragem e base de dados. A tabela também traz os maiores escores microF2 e acurácia do 1º experimento (linhas intituladas "sem subamostragem"), para facilitar a comparação.

Tabela 13 - Resultados dos modelos clássicos individuais de maior escore microF2 (%), com subamostragem – 2º experimento

Método	Melhor modelo	n final	Atribs.	MicroF2	Ac.	MicroF1	MicroR
Base com assuntos modificados sem metadados (as_mod*)							
NearMiss-1	Elastic net	6.971	Todos	77,7	80,4	59,4	**97,9**
NearMiss-2	Elastic net	6.971	Todos	92,6	95,3	85,9	97,7
NearMiss-3	Elastic net	6.971	Todos	92,8	95,7	86,6	97,5
OSS	Árvores boosted	23.226	2000	**94,5**	98,2	93,9	94,9
Tomek Links	Elastic net	33.459	Todos	**94,5**	**98,4**	**94,6**	94,3
CNN	Elastic net	6.905	Todos	94,3	97,0	90,4	97,2
Sem subamostragem	Elastic net	33.667	Todos	93,9	98,4	94,3	93,6
Sem subamostragem	Árvores boosted	33.591	2000	94,0	98,4	94,3	93,8
Sem subamostragem	SVM radial	33.591	2000	94,1	98,4	94,3	94,0
Base com assuntos modificados e metadados (as_mod)							
NearMiss-1	Elastic net	6.971	Todos	79,3	82,3	61,7	**97,9**
NearMiss-2	Elastic net	6.971	Todos	92,6	95,4	86,0	97,6
NearMiss-3	LDA	6.906	2000	92,6	96,4	88,4	95,7
OSS	SVM radial	23.226	2000	**94,3**	**98,3**	94,0	94,6
Tomek Links	Elastic net	33.459	Todos	94,1	**98,3**	**94,1**	94,1
CNN	Elastic net	6.905	Todos	**94,3**	97,0	90,4	97,2
Sem subamostragem	Elastic net	33.667	Todos	94,0	98,4	94,4	93,8
Sem subamostragem	SVM radial	33.667	2000	93,4	98,2	93,5	93,3

OSS: One-Sided Selection
CNN: Condensed Nearest Neighbor
n final: Nº totalde documentos pós-subamostragem
Atribs.: Nº de atributos do modelo
Ac.: Acurácia global
MicroR: *Recall* micromédio
Escores microF2, microF1 e microR calculados para as classes "a" e "b". Os melhores escores do primeiro experimento estão sublinhados.

A tabela mostra que:

- A subamostragem se revelou um procedimento útil para alcançar escores microF2 0,2 a 0,4 p.p. superiores aos melhores valores do experimento anterior, os quais já se encontravam em patamar elevado.

- Os três métodos NearMiss foram provavelmente afetados por ruídos (como os documentos das classes a e b rodeados de documentos da classe n, mostrados na Figura 15), terminando por eliminar documentos da classe n próximos a eles e ampliar demais as regiões de a e b no hiperespaço. Assim, obteve-se alto *recall* porém muitos documentos irrelevantes foram atribuídos às duas classes, resultando em pior desempenho – especialmente no caso do NearMiss-1.
- A subamostragem por *Condensed Nearest Neighbor* também é sensível a ruídos e eliminou uma quantidade similar de documentos que a NearMiss, equivalente a 80% do total da base de treinamento (tornando-a até menor que o conjunto de teste).
- A seleção do CNN foi mais efetiva que as dos NearMiss, embora tenha demandado custo computacional bem maior.
- As duas técnicas baseadas em subamostragem por eliminação de links de Tomek propiciaram melhor equilíbrio entre precisão e *recall* nas classes de interesse. A aplicação pura do método foi parcimoniosa ao eliminar menos de 1% dos documentos classe "n", o que permitiu determinar uma fronteira de decisão mais efetiva. O método *Tomek Links* melhorou o desempenho global da regressão Elastic net para a base as_mod*, enquanto os demais métodos levaram a valores mais elevados de *recall*, em detrimento da precisão (refletido no menor microF1).
- O OSS é o único método que favoreceu algoritmos não lineares quando aplicados às duas bases de documentos.

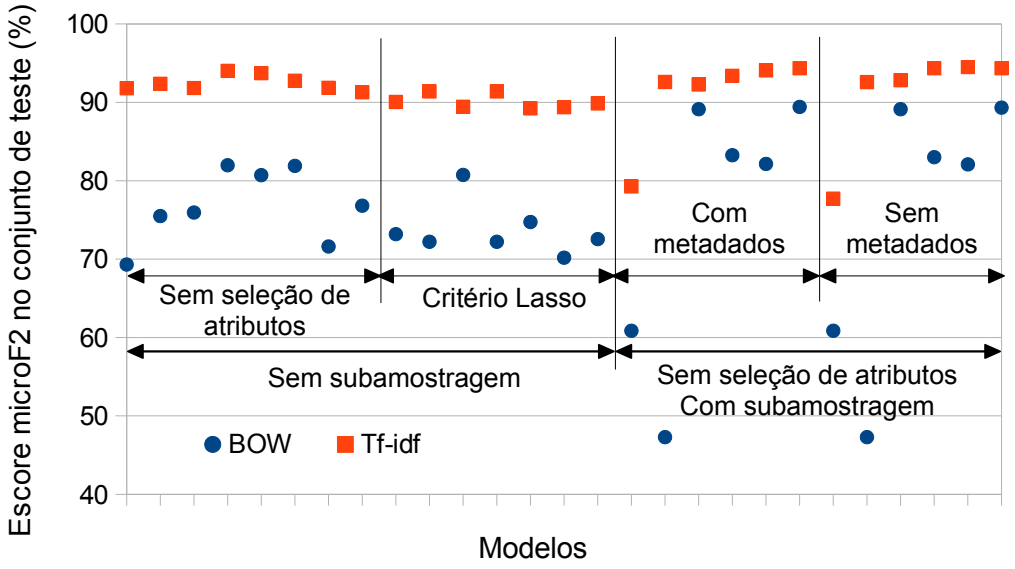
O Gráfico 3 compara a performance das representações BOW e Tf-idf, por meio dos escores microF2 obtidos pela regressão Elastic net em diferentes combinações de bases, métodos de seleção e representação de atributos. O gráfico mostra os escores obtidos nos primeiro e segundo experimentos com uso de:

i. 8 bases sem seleção de atributos e sem subamostragem;
ii. 7 bases com atributos selecionados pelo critério Lasso, sem subamostragem;
iii. base as_mod sem seleção de atributos e com subamostragem; e
iv. base as_mod* sem seleção de atributos e com subamostragem.

Cada ponto do eixo das abcissas do gráfico se refere a um modelo treinado:

- sobre uma das bases sem subamostragem descritas nos itens i e ii; ou
- com aplicação de uma das seis técnicas de subamostragem anteriores, sobre a base as_mod ou as_mod* (itens iii e iv).

Gráfico 3 - Escores microF2 obtidos pelos modelos de regressão Elastic net nos 1º e 2º experimentos



Nota-se que a regressão Elastic net se saiu melhor com tokens vetorizados pelo método tf-idf do que pelo *Bag-of-Words*, em todas as combinações. Portanto, é possível intuir que os documentos possuem tokens muito frequentes necessitam ser ponderados para não prejudicar a classificação.

O Apêndice 6 contém os resultados de todos os algoritmos testados com subamostragem de bases.

4.3.5. *Ensemble*: desenho do 3º experimento

Uma vez tendo testado diversos classificadores individuais nos experimentos anteriores, o terceiro experimento estudou a sua combinação por meio de dois tipos de *ensembles*: votação e *stacking*. A preparação dos dados processados pelos *ensembles* obedeceu às seguintes etapas:

- Selecionaram-se oito bases de documentos com e sem subamostragem, nas quais foram obtidos os maiores escores microF2 nos dois experimentos anteriores.
- As combinações algoritmo/seleção/representação das Tabelas 8 e 12, relativas a cada base, foram retreinadas por validação cruzada 10-*fold* no conjunto de treinamento, mantendo-se as configurações ótimas de hiperparâmetros determinadas pelos experimentos anteriores.
- Os modelos retreinados realizaram predições de classes e probabilidades de classes para o conjunto de validação de cada *fold*. A única exceção é o *Nearest Shrunken Centroids*, para o qual a biblioteca Scikit-learn disponibiliza somente predições de classes. Essas predições de 10 *folds* por modelo formaram o conjunto de treinamento nível 1, utilizado para treinar e selecionar os melhores comitês.
- Os modelos foram treinados em todo o conjunto de treinamento original (nível 0) e efetuaram predições de classes e probabilidades de classes para o conjunto de teste nível 0, as quais compuseram o conjunto de teste nível 1.

Os conjuntos de treinamento e teste nível 1 para os comitês foram extraídos a partir das oito bases de documentos a seguir:

- Base teor com bigramas, sem subamostragem
- Base com assuntos originais, sem metadados, sem subamostragem
- Base com assuntos modificados, sem metadados, sem subamostragem
- Base com assuntos modificados, metadados e subamostragem *Tomek Links*
- Base com assuntos modificados, metadados e subamostragem CNN
- Base com assuntos modificados, metadados e subamostragem OSS
- Base com assuntos modificados, sem metadados e com subamostragem TL
- Base com assuntos modificados, sem metadados e com subamostragem OSS

4.3.5.1. Votação

Testaram-se duas modalidades de votação: plural (*hard*) e por probabilidade (*soft*), ambas com ponderação, segundo o procedimento a seguir.

1. Primeiramente ordenaram-se todos os modelos treinados por ordem decrescente de escore microF2 no conjunto de teste. Foram experimentados todos os comitês possíveis formados pelo modelo de maior escore (doravante chamado de principal) e os demais, tanto em número de modelos – mínimo de três, incluindo o principal – como em composição do comitê.
2. Em seguida, o segundo melhor modelo passou a ser o principal e testaram-se todos os comitês possíveis compostos por ele e os de escore inferior, e assim por diante.
3. Todos os comitês votaram com as predições dos 10 *folds* do conjunto de treinamento nível 1. Foram escolhidos os cem comitês com as maiores médias de escores microF2 nos 10 *folds*, para realizarem predições no conjunto de teste nível 1 e terem seu desempenho aferido.
4. Testaram-se duas distribuições de pesos:

    4.1. Todos os pesos iguais a um. A classe de cada observação é determinada por (58), para a modalidade *hard*, ou (60), para a *soft*.

    4.2. Peso do modelo principal igual ao número de modelos do comitê menos dois, utilizando-se no mínimo três modelos. O resultado da votação *hard* é dado por (59) e o da *soft*, por (61). Na modalidade *hard*, o modelo principal será vencido se os demais modelos forem unânimes.

4.3.5.2. *Stacking*

O processamento do comitê tipo *stacking* seguiu as seguintes etapas:

- Montagem dos conjuntos de treinamento e teste nível 1 a partir das predições realizadas por **todos** os modelos nível 0 dos experimentos anteriores.
- Treinamento do metamodelo e determinação da configuração ótima de hiperparâmetros por validação cruzada 10-*fold*, com o conjunto de treinamento nível 1. Testaram-se três algoritmos como metamodelo:

1. Regressão logística.
2. KNN.
3. SVM com kernel radial.

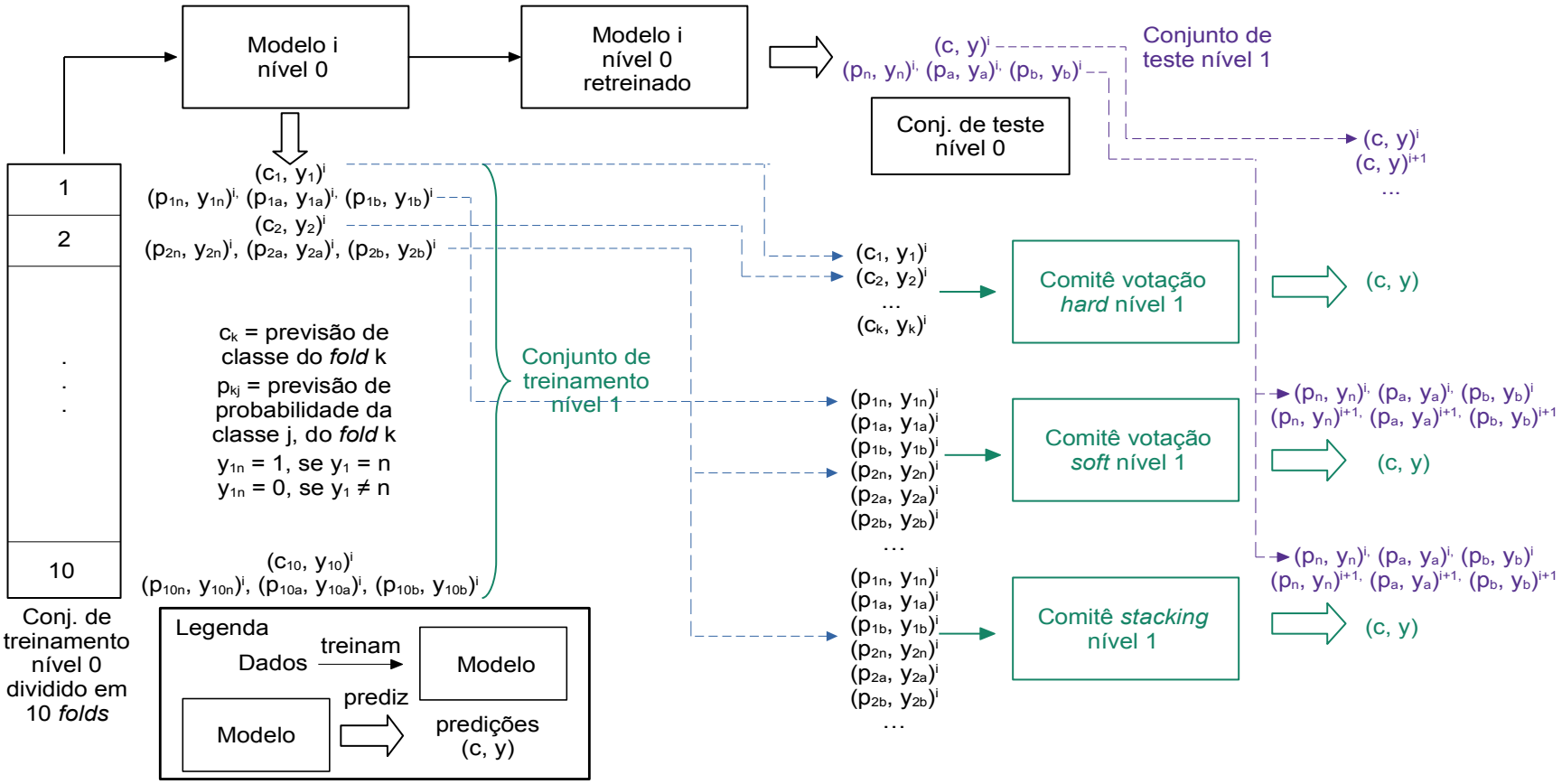
- Predições de probabilidades no conjunto de teste nível 1 pelo metamodelo treinado.

Duas modalidades de *stacking* foram experimentadas, ambas descritas na seção 3.2.5:

1. Metodologia proposta por Ting e Witten [60]. Aqui os metamodelos executaram uma classificação multiclasse em lugar de uma regressão linear para cada classe, como relatado no artigo. Para evitar a colinearidade, as predições de probabilidades da classe "b" foram descartadas para o treinamento do metamodelo.
2. *StackingC* de Seewald [61].

A Figura 17 traz uma representação esquemática da formação dos conjuntos de dados nível 1 e suas utilizações nos dois tipos de *ensembles*.

Figura 17 - Representação esquemática dos comitês



4.3.6.
Ensemble: resultados, análise e discussão

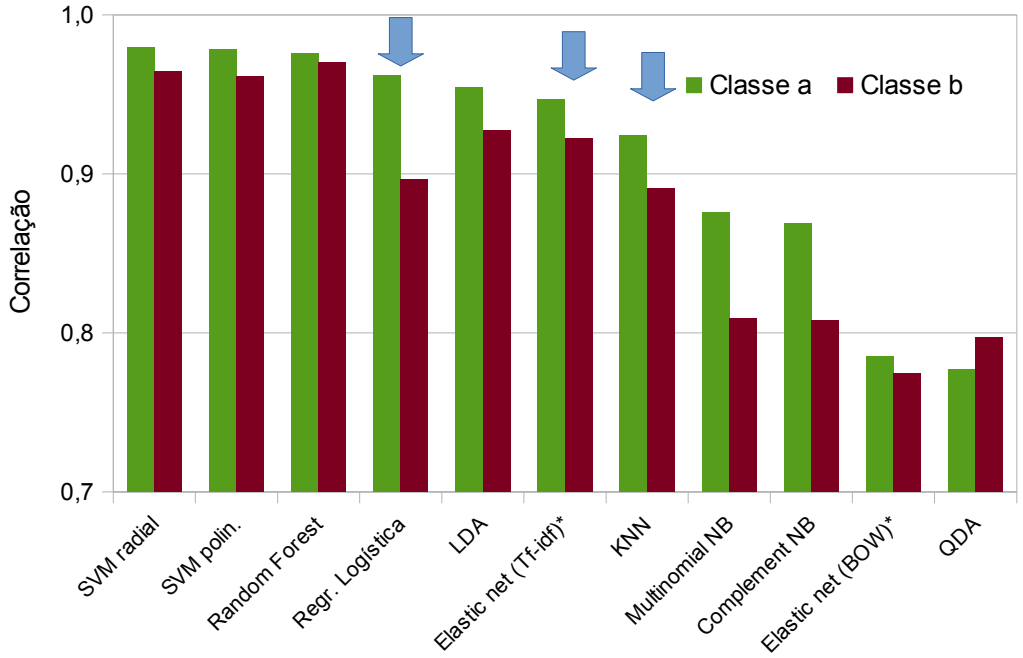
A Tabela 14 traz os resultados referentes às quatro bases em que os comitês obtiveram os maiores escores microF2, no conjunto de teste nível 1. Para os comitês de votação, as linhas preenchidas com a inscrição "Principal" (coluna Pesos) referem-se aos resultados obtidos na votação ponderada, com maior peso atribuído ao modelo nível de maior escore microF2 individual (item 4.1 da seção 4.3.5.1). Pode-se observar que:

- Procedimentos simples de votação – combinando 3 ou 4 modelos com pesos iguais, por exemplo – já são suficientes para melhorar a performance em relação ao dos modelos individuais.
- Reunir um modelo de cada tipo (linear, não linear, baseado em distâncias ou em árvores) em comitê de votação gera bom resultado. Isto aparenta estar em linha com a recomendação, realizada por Tsymbal et al. [87] e Tumer e Ghosh [88], de empregar modelos diversos entre si.
- Não é possível afirmar que os modelos com baixa correlação com o modelo principal produzem melhores comitês, como se observa no Gráfico 4. As três setas azuis mostram os modelos escolhidos pelo melhor comitê de votação *soft* encontrado no experimento. Além do modelo principal Árvores *boosted*, esse comitê ótimo foi formado pelos modelos com o 2º, 7º e 9º maiores valores de microF2, ou seja, o melhor comitê de votação **não** necessariamente é o que reúne os n melhores modelos individuais. Essa situação se repete nos demais comitês de votação, em geral, demonstrando a necessidade de testar o maior número possível de combinações.
- De modo geral, o *stacking* proporcionou resultados superiores aos das votações. O SVM de kernel radial como metamodelo apresentou o melhor escore geral microF2 (95,88%) na base com subamostragem OSS sem metadados (último quadro), representando uma melhoria de 1,4 p.p. em relação ao maior valor obtido pelos modelos individuais, embora produza mais falsos positivos.
- O SVM de kernel radial – aplicado à base de assuntos originais sem metadados e sem subamostragem (terceiro conjunto de tabelas) – apresenta

valores de microF2, microF1 e acurácia (95,65%, 94,73% e 98,49%, respectivamente) superiores aos da Tabela 13, equilibrando melhor precisão e *recall*. Este modelo, portanto, constitui o novo *benchmark* para futuros testes e representa um ganho potencial de eficiência relevante frente ao processo de triagem atual, visto que seleciona menos de 7% de falsos positivos enquanto a proporção hoje está entre 70 e 80%.

- Contrariando Seewald [61], o *StackingC* não se provou ser necessariamente a melhor opção em relação à configuração padrão de Ting e Witten [60].

Gráfico 4 - Correlações entre as predições de probabilidade das Árvores *boosted* e demais modelos (base as_mod* com subamostragem OSS)



*Todos os modelos treinados com 2.000 atributos e representação Tf-idf, exceto a regressão Elastic Net, treinada com todos os atributos.
As setas azuis indicam os modelos escolhidos para formar o melhor comitê de votação *soft* encontrado no experimento.

Tabela 14 - Resultados selecionados dos comitês (*Ensembles*) (%)

Base com assuntos modificados, sem metadados (as_mod*), sem subamostragem

Votação

Tipo	Pesos	Principal	Peso	Modelo 1	Peso 1	Modelo 2	Peso 2	MicroF2	Ac.	MicroF1	MicroR	MicroP
Soft	Principal	SVM radial	1	A. boosted	1	Elastic net*	1	94,85	**98,58**	**95,03**	94,72	**95,34**
Soft	Iguais	SVM radial	1	A. boosted	1	Elastic net*	1	94,85	**98,58**	**95,03**	94,72	**95,34**
Hard	Principal	A. boosted	1	Elastic net*	1	NSC	1	**95,30**	98,56	94,96	**95,54**	94,39
Hard	Iguais	A. boosted	1	Elastic net*	1	NSC	1	**95,30**	98,56	94,96	**95,54**	94,39

Stacking

Tipo	Metamodelo	MicroF2	Ac.	MicroR	MicroP
Padrão	Regr. Logística	94,86	98,49	94,97	94,43
Padrão	KNN	95,07	98,58	95,13	94,82
Padrão	SVM radial	**95,51**	98,64	**95,70**	94,77
StackingC	Regr. Logística	94,93	98,51	95,05	94,44
StackingC	KNN	94,54	98,49	94,48	94,79
StackingC	SVM radial	95,17	**98,69**	95,05	**95,67**

Base com assuntos modificados e subamostragem Tomek Links, <u>sem</u> metadados (as_mod*)

Votação

Tipo	Pesos	Principal	Peso	Modelo 1	P1	Modelo 2	P2	Modelo 3	P3	Modelo 4	P4	MicroF2	Ac.	MicroF1	MicroR	MicroP
Soft	Principal	Elastic net*	1	A. boosted	1	SVM polin.	1	-	-	-	-	**95,20**	**98,48**	**94,69**	**95,54**	**93,86**
Soft	Iguais	Elastic net*	1	A. boosted	1	SVM polin.	1	-	-	-	-	**95,20**	**98,48**	**94,69**	**95,54**	**93,86**
Hard	Principal	SVM radial	1	A. boosted	1	NSC	1	-	-	-	-	94,88	98,31	94,15	95,37	92,96
Hard	Iguais	Elastic net*	1	SVM radial	1	A. boosted	1	KNN	1	NSC	1	94,97	98,42	94,49	95,29	93,70

Stacking

Tipo	Metamodelo	MicroF2	Ac.	MicroF1	MicroR	MicroP
Padrão	Regr. Logística	95,36	98,49	94,74	95,78	93,73
Padrão	KNN	95,50	**98,56**	**94,97**	95,86	**94,10**
Padrão	SVM radial	**95,57**	98,54	94,91	**96,02**	93,81
StackingC	Regr. Logística	95,23	98,44	94,55	95,70	93,42
StackingC	KNN	95,29	98,52	94,81	95,62	94,01
StackingC	SVM radial	95,37	98,54	94,89	95,70	94,09

*Todos os modelos foram executados com 2.000 atributos e representação Tf-idf, exceto a regressão Elastic net, que utilizou todos os atributos.

SVM polin.: SVM polinomial
NSC: *Nearest Shrunken Centroids*
A. boosted: Árvores *boosted*

Base com assuntos originais, sem metadados (as*), sem subamostragem

Votação

Tipo	Pesos	Principal	Peso	Modelo 1	P1	Modelo 2	P2	Modelo 3	P3	Modelo 4	P4	MicroF2	Ac.	MicroF1	MicroR	MicroP
Soft	Principal	SVM polin.	2	A. boosted	1	-	-	-	-	-	-	94,13	**98,39**	**94,34**	93,99	**94,69**
Soft	Iguais	SVM radial	1	SVM polin.	1	A. boosted	1	Random Forest	1	QDA	1	94,28	98,30	94,09	94,40	93,79
Hard	Principal	SVM radial	1	A. boosted	1	NSC	1	-	-	-	-	94,70	98,30	94,06	95,13	93,02
Hard	Iguais	SVM radial	1	A. boosted	1	Regr. Logística	1	QDA	1	NSC	1	**94,73**	98,19	93,77	**95,37**	92,23

Stacking

Tipo	Metamodelo	MicroF2	Ac.	MicroF1	MicroR	MicroP
Padrão	Regr. Logística	93,89	98,34	94,09	93,75	**94,44**
Padrão	KNN	94,85	98,43	94,44	95,13	93,76
Padrão	SVM radial	**95,65**	**98,49**	**94,73**	**96,27**	93,24
StackingC	Regr. Logística	94,09	98,34	94,11	94,07	94,15
StackingC	KNN	95,16	98,47	94,61	95,54	93,71
StackingC	SVM radial	94,75	98,42	94,43	94,97	93,90

Base com assuntos modificados e subamostragem One-sided Selection, sem metadados (as_mod*)

Votação

Tipo	Pesos	Principal	Peso	Modelo 1	P1	Modelo 2	P2	Modelo 3	P3	Modelo 4	P4	MicroF2	Ac.	MicroF1	MicroR	MicroP
Soft	Principal	A. boosted	2	Elastic net*	1	Regr. Logística	1	KNN	1	-	-	**95,42**	**98,38**	**94,42**	**96,10**	**92,79**
Soft	Iguais	A. boosted	1	Elastic net*	1	-	-	-	-	-	-	95,23	98,29	94,18	95,94	92,49
Hard	Principal	A. boosted	3	Elastic net*	1	SVM radial	1	NSC	1	Complement NB	1	94,98	98,13	93,69	95,86	91,62
Hard	Iguais	A. boosted	1	Elastic net*	1	SVM radial	1	Regr. Logística	1	NSC	1	95,09	98,23	93,95	95,86	92,12

Stacking

Tipo	Metamodelo	MicroF2	Ac.	MicroF1	MicroR	MicroP
Padrão	Regr. Logística	95,83	98,08	93,57	97,40	90,02
Padrão	KNN	95,13	97,86	92,80	96,75	89,15
Padrão	SVM radial	**95,88**	98,05	93,47	**97,56**	89,70
StackingC	Regr. Logística	95,82	98,17	93,78	97,24	90,55
StackingC	KNN	95,50	98,00	93,22	97,08	89,66
StackingC	SVM radial	95,72	**98,19**	**93,87**	**97,00**	**90,94**

4.4. Classificação via redes neurais profundas

4.4.1. Desenho do experimento

Base de dados. Os experimentos com redes neurais profundas utilizaram somente a base de tokens as_mod* (tokens com assuntos modificados sem metadados) por ser a de maior quantidade de tokens e ter proporcionado bons resultados nos algoritmos clássicos. A base não sofreu subamostragem, já que as redes neurais necessitam de grande volume de documentos. Como as redes neurais profundas (com exceção da MLP) recebem tokens na sequência em que figuram nos documentos, optou-se por não agregar as variáveis categóricas de metadados porque não se encaixavam nessa sistemática.

O sequenciamento de tokens implementado na biblioteca Keras [89] exige que os documentos tenham o mesmo comprimento, portanto foi necessário limitá-lo em 1.000 tokens para não prejudicar o aprendizado, nem comprometer o tempo de execução. Textos mais longos passaram por um processo de seleção de atributos similar ao dos algoritmos clássicos: foram selecionados os tokens mais bem colocados segundo o critério combinado IM+χ^2 desde que o conjunto escolhido (mantendo as frequências e posições dos tokens no texto original) não ultrapassasse o limite estipulado. Documentos com menos de 1.000 tokens tiveram seus vetores completados com valores nulos (*padding*).

As redes neurais utilizaram os mesmos conjuntos de treinamento e teste trabalhados pelos algoritmos clássicos. O conjunto de validação foi gerado ao separar 25% do conjunto de treinamento de forma estratificada. Assim, a composição dos conjuntos que serviram de entrada para as redes neurais, para a base as_mod* selecionada, é:

Tabela 15 - Composição dos conjuntos de dados trabalhados pelas redes neurais profundas

Conjunto	Documentos
Treinamento	25.913
Validação	8.398
Teste	8.397

Representação numérica. A única representação utilizada foi a Word2vec, por ser capaz de vetorizar tokens individualmente. A dimensão vetorial (100, 200 ou 300) foi um dos hiperparâmetros testados.

Arquiteturas. Os experimentos de classificação com redes neurais profundas foram realizados com todas as arquiteturas apresentadas no Capítulo 3 nas suas formas simples, além de variantes mais sofisticadas habitualmente utilizadas em problemas de PLN, como segue:

- GRU com apenas uma camada.
- LSTM com uma e duas camadas empilhadas, com processamento bidirecional ou no mesmo sentido.
- Mecanismo de atenção de Bahdanau et al. [71] acoplado às arquiteturas GRU e LSTM.
- CNN proposta por Chollet[29], com três camadas convolucionais unidimensionais em série resumidas por *max pooling*, cuja saída é processada por uma camada densa e outra de *dropout*. A última camada, comum a todas as arquiteturas testadas, é densa com função de ativação softmax e realiza a classificação final.
- MLP com uma camada escondida (rasa) considerada como *baseline*, por sua simplicidade e rapidez de execução.

Ajuste de hiperparâmetros. Procurou-se seguir as orientações práticas de Goodfellow et al. [43] e Bengio [63]. Aproximadamente 80% das pesquisas foram exaustivas (*grid search*) e as restantes, aleatórias. No *grid search*, as redes realizaram de dois a dez treinos por configuração para testar diferentes inicializações aleatórias de pesos. Escolheu-se o conjunto de pesos que resultou no microF2 máximo no conjunto de validação para comparar o resultado da configuração com as demais.

Foram testadas no total mais de 4.000 configurações variando-se os hiperparâmetros listados na Tabela 16, por ordem de preferência.

29 https://keras.io/examples/nlp/pretrained_word_embeddings/

Tabela 16 - Faixas de valores testados de hiperparâmetros de redes neurais profundas

Hiperparâmetro	Faixa de valores testados
Taxa de aprendizado	10^{-4} a 10^{-1}
Número de unidades	32 a 480
Dimensão de vetorização	100, 200, 300
Taxa de *dropout* interno	0 a 80%
Tamanho de lote (*batch size*)	32 a 320
Localização camada extra de *dropout*	Entre a camada de *embedding* e a última camada densa
Taxa da camada extra de *dropout*	0 a 80%
Taxa de *dropout* recorrente	0 a 20%

Função perda. Adotou-se a entropia cruzada categórica (12) como função perda devido à sua adequação a problemas de classificação multiclasse.

Early Stopping. Foi adotado como padrão o *early stopping* com tolerância de três épocas sem diminuição do valor da entropia cruzada categórica, assumido no conjunto fixo de validação.

4.4.2. Resultados, análise e discussão

Elegeram-se os dez modelos (ou dez configurações de hiperparâmetros) de cada arquitetura, com os escores microF2 mais elevados no conjunto de validação, para efetuar predições no conjunto de teste. A Tabela 17 apresenta as configurações de redes neurais profundas com os maiores escores microF2 no conjunto de teste, por arquitetura testada. É possível observar que:

- A diferença entre a acurácia e o microF2 dos conjuntos de treinamento e teste denotam que o *baseline* não conseguiu generalizar bem, a despeito da camada extra de *dropout* empregada.
- O *dropout* interno, funcionalidade oferecida pela biblioteca Keras, se mostrou mais interessante que adicionar as camadas extras de *dropout* às redes testadas. O *dropout* interno é um parâmetro da arquitetura de rede constituído por máscaras aplicadas aos filtros (*gates*) de cada unidade, enquanto a camada extra de *dropout* se aplica às unidades como um todo, i.e., remove temporariamente unidades inteiras.

- Todas as arquiteturas de redes profundas ofereceram performance superior ao da MLP.
- A arquitetura CNN apresentou métricas ligeiramente superiores às demais. Isso contrasta com as observações de outros pesquisadores que estudaram os desempenhos de redes neurais profundas em tarefas de classificação, como Zulqarnain et al. [90], Yin et al. [91] e Zhao et al. [92], os quais encontraram seus melhores resultados com a arquitetura GRU, valendo ressaltar que a métrica de interesse era a acurácia nesses casos.
- Uma possível causa do desempenho ligeiramente inferior das redes em relação aos modelos clássicos individuais são as dimensões vetoriais testadas (100, 200 e 300), que podem ter sido insuficientes quando comparadas aos 2.000 atributos dos algoritmos clássicos.

De modo geral, configurações aleatórias de redes atingem microF2 de 88 a 90% no conjunto de validação sem dificuldade, cabendo à sintonia fina melhorar o escore. Três hiperparâmetros se revelaram mais influentes: a taxa de aprendizado, o número de unidades e a taxa de *dropout* interno. O Gráfico 5 exemplifica a variação de microF2 em função do número de unidades para as três dimensões vetoriais de entrada testadas, mantendo constantes os demais hiperparâmetros. O Gráfico 6 mostra a variação de microF2 em função da taxa de *dropout* interno para três taxas de aprendizado distintas.

Os gráficos mostram que, em relação aos três hiperparâmetros analisados, o comportamento do escore oscila substancialmente e apresenta máximos locais, o que dificultou a busca de configurações que oferecessem bons resultados. Múltiplos treinos com uma mesma configuração revelaram que o desempenho da rede variava de forma considerável conforme os pesos iniciais determinados aleatoriamente – em alguns casos obteve-se desvio-padrão de 2% no escore microF2 medido sobre o conjunto de validação. Essa variabilidade é relevante, face ao intervalo de dez pontos percentuais dentro do qual se buscou o melhor resultado, e dificultou o ajuste dos hiperparâmetros.

O estudo do problema de classificação foi concluído após a etapa de exploração de redes neurais, dado que os resultados obtidos com os modelos

clássicos foram suficientemente satisfatórios para motivar a sua implementação e requerem custo menor de colocação em regime de produção e manutenção futura.

Tabela 17 - Configurações de redes neurais profundas com os maiores escores microF2 no conjunto de teste (%)

Arquitetura	Tx. Apr.	Nº Unids.	Emb.	Dropout (%)	Tamanho do lote	Camada Dropout	Tx. Drop. (%)	MicroF2 Teste(%)	Ac. Teste(%)	Recall Médio (%)	Prec. Média (%)	MicroF2 Trein.	Ac. Trein.
MLP (*baseline*)	0,00053	128	200	-	128	1	21	82,04	94,69	82,14	82,15	**98,78**	**99,69**
LSTM	0,005	224	200	20	256	1	20	92,16	97,20	93,29	88,97	95,20	98,65
GRU	0,0029	208	300	20	64	-	-	92,96	97,36	94,46	88,31	95,80	98,85
LSTM com atenção	0,0063	256	300	20	128	-	-	92,68	97,59	93,49	90,52	95,81	98,79
GRU com atenção*	0,0015	96	300	42	64	1	34	93,28	**97,76**	94,30	**90,85**	95,58	98,79
LSTM Bidirecional**	0,01	256	200	20	256	-	-	91,28	97,00	92,42	88,14	95,10	98,65
LSTM com 2 camadas**	0,0078	192	200	20	256	-	-	91,91	97,33	92,95	89,88	94,74	98,49
CNN***	0,00076	192	300	-	64	1	57	**93,84**	97,34	**95,90**	87,62	94,94	98,49

* O valor do hiperparâmetro beta1 do otimizador Adam foi alterado para 0,89, no lugar do padrão 0,90. A configuração apresentada empregou valor de paciência de 5 épocas em vez do padrão de 3 épocas.
** As duas camadas empilhadas possuem sempre o mesmo número de unidades.
*** Com 3 kernels, 1 *stride*, 192 filtros e *poolsize* = 5.
Todas as configurações mostradas possuem epsilon = 10^{-7}, taxa de *dropout* recorrente = zero e paciência = 3 épocas, exceto quando indicado.
Tx. Apr.: Taxa de aprendizado.
Nº Unids.: Número de unidades na camada escondida.
Emb.: Dimensão vetorial de *embedding*.
Dropout: Taxa de *dropout* interno.
Camada Dropout: Nº de camadas extras de *dropout* aplicadas.
Tx. Drop.: Taxa da camada de *dropout.*
Ac.: Acurácia.
Recall e prec. médios: Médias de escores de *recall* e precisão das classes **a** e **b**, obtidos no conjunto de teste.
MicroF2 e Ac. Trein.: MicroF2 e acurácia obtidos no conjunto de treinamento.

Gráfico 5 - Escore microF2 no conjunto de validação para a rede GRU com atenção, em função do número de unidades e dimensão vetorial de entrada

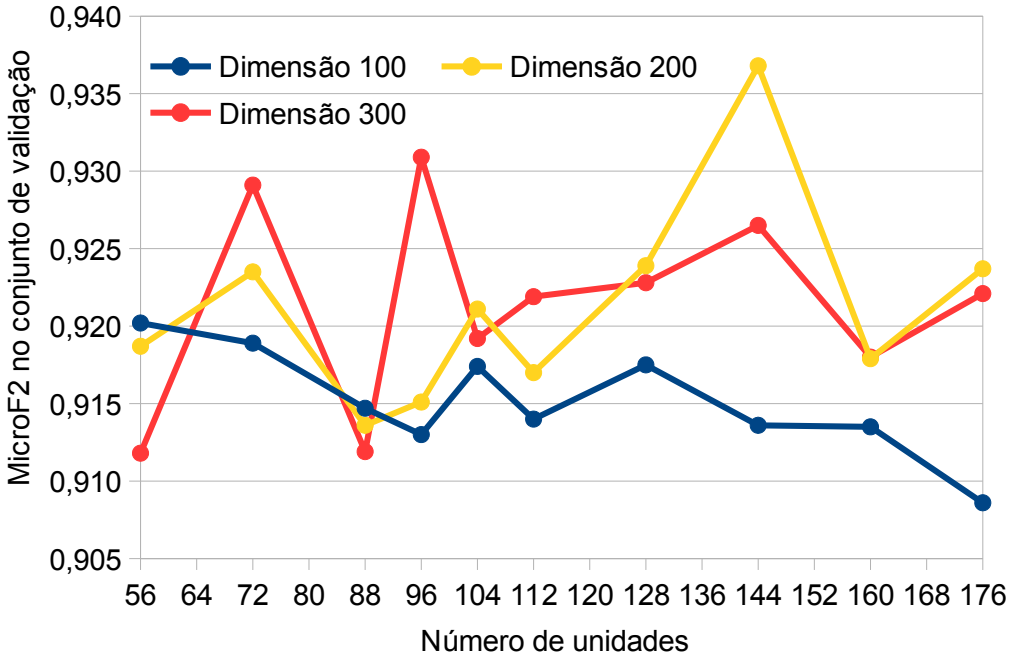
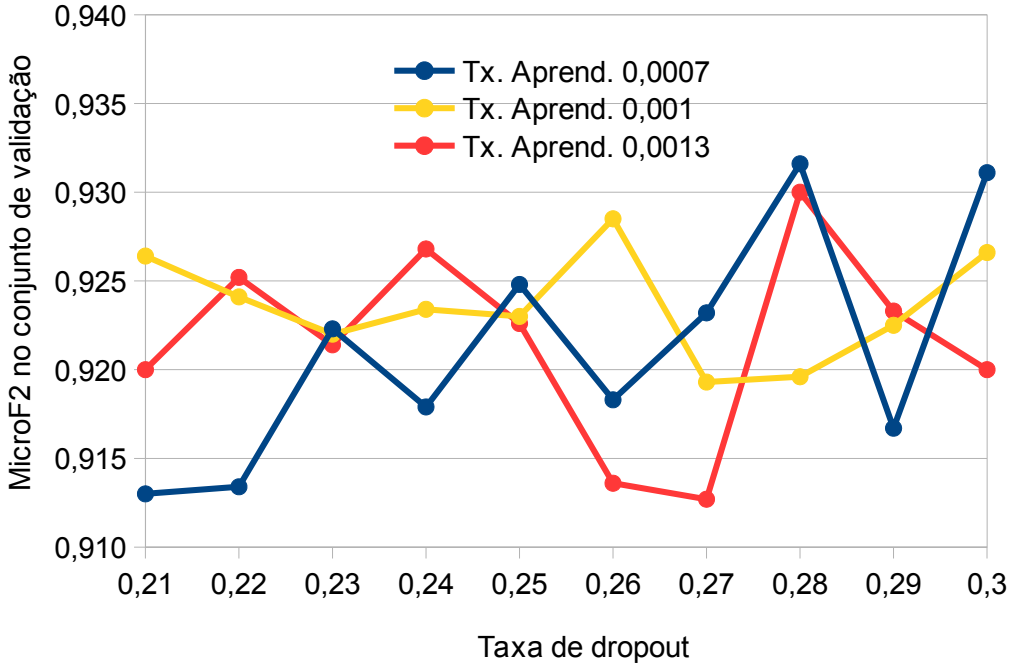


Gráfico 6 - Escore microF2 no conjunto de validação para a rede GRU com atenção, em função da taxa de dropout interno e taxa de aprendizado



5
Aplicação do processamento de linguagem natural para extração de informações sobre operações societárias

5.1.
Metodologia

Arquitetura. Decidiu-se aplicar apenas redes neurais profundas para resolver o problema pelo fato de representarem o estado-da-arte[30] e proporcionarem resultados nitidamente superiores – em alguns casos verifica-se diferença de 40 p.p. em relação a outros tipos de algoritmos, vide [93] por exemplo. O BERT é a escolha natural, visto que foi a primeira arquitetura que possibilitou esse avanço na ocasião de seu lançamento, em 2018.

Base de dados. O conjunto de dados utilizado para a extração de informações foi construído a partir dos textos originais, retirando-se poucos caracteres gráficos sem significado (p.ex., ` e ^ sem letras) e quatro palavras geradas indevidamente pela biblioteca de OCR. Sinais de pontuação e *stopwords* foram mantidos no conjunto de dados formado por texto corrido, sem necessidade de efetuar tokenização. A base de dados agregou os assuntos originais aos conteúdos dos documentos e foi dividida em frases pela biblioteca Stanza [94].

A extração de informações foi direcionada para frases (ou contextos) previamente selecionadas dos documentos porque as implementações atuais da arquitetura BERT não são capazes de processar adequadamente textos de várias páginas. Selecionaram-se frases que continham pelo menos uma expressão indicativa de informação sobre operações de aumento de capital ou incorporação. As expressões buscadas estão na Tabela 18. As reticências indicam aceitação de qualquer combinação de letras que completam a palavra ou formam uma

30 Todos os modelos que obtiveram os maiores escores na SQuAD são baseados em redes neurais (https://rajpurkar.github.io/SQuAD-explorer/).

expressão (p. ex., “aumentar o capital”). Duas bases para rotulação foram montadas: uma com 2.401 frases sobre aumento de capital e outra com 2.717 frases sobre incorporação.

Tabela 18 - Expressões buscadas para extração de informações

Assunto	Expressão
Incorporação	incorpora...
	sucessor(a)
	sucederá
Aumento de capital	aument… capital
	emissão
	emit...
	novas ações

A rotulação procurou identificar as respostas aplicáveis às seguintes perguntas, conforme a base correspondente:

Para a base sobre aumento de capital:

- Quantas ações, bônus de subscrição ou debêntures conversíveis foram emitidas no máximo por aumento de capital?

Para a base sobre incorporação:

- Quem foi incorporada?
- Quem foi a incorporadora?

A rotulação buscou todas as respostas aplicáveis nos contextos, conquanto que contivessem informações relevantes para a atividade de supervisão, sem haver sobreposição entre elas. A base sobre aumento de capital contém 1.265 frases com respostas e a base sobre incorporação possui 1.776 frases que informam sobre a companhia incorporadora e 1.500 frases sobre a incorporada.

Aferição de desempenho e métricas. O pré-processamento anterior permitiu que o problema passasse a se enquadrar na definição de *Span-based Question Answering* ou perguntas formuladas sobre passagens (também chamadas de contextos) e respostas ali buscadas. O estado-da-arte da resolução desse problema de compreensão de texto é pesquisado com o uso da base Stanford Question Answering Dataset (SQuAD) [95], compreendendo 100.000 questões sobre passagens de artigos da Wikipedia, das quais 50.000 **não** contêm as respostas desejadas. Como cada pergunta da SQuAD foi respondida por mais de

um indivíduo, ela admite múltiplas opções de resposta de mesmo conteúdo semântico, que podem estar sobrepostas no contexto. A avaliação é realizada da seguinte forma:

- Para uma dada pergunta em um contexto, a avaliação considera somente a predição de resposta mais provável (máxima probabilidade predita)
- Computa-se o escore da resposta escolhida contra todas as respostas rotuladas admitidas.
- O escore da resposta selecionada para o contexto será o maior dos escores anteriores.

Uma vez que o problema da dissertação foi modelado de forma similar ao do SQuAD, adotaram-se as mesmas métricas de desempenho aplicáveis à base de Stanford: F1 – calculada segundo (19) e considerada como principal – e EM.

No entanto, no caso das perguntas sobre aumento de capital e incorporação, as opções de resposta podem se referir a informações complementares ou a eventos distintos igualmente relevantes. A Tabela 19 contém um exemplo correspondente com as duas respostas válidas marcadas em negrito. Por isso, o cômputo das métricas de desempenho adotado é ligeiramente distinto do aplicado à base SQuAD, como descrito a seguir.

- Se uma questão possui **n** respostas rotuladas em um determinado contexto, o sistema primeiro seleciona as **n** respostas preditas pela rede neural com as maiores probabilidades.
- Para a primeira resposta rotulada, calcula-se os escores de todas as respostas preditas que foram selecionadas e o escore máximo será atribuído à resposta rotulada em questão. A resposta predita correspondente ao escore máximo é removida da amostra e repete-se o processo para a segunda resposta rotulada, até que se esgotem as **n** respostas rotuladas.
- Portanto, para efeito de cálculo de escores para o contexto em análise, a questão original de **n** respostas rotuladas é desdobrada em **n** questões iguais.

Tabela 19 - Exemplo de pergunta com múltiplas respostas complementares

Pergunta	Contexto
Quem foi incorporada?	Com relação à incorporação da **parcela cindida da BRPR 55** e da incorporação da **BRPR 56**, que são atualmente companhias securitizadoras emissoras de Certificados de Recebíveis Imobiliários - CRIs, a Companhia espera que até a data da AGE tais companhias não terão mais CRIs de sua emissão em circulação por aquisição pelas emissoras dos CRIs em circulação ou quitação da dívida securitizada.

Tal como no SQuAD 2.0, as perguntas são propostas para todos os contextos, mas alguns deles não possuem conteúdo suficiente para respondê-las. Essas perguntas são computadas como "sem resposta": os escores F1 e EM serão iguais a um se o modelo não predisser resposta alguma; e zero, caso contrário.

O cálculo anterior foi executado em sua totalidade para a métrica F1 e novamente para EM. O escore de cada configuração testada de rede foi considerado como a média dos escores referentes a todas as perguntas, desdobradas ou não, e incluindo as sem resposta.

Montagem dos conjuntos de treinamento, validação e teste. O particionamento das bases rotuladas se deu de forma estratificada, assumindo que um contexto com teor suficiente para responder pelo menos uma questão é da classe positiva e o contexto com perguntas sem respostas é da classe negativa. A Tabela 20 traz a divisão resultante das bases.

Tabela 20 - Divisão das bases de contextos em conjuntos de dados para o problema de extração de informações

Conjunto	Nº de contextos Aumento de capital	Nº de contextos Incorporação
Treinamento	1536	1738
Validação	384	435
Teste	481	544

Hiperparâmetros. Sobre cada base de contextos sobre aumento de capital e incorporação, utilizou-se a biblioteca Simple Transformers[31] para realizar a sintonia fina de um modelo-base BERT multilíngue pré-treinado[32]. Inicialmente foram testadas preliminarmente por base mais de 500 configurações aleatórias dos seguintes hiperparâmetros. As faixas de valores testados estão descritas no Apêndice 7.

31 https://github.com/ThilinaRajapakse/simpletransformers
32 BERT-Base, Multilingual Cased

- Nº de épocas de treinamento
- Taxa de aprendizado
- Tamanho do lote de treinamento (*train batch size*) em nº de contextos
- Tamanho do lote de validação (*evaluation batch size*) em nº de contextos
- Constante epsilon do algoritmo de otimização AdamW
- Nº máximo de tokens de um lote de entrada (contexto + pergunta) na rede (*maximum sequence length*)
- Nº de tokens de contexto sobrepostos entre lotes de entrada (*document stride*), para processar contextos longos
- Nº de lotes (*steps*) de tolerância, sem redução da função perda (*patience*)
- Proporção de lotes iniciais para aquecer o treinamento (*warmup ratio*)
- Expoente da taxa de decaimento da taxa de aprendizado (*polynomial decay learning rate end value*)
- Valor final da taxa de aprendizado de decaimento (*polynomial decay*)
- Taxa de aprendizado final
- Nº de lotes de acumulação de gradiente (*gradient accumulation steps*)
- Nível de corte do gradiente (*max gradient norm*)
- Nº de lotes entre cada avaliação de métricas no conjunto de validação (*evaluate during training steps*)
- Coeficiente de regularização L2 da função perda (*weight decay*)

Função perda. Adotou-se a entropia cruzada categórica, calculada conforme (14), para avaliar o conjunto de validação durante o treinamento (*early stopping*) e selecionar configurações de rede. Testes preliminares revelaram que o otimizador AdamW (Adam com decaimento de pesos) [96] obteve menores valores para a função perda que o otimizador Adafactor, tendo o primeiro sido priorizado.

Descrição dos experimentos. Para cada base sobre aumento de capital e incorporação, escolheram-se as onze configurações de rede com os menores valores da função perda no conjunto de validação durante os testes preliminares, para realizar dois experimentos:

1. O primeiro experimento consistiu em estudar o desempenho da rede em diferentes conjuntos de dados por meio de validação cruzada 5-*fold*, com os conjuntos de treinamento e validação da Tabela 20 unificados. Em seguida, a rede foi retreinada no conjunto de treinamento inteiro para realizar predições no conjunto de teste. Não se empregou *early stopping* neste experimento[33].
2. O segundo experimento compreendeu o treinamento tradicional de rede neural, com *early stopping* e conjuntos de treinamento e validação fixos tal qual descritos na Tabela 20, além de posterior predição no conjunto de teste.

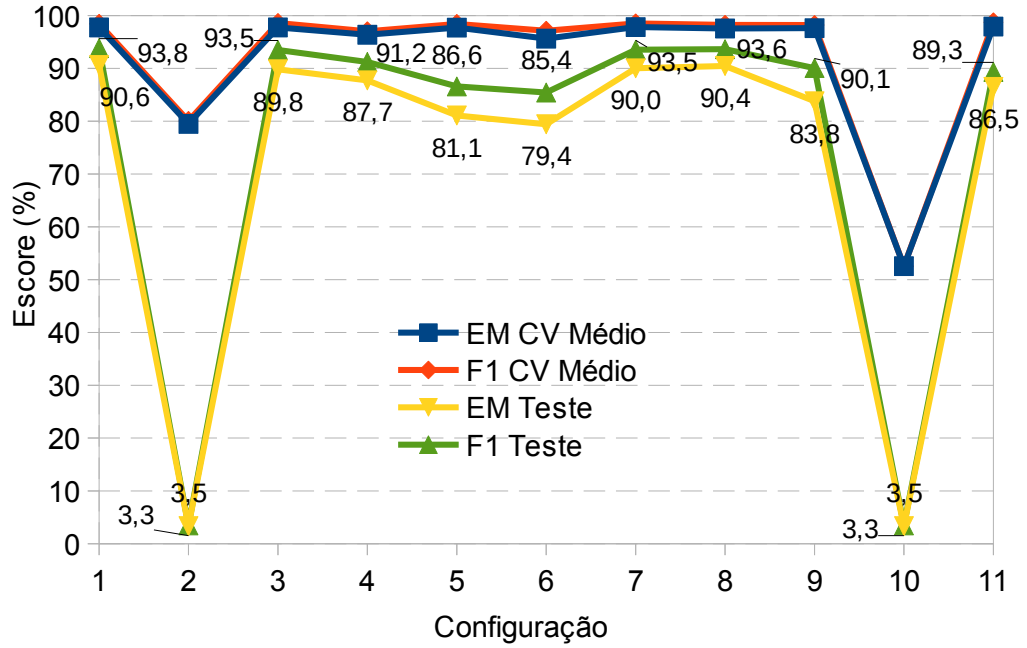
5.2. Resultados, análise e discussão

O Gráfico 7 mostra os escores EM e F1 obtidos no primeiro experimento com os contextos sobre aumento de capital. São apresentados a média dos *folds* no conjunto de validação e o escore no conjunto de teste, por configuração. Observa-se que:

- Embora a validação cruzada estime razoavelmente os escores no conjunto de teste, a ausência de *early stopping* acarretou sobreajuste de duas configurações números 2 e 10, as quais falharam nas predições no conjunto de teste.
- Com exceção das duas configurações, os desvios-padrão dos escores F1 e EM da validação cruzada são inferiores a 1%, denotando constância de desempenho das redes em conjuntos distintos.

33 A biblioteca gerou erro desconhecido nas predições no conjunto de validação, inviabilizando a avaliação durante o treinamento. No entanto, dessa forma os resultados da validação cruzada são comparáveis aos do treino final realizado sem conjunto de validação e, portanto, não possibilitava o *early stopping*.

Gráfico 7 - Escores EM e F1 no conjunto de teste e escores médios no conjunto de validação para o aumento de capital (1º experimento)



OBS: Os rótulos dos escores médios da validação cruzada foram omitidos.

Para o segundo experimento, as Tabelas 21 e 22 trazem os escores EM e F1 relativos a todas as perguntas sobre aumento de capital e incorporação, respectivamente. As colunas EM_s e F1_s trazem os escores calculados exatamente segundo a metodologia do SQuAD. As tabelas também apresentam os escores de forma estratificada por tipo de pergunta, isto é, dividindo-as entre aquelas que possuem resposta ou não. É possível concluir que:

- No caso do aumento de capital, a configuração nº 11 atingiu os melhores escores no cômputo geral e a segunda colocação para as perguntas com resposta.
- Embora algumas configurações, quando executadas sem *early stopping* (Gráfico 7), proporcionaram escores superiores aos alcançados com o mecanismo (Tabela 21), os valores de EM e F1 mais relevantes (marcados em negrito na tabela) são aqueles obtidos com *early stopping*.
- Para as perguntas sobre incorporação, a configuração nº 11 se mostra como a mais interessante, sob o mesmo racional do aumento de capital. O fato de as informações relativas a incorporações serem compostas por trechos mais

longos que as respostas do caso anterior possivelmente explica o desempenho inferior.

Tabela 21 - Escores EM e F1 no conjunto de teste para aumento de capital (2º experimento)

	Todas as perguntas				Com resposta		Sem resposta	
Config.	EM	F1	EM_s	F1_s	EM	F1	EM	F1
1	90,44	94,09	88,26	92,26	**84,65**	**92,36**	95,65	95,65
2	87,73	91,75	85,63	90,01	77,63	86,11	96,84	96,84
3	91,06	94,39	89,07	92,70	84,21	91,23	97,23	97,23
4	64,03	67,80	62,96	66,85	79,82	87,78	49,80	49,80
5	88,15	92,35	86,03	90,77	79,39	88,24	96,05	96,05
6	74,01	80,34	72,27	79,33	50,44	63,79	95,26	95,26
7	89,40	93,45	87,25	91,97	82,02	90,57	96,05	96,05
8	**91,27**	94,27	88,87	92,42	84,21	90,54	**97,63**	**97,63**
9	82,54	88,79	80,77	87,66	65,79	78,98	**97,63**	**97,63**
10	82,54	87,47	80,77	86,07	74,56	84,96	89,72	89,72
11	**91,27**	**94,83**	**89,27**	**93,44**	84,21	91,73	**97,63**	**97,63**

Tabela 22 - Escores EM e F1 no conjunto de teste para incorporação (2º experimento)

	Todas as perguntas				Com resposta		Sem resposta	
Config.	EM	F1	EM_s	F1_s	EM	F1	EM	F1
1	88,69	91,16	80,18	84,72	76,32	82,48	**96,94**	**96,94**
2	75,55	79,54	67,98	73,35	58,62	68,60	86,83	86,83
3	66,36	69,50	60,61	65,36	67,13	74,99	65,85	65,85
4	87,68	90,33	79,85	84,09	**79,77**	86,40	92,96	92,96
5	84,47	87,12	76,00	80,47	66,67	73,30	96,32	96,32
6	86,58	88,90	78,30	82,47	74,48	80,29	94,64	94,64
7	88,33	90,64	79,61	83,52	76,09	81,87	96,48	96,48
8	87,32	90,02	79,12	83,03	**79,77**	**86,54**	92,34	92,34
9	76,10	79,60	68,80	74,23	63,45	72,19	84,53	84,53
10	88,51	90,95	80,26	84,57	79,31	85,42	94,64	94,64
11	**89,71**	**91,93**	**81,33**	**85,02**	**79,77**	85,34	96,32	96,32

Tomando-se como base os escores EM_s e F1_s, a configuração nº 11 do aumento de capital obteve resultados comparáveis aos do estado-da-arte no SQuAD 2.0 (*ensemble* FPNet com EM de 90,87% e F1 de 93,18%[34]). Entretanto, há considerações necessárias:

i. a base de perguntas do SQuAD é a Wikipedia e portanto cobre uma gama vasta de assuntos, enquanto a do experimento restringe-se ao domínio financeiro;

34 https://rajpurkar.github.io/SQuAD-explorer/, consultado em 28/05/2021 às 18h25.

ii. uma parte considerável das perguntas do SQuAD exigem conhecimento geral, interpretação de múltiplas frases e/ou domínio de variações sintáticas e lexicais [95], enquanto o racional das respostas do presente problema é mais simples, pois os contextos dos documentos analisados são uniformes quanto ao estilo de redação e vocabulário e todas as respostas rotuladas são transcrições literais de trechos dos contextos; e
iii. diferentemente do presente caso, o SQuAD assume que todas as respostas de uma pergunta são equivalentes, como já relatado.

De toda forma, o desempenho conseguido pelas redes BERT testadas é suficiente para avançar em direção à sua colocação em regime de produção. Uma funcionalidade importante a ser testada é permitir que o sistema preditivo determine o número de respostas aplicáveis uma pergunta. Um meio possível de viabilizá-la é determinar um patamar de probabilidade mínima (*threshold*) para aceitar respostas. Para servir de base ao futuro desenvolvimento, os valores dos hiperparâmetros das configurações que obtiveram escore F1 acima de 90% estão listados no Apêndice 8.

6
Construção do CVMCorpus

6.1.
Motivação

A resolução dos problemas propostos teve como uma de suas etapas o levantamento de documentos financeiros entregues pelas companhias abertas brasileiras à CVM durante onze anos. Como subproduto do presente trabalho, esse corpus de documentos – denominado CVMCorpus – pode contribuir para preencher as seguintes lacunas de pesquisa científica e aplicações:

- Ausência de corpora especializados no domínio financeiro, em língua portuguesa e disponíveis para o público em geral.
- Inexistência ou escassez de corpora públicos do domínio financeiro, mesmo em outros idiomas, que estejam preparados conforme os preceitos da linguística de corpus (sentenciação, anotação etc.).
- Insuficiência de estudos sobre o mercado de capitais brasileiro que contemplem aplicações de corpora de domínio de larga escala.
- Escassez de trabalhos acadêmicos que aliem conhecimentos de finanças e de linguística de corpus.
- Escassez de aplicações de corpora em ferramentas de gestão pública no Brasil.

Pretende-se que o CVMCorpus seja de domínio público e disponibilizado futuramente no portal da CVM. Seus documentos poderão ser baixados por arquivos txt com ou sem anotações linguísticas. Os metadados dos documentos serão igualmente disponíveis e de fácil acesso por arquivo csv.

O CVMCorpus permitirá que o público em geral acesse a um conjunto de documentos difícil de ser coletado. Embora os documentos estejam disponíveis

nos portais da CVM e da B3 (bolsa de valores paulista), atualmente não é possível pesquisá-los de forma consolidada. Para reproduzir o corpus, seria necessário baixar individualmente cada documento e refazer a busca para cada empresa, além de executar a conversão de caracteres e outras etapas de preprocessamento.

Além de detalhar a construção do CVMCorpus, o presente capítulo examina as suas propriedades linguísticas e traz um exemplo de estudo linguístico que o corpus viabilizará.

6.2. Construção do corpus

Como requerido pela boa prática linguística, as próximas subseções abordarão aspectos da montagem do CVMCorpus que influenciam a sua utilização, em complemento às informações sobre pré-processamento e tokenização descritas na seção 4.2.1, além das convenções adotadas no corpus.

6.2.1. Delimitação de frases

O processo de separação de frases caracterizou-se por:

- Ter sido realizada conforme a disposição gráfica do texto, isto é, observando-se a presença de marcação de parágrafos e maiúsculas antes e após pontos, respectivamente.
- Utilizar, como recurso auxiliar, a biblioteca Stanza [94] para analisar os 5.000 primeiros caracteres de cada arquivo para identificar títulos, subtítulos e cabeçalhos, já que estes geralmente não terminam em sinais de pontuação.
- Buscar seguir os critérios adotados no projeto Linguateca: vírgulas, dois pontos e pontos e vírgulas nunca terminam frases, por exemplo.

Além dos erros de leitura observados na etapa anterior de conversão de arquivos PDF para o formato TXT, outros fatores trouxeram dificuldades para a identificação das sentenças, tais como:

- Atas de reuniões ou assembleias dispostas em formatação justificada e sem quebras de parágrafos.
- Textos digitados totalmente em caixa alta.
- Títulos e subtítulos situados no meio ou final dos arquivos contendo mais de um documento não foram processados, por falta de recurso para identificá-los e ausência de padrão de diagramação. Estima-se que 10 a 20% dos arquivos PDF recebidos pela CVM contém dois ou mais documentos concatenados. Para os problemas de PLN anteriores, documento e arquivo PDF foram considerados sinônimos porque se buscou a classificação de arquivos com base nos seus tokens integrantes, sem entrar no mérito da concatenação de documentos. No entanto, a análise linguística requer maior nível de detalhe quanto à identificação de sentenças. E para o caso de documentos concatenados, as heurísticas anteriores não funcionaram bem para separar títulos e subtítulos da primeira frase do segundo documento em diante.
- Tabelas, gráficos e outros elementos visuais porventura presentes nos documentos não foram retirados por falta de uniformidade e de ferramenta computacional de código aberto disponível, podendo ter gerado frases gramaticalmente incorretas ou tokens ilegíveis.
- Notas de rodapé, números de página, cabeçalhos, marcas d'água e outros fragmentos estranhos ao texto não puderam ser eliminados, devido à falta de padronização na diagramação dos documentos e de ferramenta computacional.

A separação das frases integrantes de cada documento viabilizou a anotação linguística, a análise de bigramas, a extração de contextos para o problema de extração de informações e outros procedimentos.

6.2.2. Limpeza final e montagem

Os tokens foram novamente inspecionados para eliminar, na medida do possível, caracteres ilegíveis por meio de expressões regulares (*regex*) e trechos

em inglês. Devido à natureza globalizada do mercado de capitais, as comunicações realizadas pelas companhias abertas frequentemente incluem versões em inglês, concatenadas ou dispostas em paralelo com o original em vernáculo no mesmo arquivo. Portanto, após a conversão para formato TXT, não raro os documentos apresentavam trechos em idiomas diferentes intercalados, o que justificou a realização da limpeza depois da etapa de delimitação de frases. Pela ausência de padrão gráfico, as frases em inglês foram identificadas com o uso de heurísticas e da biblioteca fasttext [97], a qual infere o idioma do fragmento por meio de um modelo pré-treinado de rede neural. Eliminaram-se da base as frases cuja língua inferida foi o inglês e se: (i) a probabilidade inferida correspondente foi no mínimo 85%; ou (ii) cuja frase anterior foi descartada por força da mesma heurística.

Após a limpeza, o texto do corpus foi montado com os tokens remanescentes, inserindo-se um espaço a cada dois tokens.

6.2.3. Anotação

Um desafio relevante para a anotação e o tratamento de corpora em língua portuguesa de modo geral é a falta de ferramentas computacionais especializadas que apresentem bom desempenho para o idioma, apesar de ser o quinto mais presente na Internet. Uma exceção digna de nota é o analisador (*parser*) PALAVRAS [98], porém ele não está disponível gratuitamente para aplicação em larga escala.

A preparação para o processo de anotação de elementos linguísticos envolveu as seguintes etapas:

- Tratamento para converter ênclises e contrações (formadas por preposição e artigo/pronome) nos seus tokens constituintes, como em “das → de + as” e “tratá-lo → tratar + o”. A conversão foi implementada por meio de regras, portanto vários tipos de ênclises restaram inalterados uma vez que sua resolução dependia do contexto, como em “dar-lhe → dar a ele ou a ela?”.
- Truncagem de frases. Por atenderem a exigências normativas, os documentos de companhias abertas são frequentemente redigidos pelos seus

departamentos jurídicos ou escritórios de advocacia externos. Portanto, pode-se dizer que as comunicações se dão em uma mistura de linguagem financeira e jurídica, com o uso de períodos longos, enumerações e alíneas. Essa característica acarreta um desafio técnico pois, devido ao uso de modelos de redes neurais, o tempo de execução da biblioteca Stanza aumenta consideravelmente ao anotar frases formadas por muitos tokens.

Assim, para viabilizar o corpus, particionaram-se as sentenças acima de 1.000 tokens conforme os seguintes critérios:

a. token formado apenas por um ponto divide a frase;
b. o token : particiona a sentença; e
c. quebra da frase em partes com igual número de átomos, quando a redução resultante dos critérios anteriores não foi suficiente.

Os critérios foram aplicados em sequência até que todos os segmentos da frase em análise tivessem menos de 1.000 tokens. O modelo pré-treinado com base no corpus Bosque[35] foi o selecionado para a anotação, realizada pela biblioteca Stanza [94].

A anotação compreende os seguintes dados linguísticos de cada palavra:

- *part-of-speech* ou classe gramatical universal (POS), conforme convenções estabelecidas no projeto Universal Dependencies[36];
- lema;
- propriedades morfológicas (gênero, número e pessoa); e
- número identificador da cabeça sintática (*head*) da palavra na sentença à qual ela pertence.

O corpus também contém a lista de relações de dependência por frase, com a cabeça sintática e o tipo de relação de cada palavra.

Para o CVMCorpus, foram anotados 42.060 textos (21% do corpus global), que se encontram integralmente incluídos na base rotulada. Esse subcorpus abarca todos os documentos divulgados em 2018 e 2019, ou seja, a parcela mais recente do conjunto total.

35 https://www.linguateca.pt/Floresta/corpus.html
36 https://universaldependencies.org/u/pos/

Cabe notar que o considerável tamanho do CVMCorpus torna inviável sua inspeção manual e, portanto, torna-o sujeito a erros de anotação e pré-processamento. No entanto, o usuário poderá selecionar subcorpora a partir dos metadados disponíveis, efetuar novos processamentos com os textos fornecidos e refazer as anotações, conforme a sua necessidade.

6.3. Principais características

A contabilização básica do corpus encontra-se na Tabela 23, seguindo a terminologia do projeto Linguateca.

Tabela 23 - Contabilização básica do CVMCorpus

Nº de frases	Nº de palavras (formas)	Nº de unidades (tokens)	Nº de tokens únicos
11.166.336	3.277.914	485.837.779	3.451.521

Pode-se observar que, em termos de frases e unidades, o CVMCorpus é comparável aos maiores corpora disponíveis no projeto. Em média, uma frase do corpus da CVM tem 43,5 tokens, enquanto que o índice consolidado da Linguateca é 31,2 unidades/sentença. Isso confirma a constatação anterior sobre o estilo enumerativo e jurídico dos documentos emitidos por companhias abertas.

A Tabela 24 traz a contabilização das palavras e outros tokens do CVMCorpus e do Corpus Brasileiro. Este último é o maior corpus oferecido pelo Linguateca e constitui-se majoritariamente por textos jornalísticos, artigos e trabalhos acadêmicos, sem concentração em área de conhecimento.

A contagem dos tokens do CVMCorpus obedeceu aos seguintes critérios:

- A contagem das abreviaturas utilizou a mesma lista montada para a tokenização, a qual se concentra em tokens terminados em ponto. Consideraram-se abreviaturas símbolos de moedas terminadas em cifrão, seguidos ou não de barra e outros caracteres, como R$/m.
- As palavras incluem os símbolos &, / , $, +, ', -, . e | que estejam eventualmente no seu interior, podendo portanto englobar erradamente algumas abreviaturas, como S/A ou CVM (sem pontos).

- Palavras iniciadas por minúsculas, ainda que seguidas por uma ou mais letras maiúsculas, são consideradas como escritas em minúsculas.
- Todos os tokens que contêm ao menos um dígito em qualquer posição foram computados como números, mesmo que existam letras ou outros símbolos intercalados (p.ex., R$900,00, §2º e 10h00).
- As pontuações incluem alíneas finalizadas por ., - ou).
- O grupo "Outros" abrange sinais gráficos isolados que não representam pontuação, como #, <, > e %, além de palavras que começam por sinais ou possuem quaisquer símbolos.

Tabela 24 - Contabilização de palavras e outros tokens do CVMCorpus e do Corpus Brasileiro

	CVMCorpus			Corpus Brasileiro		
	Nº de formas	%	Nº de tipos	Nº de formas	%	Nº de tipos
Palavras em minúsculas	289.558.811	59,6	536.667	855.677.272	57,9	1.098.243
Palavras iniciadas em maiúscula	66.174.879	13,6	395.835	152.007.309	10,3	875.502
Palavras todas em maiúsculas	23.441.611	4,8	204.081	26.917.075	1,8	279.970
Números	33.295.352	6,9	2.126.641	28.985.780	2,0	88.128
Palavras com números				3.484.358	0,2	267.557
Palavras mistas				5.863.090	0,4	261.858
Abreviaturas	4.910.945	1,01	1.269			
URLs	198.709	0,04	8.872			
Endereços de e-mail	88.875	0,02	4.549			
Total de palavras	417.669.182	86,0	3.277.914	1.249.088.295	84,5	4.018.026
Outros	2.460.156	0,51	160.353			
Pontuação	65.708.441	13,5	13.254	81.257.380	5,5	344.053
Unidades (total da coluna)	485.837.779	100	3.451.521	1.477.645.556	100	4.377.012

Fonte: https://www.linguateca.pt/acesso/contabilizacao.php#cbras (em 12.07.2020).
OBS: A Linguateca não informa o motivo da diferença entre os totais de tokens e dos elementos da tabela.

Os dados da tabela são comparáveis, em boa medida, aos apresentados para os corpora do Linguateca, já que as categorias não equivalentes são pouco representativas.

Analisando a tabela, pode-se observar que:

- Comparando-se a distribuição dos tipos de palavras dos dois corpora, chama a atenção as maiores proporções que as palavras escritas integralmente em maiúsculas (4,8% contra 1,8%) e os números (6,9% versus 2,0%) representam no total de tokens do CVMCorpus.
- O uso frequente de títulos e subtítulos em caixa alta no início dos documentos (um exemplo está no Apêndice 9) e o enfoque natural em valores monetários, confirmado pelo número de tipos muito maior que no Corpus Brasileiro, a despeito deste conter o triplo de tokens, justificam a observação anterior.
- O expressivo número de tokens de pontuação sob a forma de alíneas confirma o estilo enumerativo dos textos financeiros.

A Tabela 25 mostra a distribuição palavras presentes na amostra anotada do CVMCorpus em classes gramaticais (POS), conforme categorização realizada pela biblioteca Stanza.

Tabela 25 - Distribuição das palavras de uma amostra do CVMCorpus por categoria gramatical

Categoria gramatical	Nº de palavras	%
Substantivos	50.324.618	37,7
Verbos	9.439.015	7,1
Adjetivos	7.772.284	5,8
Preposições	26.595.054	19,9
Conjunções	6.320.469	4,7
Advérbios	2.406.883	1,8
Determinantes	19.636.401	14,7
Pronomes	1.666.731	1,2
Numerais	9.374.104	7,0
Interjeições	12.283	0,0

O CVMCorpus é acompanhado dos metadados, listados a seguir, de seus documentos constituintes:

- Identificação do documento
- Razão social da companhia emissora do documento
- Categoria do documento
- Tipo do documento (ou subcategoria)
- Espécie do documento (ou subtipo)

- Assunto do documento
- Data de referência do documento
- Data de entrega do documento ao sistema
- Status do documento (apresentação, reapresentação, cancelado ou outro)
- Número da versão do documento
- Número do arquivo do documento
- CNPJ da companhia emissora do documento
- Código CVM da companhia emissora do documento
- Número do protocolo do documento

6.4. Verificação da Lei de Zipf

A Tabela 26 traz as 30 palavras mais frequentes (i.e., excluindo-se sinais de pontuação) no corpus com suas frequências absoluta e relativa em relação ao total de tokens e palavras. A posição da palavra na lista de tokens também é fornecida.

Tabela 26 - As 30 palavras mais frequentes no CVMCorpus

Palavra	Pos.	Freq. abs.	Freq. relativa Tokens (%)	Freq. relativa Palavras (%)	Palavra	Pos.	Freq. abs.	Freq. relativa Tokens (%)	Freq. relativa Palavras (%)
de	1	28,5	5,9	6,8	os	24	2,4	0,5	0,6
e	3	12,3	2,5	3,0	por	25	2,4	0,5	0,6
a	4	9,4	1,9	2,3	das	26	2,1	0,4	0,5
da	5	9,3	1,9	2,2	as	27	2,0	0,4	0,5
do	7	8,5	1,7	2,0	Conselho	28	1,9	0,4	0,5
em	8	5,9	1,2	1,4	ao	29	1,7	0,4	0,4
o	9	5,5	1,1	1,3	Administração	30	1,7	0,4	0,4
que	13	3,7	0,8	0,9	ações	31	1,6	0,3	0,4
no	14	3,6	0,7	0,9	à	32	1,6	0,3	0,4
Companhia	15	3,6	0,7	0,9	não	33	1,5	0,3	0,4
ou	17	3,3	0,7	0,8	A	34	1,5	0,3	0,4
para	19	3,0	0,6	0,7	pela	35	1,3	0,3	0,3
com	21	2,8	0,6	0,7	O	36	1,1	0,2	0,3
dos	22	2,6	0,5	0,6	nos	37	1,1	0,2	0,3
na	23	2,5	0,5	0,6	pelo	38	1,1	0,2	0,3

Pos.: posição por frequência absoluta.
Frequência absoluta em milhões de palavras.

Com base na tabela, é possível tecer as seguintes observações:

- A palavra mais frequente com conteúdo financeiro é “Companhia”, a qual difere de “companhia” em minúsculas (esta ocupa a 222ª posição) porque a primeira frequentemente substitui o nome da empresa emissora.
- A segunda palavra de cunho financeiro com maior frequência é “Conselho”, que costuma indicar um Conselho de Administração ou um Conselho Fiscal, sendo ambos órgãos administrativos importantes das companhias abertas.
- O termo “ações” é o primeiro da lista que tende a ser sujeito paciente ou objeto direto das frases.
- Palavras ligadas a conceitos contábeis, como “valor”, “milhões”, “exercício”, “social” e “dezembro” formam um bloco relevante imediatamente posterior à palavra “pelo” porque são indispensáveis para a discussão de desempenho financeiro, embora o corpus (ainda) não contemple demonstrações financeiras.

A distribuição de palavras do CVMCorpus aparenta ser mais concentrada que a dos corpora em geral, sugerindo que as companhias preferem repetir termos consagrados em prol da inteligibilidade. Suas 50 palavras mais frequentes representam 35% do total de palavras e 489 tokens (0,014% do total dos tipos) são responsáveis por metade das ocorrências. A proporção das 50 palavras mais presentes no Corpus Brasileiro é 23%, enquanto uma parcela quase quatro vezes maior de tokens (0,05% do total ou 2.200 tipos) é necessária para representar metade das frequências.

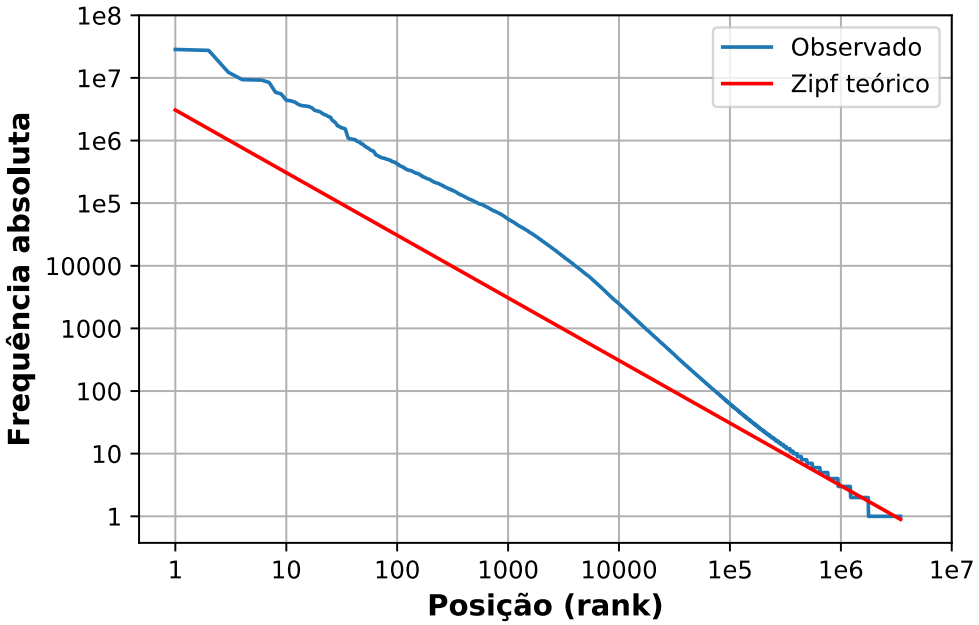
A concentração de palavras descrita no parágrafo anterior também pode ser verificada pela Lei de Zipf. A sua formulação mais simples prevê que a frequência absoluta de um token em um corpus é inversamente proporcional à sua posição no *ranking* de frequências absolutas dos tokens integrantes do mesmo corpus[37]. Portanto, a Lei de Zipf constata que a maioria dos tokens de um corpus tem baixa ocorrência e um número reduzido deles é muito frequente.

O Gráfico 8 mostra a comparação (em escala logarítmica) entre as frequências teórica, gerada pela aplicação da Lei de Zipf na formulação simples, e a observada no CVMCorpus. Observa-se que:

37 $f \cdot r = k$, onde f e r são as frequências absolutas e posições dos tokens, respectivamente.

- O ajuste da reta em geral não é satisfatório, mesmo nas posições intermediárias (que costumam ser mais aderentes à lei, em outros corpora), indicando que os tokens no topo e meio do *ranking* ocorrem com maior frequência no corpus que a prevista, isto é, a inclinação da melhor reta que se ajusta às observações é maior que um, em valor absoluto.
- No entanto, a cauda longa do CVMCorpus conforma-se bem aos resultados da lei, como descrito por [4]: quase a metade, isto é, 48,8% (1.686.229) dos tipos de tokens aparecem somente uma vez (i.e., *hapax legomena*) e 15,6% (547.208) dos átomos, duas vezes.

Gráfico 8 - Frequências absolutas téorica, calculada pela Lei de Zipf, e observada dos tokens do CVMCorpus, por posição (escala logarítmica)



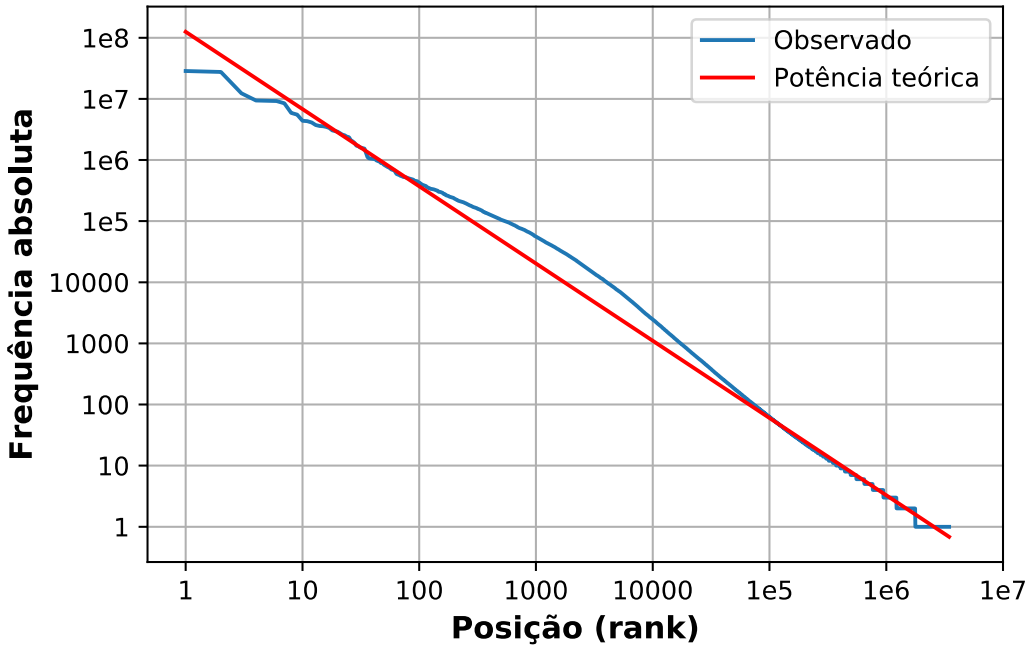
O Gráfico 9 apresenta o resultado da aplicação da Lei de Potência $f = b \cdot r^{-a}$. O coeficiente de determinação R^2 resultante é 0,976, denotando um alto poder de explicação da reta ajustada. Foi observado que:

- O coeficiente de inclinação **a** é 1,26 e é superior aos valores entre 1,02 e 1,03 obtidos com a coleção de corpora Leipzig[38], compostos de notícias coletadas na Internet.

38 https://cls.corpora.uni-leipzig.de/en/por_newscrawl_2016/3.6.1_Zipf's%20law%20(Standard%20version).html (consultado em 09.07.2020).

- Portanto, os tokens que são primeiros colocados no *ranking* de frequências possuem maiores frequências no CVMCorpus, do que as frequências, encontradas na coleção Leipzig, dos tokens que estão no topo do *ranking* da coleção.
- Isto reforça a observação anterior sobre a maior concentração de vocabulário do CVMCorpus, em linha com o resultado reportado por Sinclair, ao comparar um corpus de domínio e outro de referência geral [4].

Gráfico 9 - Frequências absolutas téorica, calculada pela Lei de Potência, e observada dos tokens do CVMCorpus, por posição (escala logarítmica)



6.5. Comparação com corpus de referência

Uma das formas de caracterização de um corpus é identificar as palavras-chave que o compõem. Palavras-chave são as palavras cujas frequências no corpus analisado são significativamente maiores, segundo testes estatísticos, que as frequências delas em outro corpus de escopo amplo, denominado corpus de referência. Assim, as palavras-chave potencialmente indicam os temas presentes no corpus em análise, e/ou seu vocabulário ou discurso típico.

Para analisar as palavras-chave do CVMCorpus, será utilizada a estatística LL (Log verossimilhança), recomendada por Dunning [99] para estudos linguísticos nos quais a premissa de normalidade não é válida. A estatística é calculada por meio da tabela de contingência a seguir (Tabela 27) e da fórmula (75). LL equivale à estatística do teste da razão de log verossimilhança, a qual segue distribuição qui-quadrado com um grau de liberdade.

Tabela 27 - Tabela de contingência para cálculo da estatística Log verossimilhança (LL)

Frequências	Corpus em análise	Corpus de referência	Total
Palavra analisada	a	b	a + b
Demais palavras	c - a	d - b	c + d - a - b
Total	c	d	c + d

Fonte: [100]

$$LL = 2\left[a \cdot \ln \frac{a(c+d)}{c(a+b)} + b \cdot \ln \frac{b(c+d)}{d(a+b)}\right] \quad (75)$$

A Tabela 28 exibe as vinte palavras do CVMCorpus que possuem conteúdo semântico e apresentam os maiores valores de LL. É possível observar que:

- Para todas as palavras, os valores de LL rejeitam a hipótese nula de as frequências absolutas no corpus de análise serem menores ou iguais às do corpus de referência, a um nível de significância inferior a 0,01% (valor crítico de 15,13).
- A maioria dos termos listados são característicos do mercado de capitais e referem-se a órgãos de administração de companhias e valores monetários, como “Geral” (Assembleia Geral), “capital” (capital social) e “remuneração”, além das palavras já discutidas anteriormente.
- A tabela também traz o efeito de cada palavra-chave, o qual indica quantas vezes a frequência relativa da palavra-chave é maior no CVMCorpus que no corpus de referência. O indicador apresenta maior magnitude para os termos específicos do mercado de valores mobiliários.

Tabela 28 - Palavras-chave do CVMCorpus (amostra)

Posição	Palavra	Frequência	LL	Efeito
2	Companhia	1.403.308	42128,7	42.446,5
5	Conselho	767.568	23043,2	23.217,0
6	ações	707.785	21248,4	21.408,7
8	Administração	679.090	20387,0	20.540,7
13	milhões	481.953	14468,7	14.577,8
14	exercício	423.604	12717,0	12.812,9
16	nº	377.786	11341,5	11.427,1
19	acionistas	326.137	9553,4	274,0
20	Geral	303.442	9109,6	9.178,3
21	Presidente	299.396	8988,2	9.056,0
23	valor	458.051	8694,3	8,6
27	remuneração	257.609	7733,7	7.792,0
28	Diretor	255.677	7675,7	7.733,6
29	São	237.768	7138,0	7.191,9
31	Artigo	233.176	7000,2	7.053,0
32	Paulo	229.474	6889,0	6.941,0
33	Assembleia	227.549	6831,2	6.882,8
35	dezembro	301.196	6571,4	13,1
38	capital	361.036	6517,5	7,6
39	Diretoria	216.959	6513,3	6.562,5

Posição: posição da palavra-chave na lista completa

6.6. Exemplo de aplicação

Para demonstrar uma potencial aplicação do CVMCorpus no campo linguístico, são analisados dois casos simples de colocações, envolvendo as palavras “perspectivas” e “cenário”. Colocações são duas ou mais palavras juntas que correspondem a uma forma convencional de expressar ideias ou coisas [101], p. ex., a expressão “vinho branco” que, na realidade, denomina um vinho de cor amarelada. Dividiram-se os documentos do corpus por ano de divulgação e o programa AntConc [102] analisou amostras que compreendiam de 70 a 90% dos textos dos anos de 2009, 2014, 2017, 2018 e 2019. Foram selecionadas as colocações que abrangiam duas palavras à esquerda e à direita do termo buscado. A análise assumiu grafia única, em minúsculas, para todas as palavras.

A Tabela 29 mostra as palavras que formam colocações com “perspectivas” com conotação positiva ou negativa, em ordem decrescente de valores de

Informação Mútua. Para mitigar a tendência da medida a favorecer termos raros, foram selecionados colocados com no mínimo 10 ocorrências na amostra em análise. Observa-se a prevalência de colocações com sentido positivo e altos escores de informação mútua em todos os anos (p.ex., "perspectivas positivas" apresenta o maior valor de 2017, 2018 e 2019). A única exceção é "desaceleração", registrada na amostra de 2018.

Tabela 29 - Colocações com conotação positiva ou negativa observadas com a palavra "perspectivas" no CVMCorpus

Ano: 2009

Pos.	Freq.	E	D	IM	Colocação
6	170	1	169	9,3	crescimento
9	17	0	17	8,7	rentabilidade

Ano: 2014

Pos.	Freq.	E	D	IM	Colocação
2	12	0	12	11,3	positivas
4	22	21	1	10,4	boas
12	224	2	222	8,6	crescimento
17	19	0	19	7,9	rentabilidade

Ano: 2017

Pos.	Freq.	E	D	IM	Colocação
1	17	0	17	12,1	positivas
4	18	16	2	10,1	boas
5	48	2	46	9,6	rentabilidade
9	110	0	110	8,3	crescimento

Ano: 2018

Pos.	Freq.	E	D	IM	Colocação
1	32	0	32	12,9	positivas
5	12	0	12	10,5	desaceleração
6	23	21	2	10,1	boas
8	34	0	34	8,9	rentabilidade
9	180	2	178	8,8	crescimento

Ano: 2019

Pos.	Freq.	E	D	IM	Colocação
1	29	0	29	13,4	positivas
2	29	28	1	10,7	boas
6	23	0	23	9,3	rentabilidade
7	92	3	89	8,8	crescimento
15	12	0	12	7,0	expansão

Pos.: posição nas tabelas completas de colocações.
Freq.: frequência absoluta total da colocação.
E: frequência absoluta com o termo colocado à esquerda de "perspectivas".
D: frequência absoluta com o termo colocado à direita de "perspectivas".
IM: informação mútua da colocação.

Portanto, a palavra "perspectivas" exibe prosódia semântica positiva, sugerindo que as companhias procuram caracterizar o futuro como promissor, já que as projeções de resultado vindouro são normalmente considerados os elementos mais importantes para a precificação de suas ações.

O resultado da análise das colocações do termo "cenário", segundo os mesmos critérios, é mostrado na Tabela 30. Foram identificadas 48 coocorrências com sentido negativo (em preto) e 17 colocações com conotação positiva (em azul). Pode-se notar que os casos com conotação negativa possuem maior informação mútua. Logo, o termo "cenário" tende a apresentar prosódia semântica negativa, sendo comumente empregado para justificar uma performance

insatisfatória recente da companhia. Embora também se possa prever um cenário futuro favorável, esse uso é bem menos frequente no corpus.

Tabela 30 - Colocações com conotação positiva ou negativa observadas com a palavra “cenário” no CVMCorpus

Ano: 2009

Pos.	Freq.	E	D	IM	Colocação
1	12	0	12	14,2	pessimista
4	12	6	6	12,4	deterioração
5	12	0	12	11,4	adverso
6	10	0	10	10,9	competitivo
14	13	3	10	8,5	favorável

Ano: 2014

Pos.	Freq.	E	D	IM	Colocação
4	41	1	40	12,2	desafiador
5	13	13	0	11,8	piora
9	40	4	36	11,4	adverso
10	52	26	26	11,3	deterioração
11	14	0	14	10,9	estresse
14	14	1	13	10,4	desfavorável
15	23	0	23	9,9	competitivo
21	10	7	3	8,5	crise
34	12	12	0	7,6	melhora
38	20	0	20	7,4	favorável

Ano: 2017

Pos.	Freq.	E	D	IM	Colocação
3	21	0	21	12,2	recessivo
4	118	9	109	11,9	adverso
5	100	5	95	11,9	desafiador
6	14	0	14	11,5	incerto
7	120	76	44	11,4	deterioração
13	42	0	42	10,6	desfavorável
14	24	23	1	10,5	piora
15	24	0	24	10,4	estresse
18	36	6	30	10,1	competitivo
21	15	1	14	9,3	recessão
23	55	53	2	9,0	melhora
36	12	0	12	7,5	retração
40	31	3	28	7,3	favorável
47	10	0	10	6,9	complexo
51	16	8	8	6,7	incertezas
57	30	28	2	6,2	melhor
65	10	3	7	5,8	negativo
118	11	4	7	3,7	recuperação
119	13	4	9	3,7	crescimento

Ano: 2018

Pos.	Freq.	E	D	IM	Colocação
3	105	3	102	12,1	desafiador
4	82	5	77	11,7	adverso
5	13	0	13	11,6	recessivo
9	13	0	13	11,0	incerto
13	58	39	19	10,6	deterioração
15	59	0	59	10,4	competitivo
16	16	14	2	10,3	piora
17	25	0	25	10,3	estresse
18	29	0	29	10,1	desfavorável
19	11	4	7	9,8	instabilidade
20	12	7	5	9,4	difícil
21	13	0	13	9,4	recessão
24	60	59	1	8,8	melhora
32	21	0	21	8,2	crise
39	11	1	10	7,7	retração
41	11	9	2	7,3	incerteza
51	14	7	7	6,4	incertezas
72	11	10	1	5,6	melhoria
78	14	2	12	5,5	favorável
82	18	17	1	5,4	melhor
104	14	1	13	4,8	expansão
129	17	5	12	3,9	crescimento
134	14	8	6	3,8	recuperação

Ano: 2019

Pos.	Freq.	E	D	IM	Colocação
2	43	0	43	12,4	desafiador
3	12	0	12	12,3	recessivo
7	28	2	26	10,5	adverso
12	10	7	3	9,7	dificuldades
21	12	8	4	7,9	incerteza
26	22	18	4	7,4	incertezas
35	13	5	8	6,7	favorável
45	13	0	13	5,9	expansão

Pos.: posição nas tabelas completas de colocações.
Freq.: frequência absoluta total da colocação.
E: frequência absoluta com o termo colocado à esquerda de “cenário”.
D: frequência absoluta com o termo colocado à direita de “cenário”.
IM: informação mútua da colocação.

7
Conclusão e trabalhos futuros

Este trabalho analisou e resolveu dois problemas de PLN aplicados à supervisão do mercado de capitais, com a utilização de algoritmos de aprendizado de máquina e uma base de documentos divulgados por companhias abertas brasileiras entre 2009 e 2019. O primeiro problema consistiu em classificar os documentos para identificar aqueles que continham informações relevantes sobre operações de reorganização societária e aumento de capital. O segundo problema compreendeu a extração de informações contidas nos mesmos documentos sobre as duas operações anteriores.

O problema de classificação foi abordado de duas formas. A primeira abordagem empregou diversos algoritmos clássicos de aprendizado de máquina e estudou o efeito da aplicação de diferentes formas de:

- Vetorização de tokens e documentos
- Critérios de seleção de atributos
- Técnicas de subamostragem
- Adoção de bigramas
- Inclusão de metadados

De modo geral, todos os fatores anteriores influenciaram o desempenho dos algoritmos na base desbalanceada de documentos, corroborando a necessidade de analisar a combinação de estratégias distintas por meio de múltiplos experimentos. O esforço resultou em um modelo de boa performance a um custo computacional aceitável. A montagem de comitês relativamente simples de votação e *stacking* com os modelos treinados proporcionou um incremento de performance

interessante, demonstrando a importância de explorar as combinações de modelos para buscar a opção de *ensemble* com a melhor relação entre viés e variância.

A segunda abordagem consistiu na aplicação de redes neurais profundas, as quais também obtiveram escores aceitáveis e acima do *baseline* proposto, porém inferiores aos dos algoritmos clássicos e tendo demandado maior esforço computacional.

Embora algumas particularidades dos documentos financeiros – como a presença de tabelas e outros elementos gráficos, além de textos longos – requereram pré-processamentos e cuidados adicionais, a natureza homogênea dos documentos em termos de estilo e vocabulário, já que são usualmente preparados por escritórios de advocacia ou departamentos de relações com investidores de companhias, pode ter contribuído para o bom desempenho dos algoritmos de aprendizado de máquina na resolução do problema de classificação.

O problema de extração de informações foi solucionado com a aplicação de uma rede neural de arquitetura BERT, tendo obtido resultados animadores para todas as informações desejadas. Alguns possíveis fatores do bom desempenho são:

- o fato de o domínio dos documentos ser restrito a assuntos financeiros de companhias abertas, a pré-seleção direcionada dos documentos;
- o estilo e vocabulário uniformes dos contextos e respostas; e
- a baixa complexidade do racional das respostas.

Dessa forma, a transferência do aprendizado contido no modelo multilíngue disponibilizado pelo Google foi conseguida e resultou em dois modelos especializados no problema, tudo isso a um custo de treinamento relativamente baixo.

Em suma, o presente trabalho apresentou e discutiu a aplicação de uma diversidade de algoritmos e técnicas de aprendizado de máquina a um problema real de supervisão de mercado de capitais, em linha com as experiências de reguladores mundiais discutidas no Capítulo 2. Insere-se no escopo pioneiro de iniciativas de incorporação de ferramentas de inteligência artificial às atividades da CVM. Foi demonstrado que o aprendizado de máquina é capaz de obter bom

desempenho em duas tarefas de processamento de linguagem natural executadas dentro de um domínio específico, como o financeiro, para atender às necessidades particulares da CVM, viabilizando assim a futura implementação dos modelos testados sob a forma de um sistema de apoio à decisão. O sistema permitirá um aumento de eficiência no processo de triagem de documentos, sobretudo por incremento da precisão sem haver perda significativa de *recall*, promovendo assim maior efetividade da pré-análise de documentos, a qual hoje é totalmente manual.

Para trabalhos futuros, as abordagens dos problemas podem ser aprofundadas de diversas formas. Por exemplo, no caso de algoritmos clássicos, é possível testar o efeito de:

- normalizar palavras adotando a stemmização ou a lematização;
- alterar os parâmetros do algoritmo de detecção de bigramas;
- testar o uso de trigramas;
- buscar uma quantidade ótima de atributos utilizando os critérios aqui discutidos ou outros;
- aplicar representações vetoriais alternativas, como ELMo e Fasttext;
- testar novos algoritmos de classificação;
- construir comitês do tipo *bagging*, *boosting* e;
- incorporar níveis adicionais de metamodelos sobre predições de outros metamodelos.

A classificação por redes neurais tampouco foi esgotada neste trabalho, havendo espaço para novos experimentos com redes que combinam duas ou mais arquiteturas, comitê de redes e arquiteturas mais complexas, como o próprio BERT.

O problema proposto de extração de informações pode ser atacado alternativamente com várias arquiteturas de rede derivadas ou concorrentes do BERT, por exemplo, RoBERTa, ALBERT, DMM e BiDAF. Além disso, seria interessante testar a performance humana para confirmar a efetividade do modelo.

A CVM e outros órgãos reguladores e fiscalizadores da administração pública podem se beneficiar de trabalhos futuros similares, pois recebem

diariamente volumes significativos de texto não estruturado que precisam ser analisados. Diante de um quadro de escassez de recursos humanos e crescimento do mercado de capitais, torna-se imperativo investir em mecanismos automatizados de triagem de documentos, análise de completude, consistência e conformidade com as normas vigentes, além de cruzamento com informações de outros textos ou dados estruturados. Portanto, é fácil notar que os dois problemas aqui tratados representam uma pequena parte do universo de tarefas de processamento de linguagem natural aplicáveis às atividades de supervisão e fiscalização.

Finalmente, o presente trabalho traz também como contribuição um corpus especializado no domínio financeiro – o CVMCorpus. Ele representa em boa medida a linguagem empregada pelas maiores empresas do capitalismo nacional e abrange uma parte relevante das divulgações realizadas por todas as companhias abertas brasileiras e estrangeiras registradas na CVM. Assim, o CVMCorpus fornece um retrato (*snapshot* corpus) da linguagem pública corporativa entre 2009 e 2019, podendo ser considerado uma fonte de pesquisa para as áreas linguística e financeira, como definido por Sinclair[39] [4].

Do ponto de vista linguístico e financeiro, numerosos tipos de pesquisas futuras são possíveis com o CVMCorpus. Pode-se, por exemplo, estudar as particularidades do discurso financeiro, sob a forma de questões como:

- Quais as colocações prevalentes?
- O discurso das companhias é completamente neutro e moderado ou existem nele elementos de otimismo exagerado ou eufemismos?
- Existem outras ocorrências do fenômeno da prosódia semântica, além daquelas aqui discutidas?

Do lado financeiro, espera-se que o corpus sirva de base para estudos futuros sobre linguagem contábil, indicadores antecedentes de fraudes e irregularidades presentes no conteúdo de relatórios e informes, além da relação entre discurso empregado nas divulgações e desempenho de ações.

39 "A corpus is a collection of pieces of language text in electronic form, selected according to external criteria to represent, as far as possible, a language or language variety as a source of data for linguistic research."

8
Referências bibliográficas

1 JURAFSKY, D.; MARTIN, J. H. **Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition**. 2nd ed. Upper Saddle River, N.J: Pearson Prentice Hall, 2009.

2 LEWIS, C.; YOUNG, S. Fad or future? Automated analysis of financial text and its implications for corporate reporting. **Accounting and Business Research**, v. 49, n. 5, p. 587–615, 29 jul. 2019.

3 LIU, H.; GEGOV, A.; COCEA, M. **Rule Based Systems for Big Data**. Cham: Springer International Publishing, 2016. v. 13

4 SINCLAIR, J. Corpus and Text – Basic Principles. *In:* WYNNE, M. (Ed.). **Developing Linguistic Corpora: a Guide to Good Practice**. Oxford: Oxbow Books, 2006. p. 1–16.

5 XIAO, R. Corpus Creation. *In:* **Handbook of Natural Language Processing**. 2nd. ed. Boca Raton, FL: Chapman & Hall/CRC, 2010. p. 147–162.

6 MCENERY, T.; HARDIE, A. **Corpus linguistics**: method, theory and practice. Cambridge ; New York: Cambridge University Press, 2012.

7 COHEN, K. B. et al. The Colorado Richly Annotated Full Text (CRAFT) Corpus: Multi-Model Annotation in the Biomedical Domain. *In:* IDE, N.; PUSTEJOVSKY, J. (Eds.). **Handbook of Linguistic Annotation**. Dordrecht: Springer Netherlands, 2017. p. 1379–1394.

8 THOMPSON, P.; ANANIADOU, S.; TSUJII, J. The GENIA Corpus: Annotation Levels and Applications. *In:* IDE, N.; PUSTEJOVSKY, J. (Eds.). **Handbook of Linguistic Annotation**. Dordrecht: Springer Netherlands, 2017. p. 1395–1432.

9 LOPES, L. **Extração Automática de Conceitos a partir de Textos em Língua Portuguesa**. Porto Alegre: PUC-RS, 2012.

10 KOGAN, S. et al. Predicting risk from financial reports with regression. Proceedings of Human Language Technologies. *In:* HUMAN LANGUAGE TECHNOLOGIES: THE 2009 ANNUAL CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Anais**... Boulder, Colorado: Association for Computational Linguistics, 2009. Disponível em: <http://portal.acm.org/citation.cfm?doid=1620754.1620794>. Acesso em: 28 jun. 2020

11 HUMPHERYS, S. L. et al. Identification of fraudulent financial statements using linguistic credibility analysis. **Decision Support Systems**, v. 50, n. 3, p. 585–594, fev. 2011.

12 LEE, H. et al. On the Importance of Text Analysis for Stock Price Prediction. *In:* NINTH INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION (LREC'14). **Anais**... Reykjavik: Association for Computational Linguistics, 2014, p. 1170-1175.

13 DAUDERT, T.; AHMADI, S. **CoFiF**: A Corpus of Financial Reports in French Language. Proceedings of the First Workshop on Financial Technology and Natural Language Processing. **Anais…** Macao, China: ago. 2019

14 EL-HAJ, M. et al. Retrieving, classifying and analysing narrative commentary in unstructured (glossy) annual reports published as PDF files. **Accounting and Business Research**, v. 50, n. 1, p. 6–34, 2 jan. 2020.

15 GARCIA, E.; BIDARRA, J. Uma Contribuição Ao Estudo Do Fenômeno Da Ambiguidade Lexical Na Atividade Contábil. **Revista Ciências Sociais em Perspectiva**, v. 16, n. 30, 2017.

16 AGUIAR, M. **Sentiment Analysis em Relatórios de Administração Divulgados por Firmas Brasileiras**. Vitória, PE: Fundação Instituto Capixaba de Pesquisas em Contabilidade, Economia e Finanças – FUCAPE, 2012.

17 ALVES, D. S. **Uso de técnicas de Computação Social para tomada de decisão de compra e venda de ações no mercado brasileiro de bolsa de valores**. 2015. Tese (Doutorado em Engenharia de Sistemas Eletrônicos e Automação) – Universidade de Brasília, Brasília, 23 out. 2015.

18 EL-HAJ, M. et al. In search of meaning: Lessons, resources and next steps for computational analysis of financial discourse. **Journal of Business Finance & Accounting**, v. 46, n. 3–4, p. 265–306, mar. 2019.

19 KOSHIYAMA, A. S.; FIROOZYE, N.; TRELEAVEN, P. Algorithms in Future Capital Markets. **SSRN Electronic Journal**, 2020.

20 MORENO, A.; REDONDO, T. Text Analytics: the convergence of Big Data and Artificial Intelligence. **International Journal of Interactive Multimedia and Artificial Intelligence**, v. 3, n. 6, p. 57, 2016.

21 BACH, M. et al. Text Mining for Big Data Analysis in Financial Sector: A Literature Review. **Sustainability**, v. 11, n. 5, p. 1277, 28 fev. 2019.

22 MCCLANE, J. Regulating Substance Through Form: Lessons from the SEC's Plain English Initiative. **Harvard Journal on Legislation**, University of Illinois College of Law Legal Studies Research Paper. v. 55, n. No. 265, 10 jun. 2018.

23 BADAWI, A. B. **How Informative Is the Text of Securities Complaints?**, 30 set. 2019.

24 BROWN, N. C.; ELLIOTT, W. B.; CROWLEY, R. What are You Saying? Using Topic to Detect Financial Misreporting. **SSRN Electronic Journal**, 2016.

25 HOBERG, G.; LEWIS, C. Do fraudulent firms produce abnormal disclosure? **Journal of Corporate Finance**, v. 43, p. 58–85, abr. 2017.

26 LI, A.; WU, J.; LIU, Z. Market Manipulation Detection Based on Classification Methods. **Procedia Computer Science**, v. 122, p. 788–795, 2017.

27 WANG, Q.; XU, W.; HUANG, X.; YANG, K. Enhancing intraday stock price manipulation detection by leveraging recurrent neural networks with ensemble learning. **Neurocomputing**, v. 347, p. 46–58, jun. 2019.

28 DONG, W.; LIAO, S.; LIANG, L. Financial statement fraud detection using text mining. 20TH PACIFIC ASIA CONFERENCE ON INFORMATION SYSTEMS (PACIS 2016). *In*: **Anais**... Chiayi, Taiwan. 2016.

29 ARACI, D. FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. **arXiv:1908.10063 [cs]**, 27 ago. 2019.

30 HIEW, J. Z. G. et al. BERT-based Financial Sentiment Index and LSTM-based Stock Return Predictability. **arXiv:1906.09024 [q-fin]**, 21 jun. 2019.

31 FINANCIAL STABILITY BOARD. **Artificial intelligence and machine learning in financial services: Market developments and financial stability implications**. 01 nov. 2017. [S. l.: s. n.]. Disponível em: <https://www.fsb.org/wp-content/uploads/P011117.pdf>.

32 FINANCIAL STABILITY BOARD. **The Use of Supervisory and Regulatory Technology by Authorities and Regulated Institutions: Market developments and financial stability implications**. 09 out. 2020. [S.l: s.n.]. Disponível em: <https://www.fsb.org/wp-content/uploads/P091020.pdf>.

33 COELHO, R.; DI SIMONI, M.; PRENIO, J. **Suptech applications for anti-money laundering**: FSI Insights on policy implementation. [s.l.] Financial Stability Institute, ago. 2019. Disponível em: <https://www.bis.org/fsi/publ/insights18.pdf>.

34 HEUVER, R.; TRIEPELS, R. Liquidity Stress Detection in the European Banking Sector. **DNB Working Paper**, v. 642, p. 14, jun. 2019.

35 PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, n. 85, p. 2825–2830, 2011.

36 MIKOLOV, T. et al. Efficient Estimation of Word Representations in Vector Space. **arXiv:1301.3781 [cs]**, 6 set. 2013.

37 MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to information retrieval**. New York: Cambridge University Press, 2008.

38 MASINI, R. P.; MEDEIROS, M. C.; MENDES, E. F. Machine Learning Advances for Time Series Forecasting. **arXiv:2012.12802 [cs, econ, stat]**, 9 abr. 2021.

39 WITTEN, I. H. et al. **Data mining: practical machine learning tools and techniques**. Fourth Edition ed. Amsterdam: Elsevier, 2017.

40 HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H. **The elements of statistical learning: data mining, inference, and prediction**. 2nd ed. New York, NY: Springer, 2009.

41 JAMES, G. et al. **An introduction to statistical learning: with applications in R**. New York: Springer, 2013.

42 AGGARWAL, C. C. **Machine learning for text**. 1st ed. New York, NY: Springer Science+Business Media, 2018.

43 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge, Massachusetts: The MIT Press, 2016.

44 KOWSARI, K. et al. Text Classification Algorithms: A Survey. **arXiv:1904.08067 [cs, stat]**, 20 maio 2020.

45 AGGARWAL, C. C.; ZHAI, C. A Survey of Text Classification Algorithms. *In:* AGGARWAL, C. C.; ZHAI, C. (Eds.). **Mining Text Data**. Boston, MA: Springer US, 2012. p. 163–222.

46 TIBSHIRANI, R. Regression Shrinkage and Selection Via the Lasso. **Journal of the Royal Statistical Society**: Series B (Methodological), v. 58, n. 1, p. 267–288, jan. 1996.

47 HOERL, A. E.; KENNARD, R. W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. **Technometrics**, v. 12, n. 1, p. 55–67, fev. 1970.

48 ZOU, H.; HASTIE, T. Regularization and variable selection via the elastic net. **Journal of the Royal Statistical Society**: Series B (Statistical Methodology), v. 67, n. 2, p. 301–320, abr. 2005.

49 FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. Regularization Paths for Generalized Linear Models via Coordinate Descent. **Journal of Statistical Software**, v. 33, n. 1, 2010.

50 RENNIE, J. D. M. et al. **Tackling the Poor Assumptions of Naive Bayes Classifiers**. *In:* ICML. T. Fawcett & N. Mishra, 2003

51 COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n. 1, p. 21–27, jan. 1967.

52 TIBSHIRANI, R.; HASTIE, T.; NARASIMHAN, B.; CHU, G. Diagnosis of multiple cancer types by shrunken centroids of gene expression. **Proceedings of the National Academy of Sciences**, v. 99, n. 10, p. 6567–6572, 14 maio 2002.

53 CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, [S.l] Springer, set. 1995.

54 KUHN, H. W.; TUCKER, A. W. **Nonlinear programming**. *In:* PROCEEDINGS OF THE SECOND BERKELEY SYMPOSIUM ON MATHEMATICAL STATISTICS AND PROBABILITY. **Anais…** Berkeley and Los Angeles: University of California Press, 1951

55 BREIMAN, L. Bagging Preditors. **Machine Learning**, v. 24, n. 2, p. 123–140, [S.l], Springer, 1996.

56 BREIMAN, L. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, [S.l,], Springer, 2001.

57 FREUND, Y.; SCHAPIRE, R. E. **Experiments with a New Boosting Algorithm**. *In:* PROCEEDINGS OF THE 13TH INTERNATIONAL CONFERENCE ON INTERNATIONAL CONFERENCE ON MACHINE LEARNING. **Anais**... ICML'96. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996.

58 FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. **The Annals of Statistics**, v. 29, n. 5, p. 1189–1232, out. 2001.

59 ZHOU, Z.-H. **Ensemble methods**: foundations and algorithms. Boca Raton, Fla.: CRC Press/Taylor & Francis, 2012.

60 TING, K. M.; WITTEN, I. H. Issues in Stacked Generalization. **Journal of Artificial Intelligence Research**, v. 10, p. 271–289, 1 maio 1999.

61 SEEWALD, A. **How to Make Stacking Better and Faster While Also Taking Care of an Unknown Weakness**. *In*: PROCEEDINGS OF THE 19TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING. **Anais**… (*S.l.*) Morgan Kaufmann Publishers Inc., jul. 2002.

62 HAYKIN, S. S. **Neural networks and learning machines**. 3rd ed. New York: Prentice Hall, 2009.

63 BENGIO, Y. Practical recommendations for gradient-based training of deep architectures. **arXiv:1206.5533 [cs]**, 16 set. 2012.

64 HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1 nov. 1997.

65 BAHRUDIN, H. **In depth LSTM Implementation using CNTK on .NET platform.** Bahrudin Hrnjica Blog. Disponível em: <https://hrnjica.net/2019/04/08/in-depth-lstm-implementation-using-cntk-on-net-platform/>. Acesso em: 30 maio 2021

66 DUAN, W. et al. An Improved Gated Recurrent Unit Network Model for State-of-Charge Estimation of Lithium-Ion Battery. **Energies**, v. 13, n. 23, p. 6366, MDPI, 3 dez. 2020.

67 CHO, K. et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. **arXiv:1406.1078 [cs, stat]**, 2 set. 2014.

68 CHUNG, J. et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. **arXiv:1412.3555 [cs]**, 11 dez. 2014.

69 GREFF, K. et al. LSTM: A Search Space Odyssey. **arXiv:1503.04069 [cs]**, 4 out. 2017.

70 Keras Conv1D: Working with 1D Convolutional Neural Networks in Keras. **missinglink.ai**. Disponível em: <https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/>. Acesso em: 18 jan. 2021

71 BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural Machine Translation by Jointly Learning to Align and Translate. **arXiv:1409.0473 [cs, stat]**, 19 maio 2016.

72 VASWANI, A. et al. Attention Is All You Need. **arXiv:1706.03762 [cs]**, 5 dez. 2017.

73 ALAMMAR, J. **The Illustrated TransformerJay Alammar**. Disponível em: <https://jalammar.github.io/illustrated-transformer/>. Acesso em: 27 jan. 2021

74 HE, K. et al. Deep Residual Learning for Image Recognition. **arXiv:1512.03385 [cs]**, 10 dez. 2015.

75 DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **arXiv:1810.04805 [cs]**, 24 maio 2019.

76 FERNÁNDEZ, A. et al. **Learning from Imbalanced Data Sets**. 1st ed. 2018 ed. Cham: Springer International Publishing : Imprint: Springer, 2018.

77 ZHANG, J.; MANI, I. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. *In*: PROCEEDINGS OF THE ICML'2003 WORKSHOP ON LEARNING FROM IMBALANCED DATASETS. **Anais**... ICML'2003 Workshop On Learning From Imbalanced Datasets. Washington, DC: 21 ago. 2003

78 TOMEK, I. Two Modifications of CNN. **IEEE Transactions on Systems, Man, and Cybernetics**, v. SMC-6, n. 11, p. 769–772, nov. 1976.

79 HART, P. The condensed nearest neighbor rule (Corresp.). **IEEE Transactions on Information Theory**, v. 14, n. 3, p. 515–516, maio 1968.

80 LEMAÎTRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. **Journal of Machine Learning Research**, v. 18, n. 17, p. 1–5, 2017.

81 KUBAT, M.; MATWIN, S. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. *In:* PROCEEDINGS OF THE FOURTEENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING. **Anais**… p. 179-186, Morgan Kaufmann, jul. 1997.

82 MUZART, E. State-of-the-art Multilingual Lemmatization. **towards data science**. Acesso em: 13 fev. 2021

83 VIERA, A. F. G.; VIRGIL, J. Uma revisão dos algoritmos de radicalização em língua portuguesa. **Information Research**, v. 12(3), p. 315, 2007.

84 NIERADZIK, L. Portuguese Lemmatizers (2020 update). **My personal blog: Machine learning, computer vision, languages**. Disponível em: <https://lars76.github.io/2018/05/08/portuguese-lemmatizers.html>. Acesso em: 13 fev. 2021.

85 REHUREK, R.; SOJKA, P. **Software Framework for Topic Modelling with Large Corpora**. . *In:* PROCEEDINGS OF THE LREC 2010 WORKSHOP ON NEW CHALLENGES FOR NLP. Valletta, Malta: ELRA, 22 maio 2010.

Disponível em: <http://rgdoi.net/10.13140/2.1.2393.1847>. Acesso em: 15 fev. 2021

86 CHEN, T.; GUESTRIN, C. XGBoost: A Scalable Tree Boosting System. *In:* PROCEEDINGS OF THE 22ND ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING. **Anais**... *In:* KDD'16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining. San Francisco California USA: ACM, 13 ago. 2016. Disponível em: <https://dl.acm.org/doi/10.1145/2939672.2939785>. Acesso em: 17 fev. 2021

87 TSYMBAL, A.; PECHENIZKIY, M.; CUNNINGHAM, P. Diversity in search strategies for ensemble feature selection. **Information Fusion**, v. 6, n. 1, p. 83–98, mar. 2005.

88 TUMER, K.; GHOSH, J. Error Correlation and Error Reduction in Ensemble Classifiers. **Connection Science**, v. 8, n. 3–4, p. 385–404, dez. 1996.

89 CHOLLET, F. **Keras GitHub repository**. GitHub. 2015. Disponível em: <https://github.com/fchollet/keras>

90 ZULQARNAIN, M. et al. A comparative review on deep learning models for text classification. **Indonesian Journal of Electrical Engineering and Computer Science**, v. 19, n. 1, p. 325, 1 jul. 2020.

91 YIN, W.; KANN, K.; YU, M.; SCHÜTZE, H. Comparative Study of CNN and RNN for Natural Language Processing. **arXiv:1702.01923 [cs]**, 7 fev. 2017.

92 ZHAO, Y.; SHEN, Y.; YAO, J. Recurrent Neural Network for Text Classification with Hierarchical Multiscale Dense Connections. *In:* PROCEEDINGS OF THE TWENTY-EIGHTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE. **Anais…** Macao, China: International Joint Conferences on Artificial Intelligence Organization, ago. 2019. Disponível em: <https://www.ijcai.org/proceedings/2019/757>. Acesso em: 25 maio. 2021

93 MINAEE, S. et al. Deep Learning--based Text Classification: A Comprehensive Review. **ACM Computing Surveys**, v. 54, n. 3, p. 1–40, maio 2021.

94 QI, P. et al. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. *In:* PROCEEDINGS OF THE 58TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: SYSTEM DEMONSTRATIONS. **Anais**... Online: Association for Computational Linguistics, 2020. Disponível em: <https://www.aclweb.org/anthology/2020.acl-demos.14>. Acesso em: 9 jun. 2021

95 RAJPURKAR, P.; ZHANG, J.; LOPYREV, K.; LIANG, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. **arXiv:1606.05250 [cs]**, 10 out. 2016.

96 LOSHCHILOV, I.; HUTTER, F. Decoupled Weight Decay Regularization. **arXiv:1711.05101 [cs, math]**, 4 jan. 2019.

97 JOULIN, A.; GRAVE, E.; BOJANOWSKI, P.; MIKOLOV, T. Bag of Tricks for Efficient Text Classification. **arXiv:1607.01759 [cs]**, 9 ago. 2016.

98 BICK, E. **The parsing system "Palavras": automatic grammatical analysis of Portuguese in a constraint grammar framework**. Aarhus [Denmark]; Oakville, Conn: Aarhus University Press, 2000.

99 DUNNING, T. Accurate Methods for the Statistics of Surprise and Coincidence. **Computational Linguistics**, v. 19, n. 1, p. 61–74, mar. 1993.

100 CANTOS GÓMEZ, P. **Statistical methods in language and linguistic research**. Oakville, CT: Equinox Pub. Ltd, 2012.

101 MANNING, C. D.; SCHÜTZE, H. **Foundations of statistical natural language processing**. Cambridge, Mass: MIT Press, 1999.

102 LAWRENCE, A. AntConc 3.5.9. **Lawrence Anthony's Website**, 2020. Disponível em: <https://www.laurenceanthony.net/software>. Acesso em: 21 jul. 2021

Apêndice 1 – Convenções adotadas de atomização

Considerou-se como um átomo ou token:

- R, US com $ e os valores subsequentes.
- Sequências máximas de dígitos separadas por ponto ou vírgula são consideradas números.
- Números seguidos de %.
- EUR, USD, ARS, GBP, CAD, CNY, JPY, £, €, $, ¥ e § com números subsequentes.
- Alíneas formadas por números arábicos, números romanos ou letras, seguidos de ponto, quando iniciam parágrafos.
- Alíneas formadas por números arábicos, números romanos ou letras, seguidos de) ou hífen.
- Alíneas formadas por números arábicos seguidos de º, .º, ª, .ª, .o, .os, .a ou .as, quando iniciam parágrafos.
- Sequências máximas de !, ? e reticências.
- (…) e […].
- URLs identificados.
- Duas palavras entre & ou + que se encontravam unidas no texto.
- Números de telefone, com código de área e país.
- Alíneas formadas por números arábicos, números romanos ou letras entre parênteses.
- Endereços de e-mail identificados.
- Números separados por /.
- Sequências de números entre parênteses, mesmo que haja pontuação entre eles.
- Sinais de pontuação seguidos de espaço.
- Abreviaturas finalizadas com apenas um ponto que terminam frases.
- Palavras ligadas por hífen ou /.

Devido a falhas ocorridas na conversão de PDF em texto e para facilitar a leitura pelo sistema de apoio à decisão para o qual o corpus foi construído, os seguintes ajustes e uniformizações se fizeram necessários:

- Transformar " (dois apóstrofos), ``, “, ” , « e » em " (aspas ASCII).
- Transformar ‘, ’ e ′ em '.

Apêndice 2 – Hiperparâmetros testados dos algoritmos clássicos

As tabelas contêm os nomes dados pelas bibliotecas aos hiperparâmetros.

SVM com kernel radial

Nome	Descrição	Valores experimentados
gamma	γ de (46)	0.05, 0.1; 0.5, 1, 5 + 5 aleatórios entre 0.01 e 10
C	1/C é o grau de regularização	1, 5, 10 + 5 aleatórios entre 0.1 e 100

SVM com kernel polinomial

Nome	Descrição	Valores experimentados
Grau do polinômio	d de (45)	1, 2 e 3
C	1/C é o grau de regularização	0.5, 1, 2.5, 5 + 20 aleatórios entre 0.1 e 10

K-Neighbors

Nome	Descrição	Valores experimentados
n_neighbors	Quantidade de observações vizinhas consideradas	2 a 9
weights	Forma de ponderação das distâncias	Uniforme e inverso da distância
metric	Tipo de distância	Euclidiana

Regressão Elastic net

Nome	Descrição	Valores experimentados
alpha	α de (25)	0 a 1, a cada 0.1, 0.001 e 0.999
n_lambda	Nº de valores de λ testados	800 e 1.000
min_lambda_ratio	Razão entre λ máximo e mínimo testados	10^{-6}
tol	Tolerância de convergência	10^{-4}

Regressão logística

Nome	Descrição	Valores experimentados
solver	Algoritmo de otimização	newton-cg, saga, sag e lbgfs

Naive Bayes multinomial e complementar

Nome	Descrição	Valores experimentados
alpha	Parâmetro de suavização aditiva	0.001, 0.005, 0.01, 0.05, 0.1, 0.5 e 1

Árvores *boosted*

Nome	Descrição	Valores experimentados
n_estimators	Nº de árvores construídas	10 a 199
max_depth	Profundidade máxima	4 a 20
min_child_weight	Soma mínima requerida de pesos[40] para ramificar	1 a 10
subsample	% observações sorteadas para treinar cada árvore	0.5, 0.6, 0.7, 0.8 e 0.9
colsample_bytree	% atributos sorteados para construir uma árvore	0.5, 0.6, 0.7, 0.8 e 0.9
colsample_bylevel[41]	% atributos sorteados ao atingir um novo nível de profundidade	0.5, 0.6, 0.7, 0.8 e 0.9
colsample_bynode[42]	% atributos sorteados para uma ramificar	0.5, 0.6, 0.7, 0.8 e 0.9
objective	função objetivo	Softmax e Softprob para classificação multiclasse
eta	% redução da contribuição de árvores individuais	0.01, 0.015, 0.025, 0.05, 0.1, 0.2, 0.25, 0.275, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45, 0.475 e 0.5
gamma	% mínimo aceitável de redução da função perda para ramificar	0.05, 0.075, 0.1, 0.3, 0.5, 0.7, 0.9 e 1.0
lambda	termo de regularização L2 aplicada aos pesos das folhas	0, 0.01, 0.05, 0.1, 1 + 20 aleatórios entre 0.01 e 1
alpha	termo de regularização L1 aplicada aos pesos das folhas	0, 0.1, 0.5, 1 + 20 aleatórios entre 0.1 e 1

40 O algoritmo atribui pesos às observações e a sua soma é a medida da pureza de um nó.
41 Percentual sobre os atributos sorteados para a árvore (pós-aplicação de colsample_bytree).
42 Percentual sobre os atributos sorteados para o nível (pós-aplicação de colsample_bylevel).

Random Forest

Nome	Descrição	Valores experimentados
n_estimators	Nº de árvores construídas	300, 400, 500, 550 e 600
max_features	Nº de atributos analisados para ramificação	(nº atributos)$^{1/2}$ e int[(0.025, 0.05, 0.1, 0.2, 0.25)*nº atributos]
min_samples_leaf	Nº mínimo de observações de um nó terminal	1, 2, 3, 4 e 5
criterion	Critério de ramificação	Gini e Entropia
max_samples	% observações sorteadas para treinar cada árvore	0.2, 0.275, 0.35, 0.425, 0.5, 0.575, 0.65, 0.7215 e 0.8

Nearest Shrunken Centroids

Nome	Descrição	Valores experimentados
metric	Métrica de distância	Euclidiana e Cosseno
shrink_threshold	Δ de (36)	0.1 a 100.0, a cada 0.1

Análise Discriminante Linear (LDA)

Nome	Descrição	Valores experimentados
solver	Algoritmo de otimização	SVD e LSQR
shrinkage	γ de (43)	Automática Escala log: 10^{-6} a 1, $5{\cdot}10^{-6}$ a 0.5, Escala linear: $1{\cdot}10^{-5}$ a $9{\cdot}10^{-5}$ e $1{\cdot}10^{-4}$ a $9{\cdot}10^{-4}$

Análise Discriminante Quadrática (QDA)

Nome	Descrição	Valores experimentados
reg_param	β de (42)	Escala log: 10^{-6} a 1, $5{\cdot}10^{-6}$ a 0.5, Escala linear: $1{\cdot}10^{-3}$ a $9{\cdot}10^{-3}$ e $1{\cdot}10^{-4}$ a $9{\cdot}10^{-4}$

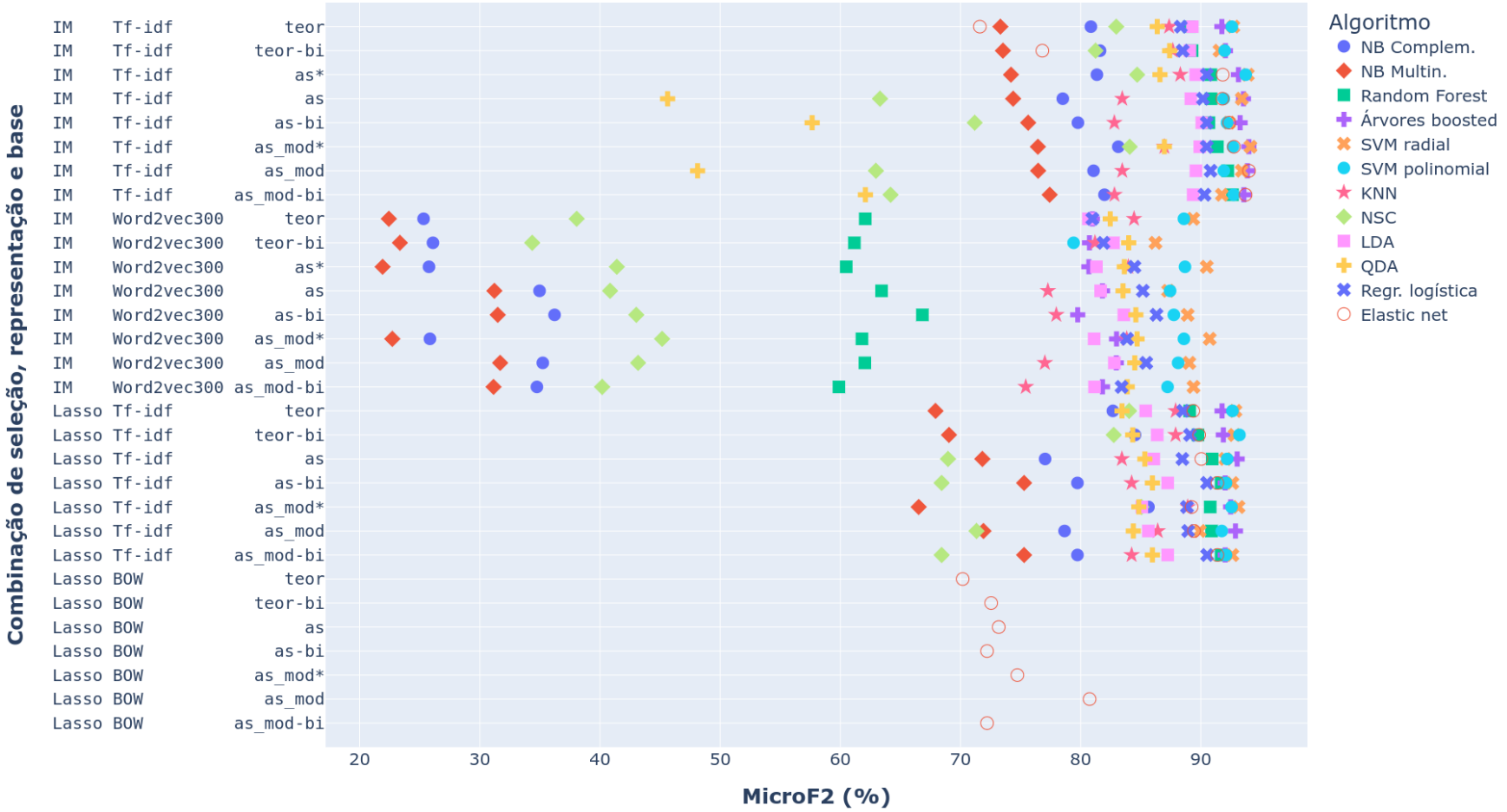
Apêndice 3 – Hiperparâmetros empregados nos testes de subamostragem

Todas os testes efetuaram subamostragem apenas sobre os documentos da classe dominante “n”.

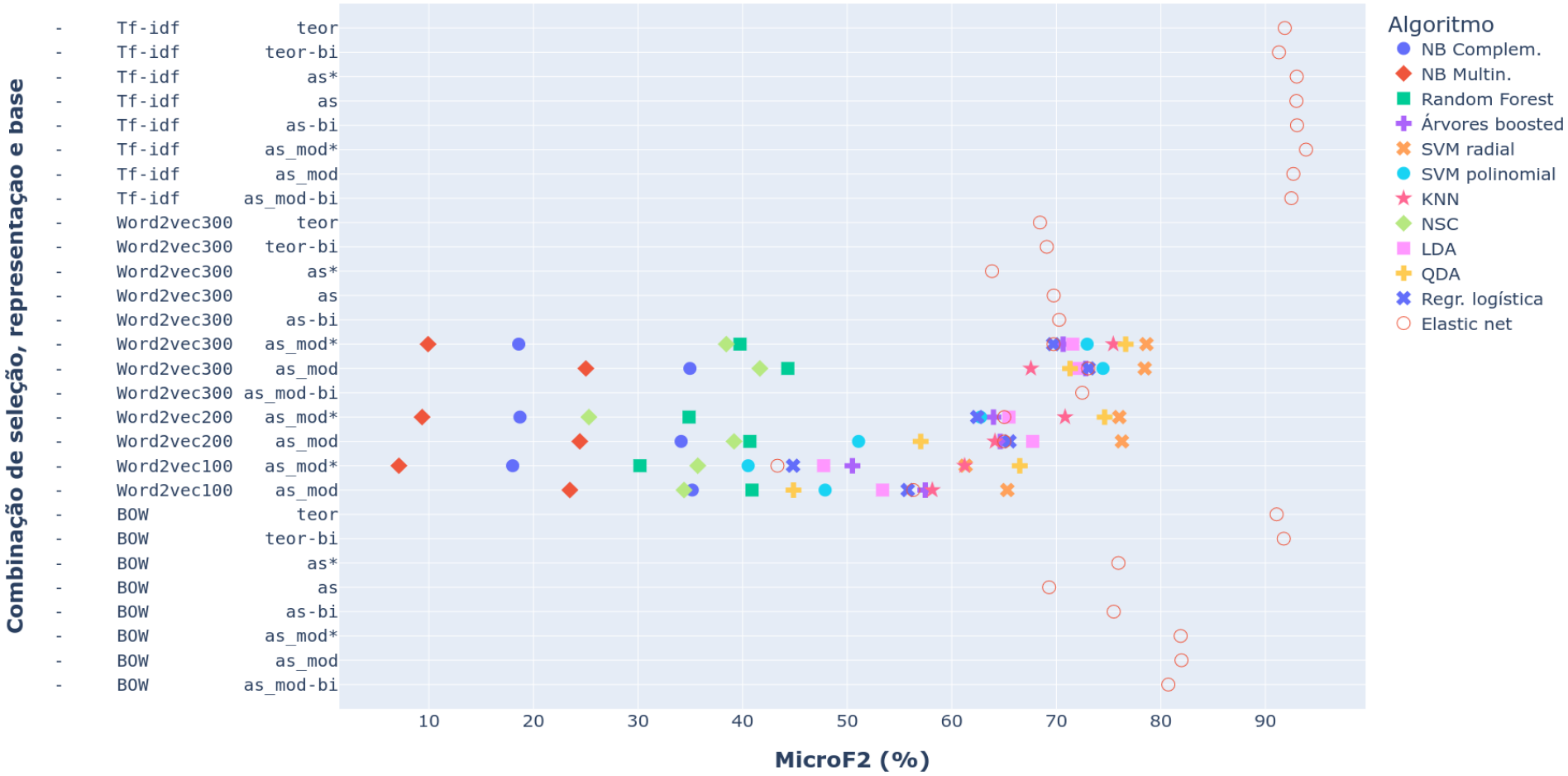
Técnica	Hiperparâmetro	Valor adotado
NearMiss-1	Nº de observações positivas para calcular distâncias (n)	3
NearMiss-2	Nº de observações positivas para calcular distâncias (n)	3
NearMiss-3	Nº de observações positivas para calcular distâncias (n)	3
	Nº de observações negativas amostradas (m)	3
OSS	Nº de observações vizinhas para calcular distâncias pelo KNN (k)	1
	Nº de sementes (s)	1
CNN	Nº de observações vizinhas para calcular distâncias pelo KNN (k)	1
	Nº de sementes (s)	20

OSS: One-Sided Selection
CNN: Condensed Nearest Neighbor

Apêndice 4 – Gráfico dos escores microF2 do primeiro experimento



Apêndice 4 – Gráfico dos escores microF2 do primeiro experimento (cont.)



Apêndice 5 – Resultados do primeiro experimento

a. Seleção de atributos pelo critério combinado IM+χ^2

Base teor - apenas texto dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	80,85	91,54	65,21	86,00	74,18
Multinomial NB	2000	Tf-idf	73,32	93,59	81,54	71,52	76,20
RandomForest	2000	Tf-idf	89,04	97,26	92,57	88,20	90,33
Árvores boosted	2000	Tf-idf	91,74	97,79	92,89	91,46	92,17
SVM radial	2000	Tf-idf	**92,70**	98,03	93,81	92,43	93,11
SVM polinomial	2000	Tf-idf	92,58	98,06	94,18	92,19	93,17
KNN	2000	Tf-idf	87,36	96,88	91,01	86,49	88,69
Nearest S. Centroid	2000	Tf-idf	82,96	90,63	61,81	90,72	73,52
LDA	2000	Tf-idf	89,29	97,04	90,41	89,02	89,71
QDA	2000	Tf-idf	86,37	92,38	67,02	93,08	77,93
Regressão Logística	2000	Tf-idf	88,33	96,71	88,50	88,28	88,39
Elastic net	2000	Tf-idf	91,07	97,66	92,83	90,64	91,72
Elastic net	Todos	BOW	71,61	94,35	90,40	68,07	77,66
Elastic net	Todos	Tf-idf	91,85	97,93	94,06	91,31	92,66

Base teor bi - apenas texto dos documentos, com bigramas

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	81,61	92,14	67,56	86,09	75,71
Multinomial NB	2000	Tf-idf	73,53	93,66	81,95	71,68	76,48
RandomForest	2000	Tf-idf	89,27	97,25	91,67	88,69	90,16
Árvores boosted	2000	Tf-idf	**92,04**	97,84	93,07	91,78	92,42
SVM radial	2000	Tf-idf	91,55	97,79	93,26	91,13	92,18
SVM polinomial	2000	Tf-idf	91,98	97,96	94,14	91,46	92,78
KNN	2000	Tf-idf	87,67	96,95	91,27	86,82	88,99
Nearest S. Centroid	2000	Tf-idf	81,24	88,42	55,79	91,70	69,38
LDA	2000	Tf-idf	89,10	97,00	90,46	88,77	89,61
QDA	2000	Tf-idf	87,40	93,31	70,43	93,00	80,15
Regressão Logística	2000	Tf-idf	88,49	96,90	89,66	88,20	88,93
Elastic net	Todos	Tf-idf	91,75	97,89	93,90	91,23	92,54
Elastic net	Todos	BOW	76,81	95,20	91,00	73,92	81,58
Elastic net	2000	Tf-idf	91,29	97,70	92,93	90,89	91,90

Atribs.: Nº de atributos do modelo
Emb.: Tipo de representação (*embedding*)
Ac.: Acurácia global
MicroP: Precisão micromédia
MicroR: *Recall* micromédio
BOW: *Bag-of-words*
Escores microF2, microP, microF1 e microR calculados para as classes “a” e “b”.

Base as* - com assuntos originais, sem metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	81,35	91,80	66,00	86,36	74,82
Multinomial NB	2000	Tf-idf	74,20	93,71	81,57	72,56	76,80
RandomForest	2000	Tf-idf	90,83	97,52	92,15	90,50	91,32
Árvores boosted	2000	Tf-idf	93,13	98,06	93,25	93,10	93,18
SVM radial	2000	Tf-idf	**93,91**	98,30	94,21	93,83	94,02
SVM polinomial	2000	Tf-idf	93,73	98,32	94,66	93,51	94,08
KNN	2000	Tf-idf	88,29	97,03	90,91	87,66	89,26
Nearest S. Centroid	2000	Tf-idf	84,70	90,38	60,83	93,91	73,84
LDA	2000	Tf-idf	89,57	97,04	89,75	89,53	89,64
QDA	2000	Tf-idf	86,59	92,49	67,31	93,26	78,19
Regressão Logística	2000	Tf-idf	90,50	97,34	90,79	90,42	90,61
Elastic net	Todos	BOW	75,94	95,02	90,72	72,97	80,88
Elastic net	2000	Tf-idf	91,82	97,85	93,22	91,48	92,34
Elastic net	Todos	Tf-idf	92,99	98,22	94,84	92,53	93,67

Base as - com assuntos originais e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	78,51	89,76	59,69	85,23	70,21
Multinomial NB	2000	Tf-idf	74,38	92,74	73,85	74,51	74,18
RandomForest	2000	Tf-idf	90,85	97,57	92,60	90,42	91,50
Árvores boosted	2000	Tf-idf	**93,55**	98,19	94,04	93,43	93,73
SVM radial	2000	Tf-idf	93,40	98,15	93,65	93,34	93,50
SVM polinomial	2000	Tf-idf	91,91	97,79	92,70	91,72	92,21
KNN	2000	Tf-idf	83,46	95,84	87,30	82,55	84,86
Nearest S. Centroid	2000	Tf-idf	63,30	82,27	39,78	74,27	51,81
LDA	2000	Tf-idf	89,17	96,97	89,70	89,04	89,37
QDA	2000	Tf-idf	45,63	90,62	89,15	40,67	55,85
Regressão Logística	2000	Tf-idf	90,18	97,19	90,18	90,18	90,18
Elastic net	Todos	BOW	69,31	93,86	89,10	65,67	75,61
Elastic net	2000	Tf-idf	91,79	97,82	93,06	91,48	92,26
Elastic net	Todos	Tf-idf	92,96	98,16	94,38	92,61	93,49

Base as bi - com assuntos originais, bigramas e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	79,77	90,68	62,45	85,71	72,25
Multinomial NB	2000	Tf-idf	75,63	93,35	76,88	75,32	76,10
RandomForest	2000	Tf-idf	90,67	97,38	91,03	90,58	90,81
Árvores boosted	2000	Tf-idf	**93,26**	98,10	93,56	93,18	93,37
SVM radial	2000	Tf-idf	92,34	97,90	93,19	92,13	92,65
SVM polinomial	2000	Tf-idf	92,18	97,84	92,72	92,05	92,38
KNN	2000	Tf-idf	82,80	95,79	87,71	81,66	84,57
Nearest S. Centroid	2000	Tf-idf	71,18	84,11	45,02	83,28	58,44
LDA	2000	Tf-idf	90,08	96,98	89,04	90,34	89,69
QDA	2000	Tf-idf	57,66	91,90	85,44	53,33	65,67
Regressão Logística	2000	Tf-idf	90,50	97,38	91,15	90,34	90,75
Elastic net	Todos	BOW	75,49	94,94	90,48	72,48	80,49
Elastic net	2000	Tf-idf	92,36	97,88	92,96	92,21	92,58
Elastic net	Todos	Tf-idf	93,02	98,12	94,00	92,78	93,38

Base as_mod* - com assuntos modificados, sem metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	83,13	92,19	66,95	88,47	76,22
Multinomial NB	2000	Tf-idf	76,44	94,11	82,02	75,16	78,44
RandomForest	2000	Tf-idf	91,38	97,65	92,65	91,07	91,85
Árvores boosted	2000	Tf-idf	94,00	98,37	94,68	93,83	94,25
SVM radial	2000	Tf-idf	**94,13**	98,40	94,69	93,99	94,34
SVM polinomial	2000	Tf-idf	92,67	97,95	93,21	92,53	92,87
KNN	2000	Tf-idf	86,96	96,93	92,31	85,71	88,89
Nearest S. Centroid	2000	Tf-idf	84,08	91,06	63,06	91,72	74,74
LDA	2000	Tf-idf	89,91	97,18	90,80	89,69	90,24
QDA	2000	Tf-idf	86,96	92,58	67,60	93,67	78,53
Regressão Logística	2000	Tf-idf	90,49	97,50	92,12	90,10	91,10
Elastic net	Todos	BOW	81,89	96,06	91,95	79,71	85,39
Elastic net	2000	Tf-idf	92,75	98,12	94,28	92,37	93,32
Elastic net	Todos	Tf-idf	93,88	98,42	95,05	93,59	94,31

Base as_mod - com assuntos modificados e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	81,07	90,34	61,20	88,23	72,27
Multinomial NB	2000	Tf-idf	76,46	93,21	75,20	76,79	75,98
RandomForest	2000	Tf-idf	92,24	97,86	93,03	92,05	92,53
Árvores boosted	2000	Tf-idf	93,87	98,33	94,67	93,67	94,17
SVM radial	2000	Tf-idf	93,40	98,15	93,65	93,34	93,50
SVM polinomial	2000	Tf-idf	91,91	97,79	92,70	91,72	92,21
KNN	2000	Tf-idf	83,46	95,84	87,30	82,55	84,86
Nearest S. Centroid	2000	Tf-idf	62,96	82,24	39,68	73,78	51,60
LDA	2000	Tf-idf	89,59	97,08	90,47	89,37	89,91
QDA	2000	Tf-idf	48,13	90,90	88,81	43,18	58,11
Regressão Logística	2000	Tf-idf	90,81	97,43	91,41	90,67	91,04
Elastic net	Todos	BOW	81,97	95,98	91,20	79,95	85,21
Elastic net	Todos	Tf-idf	**94,01**	98,43	95,06	93,75	94,40
Elastic net	2000	Tf-idf	92,67	98,06	93,89	92,37	93,13

Base as_mod bi - com assuntos modificados, bigramas e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	81,98	90,99	63,23	88,55	73,77
Multinomial NB	2000	Tf-idf	77,42	93,73	78,08	77,25	77,66
RandomForest	2000	Tf-idf	92,63	97,81	92,10	92,77	92,43
Árvores boosted	2000	Tf-idf	93,58	98,27	94,57	93,34	93,95
SVM radial	2000	Tf-idf	91,75	97,76	92,90	91,47	92,18
SVM polinomial	2000	Tf-idf	92,73	97,95	93,21	92,61	92,91
KNN	2000	Tf-idf	82,81	95,85	88,22	81,56	84,76
Nearest S. Centroid	2000	Tf-idf	64,18	82,77	40,81	74,90	52,84
LDA	2000	Tf-idf	89,36	97,12	91,02	88,95	89,98
QDA	2000	Tf-idf	62,09	92,46	85,04	58,16	69,08
Regressão Logística	2000	Tf-idf	90,31	97,37	91,20	90,09	90,64
Elastic net	Todos	BOW	80,71	95,98	92,77	78,17	84,85
Elastic net	Todos	Tf-idf	**93,72**	98,32	94,58	93,51	94,04
Elastic net	2000	Tf-idf	92,48	98,02	93,95	92,12	93,03

Word2vec calculado sobre atributos representados por Tf-idf

Base teor - apenas texto dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	word2vec300	25,32	73,79	18,05	28,15	22,00
Multinomial NB	2000	word2vec300	22,43	78,45	21,72	22,62	22,16
RandomForest	2000	word2vec300	62,07	93,11	92,40	57,36	70,78
Árvores boosted	2000	word2vec300	80,98	95,88	91,23	78,76	84,54
SVM radial	2000	word2vec300	**89,36**	97,37	92,83	88,53	90,63
SVM polinomial	2000	word2vec300	88,59	97,14	91,91	87,79	89,80
KNN	2000	word2vec300	84,44	95,94	87,07	83,81	85,41
Nearest S. Centroid	2000	word2vec300	38,06	67,78	19,80	49,47	28,28
LDA	2000	word2vec300	80,64	95,02	84,84	79,66	82,17
QDA	2000	word2vec300	82,46	88,88	56,61	93,08	70,40
Regressão Logística	2000	word2vec300	80,98	95,46	87,93	79,41	83,45
Elastic net	Todos	word2vec300	68,44	93,23	83,61	65,48	73,44

Base teor bi - apenas texto dos documentos, com bigramas

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	word2vec300	26,10	74,06	18,70	28,97	22,73
Multinomial NB	2000	word2vec300	23,35	78,59	22,39	23,60	22,98
RandomForest	2000	word2vec300	61,17	92,95	92,52	56,39	70,07
Árvores boosted	2000	word2vec300	80,73	95,74	90,53	78,60	84,15
SVM radial	2000	word2vec300	**86,22**	96,88	93,27	84,62	88,74
SVM polinomial	2000	word2vec300	79,41	95,40	89,61	77,22	82,95
KNN	2000	word2vec300	81,19	95,70	88,74	79,50	83,86
Nearest S. Centroid	2000	word2vec300	34,36	67,34	18,19	44,18	25,77
LDA	2000	word2vec300	82,73	95,49	86,43	81,86	84,08
QDA	2000	word2vec300	84,00	90,78	61,98	92,19	74,12
Regressão Logística	2000	word2vec300	81,90	95,52	87,02	80,72	83,75
Elastic net	Todos	word2vec300	69,09	93,34	84,18	66,13	74,07

Base as* - com assuntos originais, sem metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	word2vec300	25,77	74,06	18,52	28,57	22,47
Multinomial NB	2000	word2vec300	21,91	78,39	21,27	22,08	21,66
RandomForest	2000	word2vec300	60,49	92,87	92,45	55,68	69,50
Árvores boosted	2000	word2vec300	80,68	95,67	89,97	78,65	83,93
SVM radial	2000	word2vec300	**90,49**	97,49	92,42	89,36	91,20
SVM polinomial	2000	word2vec300	88,68	97,12	91,55	87,99	89,74
KNN	2000	word2vec300	83,96	96,17	89,01	82,79	85,79
Nearest S. Centroid	2000	word2vec300	41,40	67,60	21,18	54,38	30,48
LDA	2000	word2vec300	81,31	95,36	86,87	80,03	83,31
QDA	2000	word2vec300	83,63	89,22	57,43	94,40	71,42
Regressão Logística	2000	word2vec300	84,47	96,15	89,23	83,36	86,19
Elastic net	Todos	word2vec300	63,85	92,32	81,02	60,63	69,36

Base as - com assuntos originais e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	word2vec300	34,97	75,71	24,17	39,37	29,95
Multinomial NB	2000	word2vec300	31,21	79,03	27,24	32,39	29,59
RandomForest	2000	word2vec300	63,43	93,43	93,77	58,69	72,19
Árvores boosted	2000	word2vec300	81,82	95,97	91,10	79,79	85,07
SVM radial	2000	word2vec300	87,28	96,80	91,17	86,36	88,70
SVM polinomial	2000	word2vec300	**87,44**	96,83	90,96	86,61	88,73
KNN	2000	word2vec300	77,27	94,77	86,17	75,32	80,38
Nearest S. Centroid	2000	word2vec300	40,84	66,44	20,51	54,30	29,77
LDA	2000	word2vec300	81,66	95,41	86,93	80,44	83,56
QDA	2000	word2vec300	83,53	90,16	60,13	92,53	72,89
Regressão Logística	2000	word2vec300	85,17	96,17	88,00	84,50	86,21
Elastic net	Todos	word2vec300	69,75	93,39	83,67	66,96	74,39

Base as bi - com assuntos originais, bigramas e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	word2vec300	36,22	76,86	25,85	40,26	31,48
Multinomial NB	2000	word2vec300	31,49	79,39	27,85	32,55	30,01
RandomForest	2000	word2vec300	66,83	93,89	93,21	62,42	74,77
Árvores boosted	2000	word2vec300	79,75	95,64	91,02	77,35	83,63
SVM radial	2000	word2vec300	88,90	97,24	92,73	87,99	90,30
SVM polinomial	2000	word2vec300	**87,75**	96,82	90,47	87,09	88,75
KNN	2000	word2vec300	77,97	94,98	87,15	75,97	81,18
Nearest S. Centroid	2000	word2vec300	43,02	67,38	21,82	56,82	31,53
LDA	2000	word2vec300	83,59	95,77	86,96	82,79	84,82
QDA	2000	word2vec300	84,59	90,53	61,10	93,59	73,93
Regressão Logística	2000	word2vec300	86,31	96,49	89,47	85,55	87,47
Elastic net	Todos	word2vec300	70,27	93,45	83,38	67,61	74,68

Base as_mod* - com assuntos modificados, sem metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	word2vec300	25,85	74,43	18,87	28,49	22,70
Multinomial NB	2000	word2vec300	22,72	78,35	21,75	22,97	22,35
RandomForest	2000	word2vec300	61,80	93,09	92,50	57,06	70,58
Árvores boosted	2000	word2vec300	82,99	96,15	91,16	81,17	85,87
SVM radial	2000	word2vec300	**90,74**	97,53	92,37	90,34	91,34
SVM polinomial	2000	word2vec300	88,59	97,12	91,77	87,82	89,76
KNN	2000	word2vec300	83,83	96,24	90,14	82,39	86,09
Nearest S. Centroid	2000	word2vec300	45,17	67,25	22,59	60,23	32,85
LDA	2000	word2vec300	81,12	95,40	87,29	79,71	83,33
QDA	2000	word2vec300	84,69	90,56	61,22	93,67	74,05
Regressão Logística	2000	word2vec300	83,86	96,09	89,52	82,55	85,90
Elastic net	Todos	word2vec300	69,74	93,36	83,10	67,05	74,21

Base as_mod - com assuntos modificados, com metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	word2vec300	34,75	76,33	24,59	38,75	30,09
Multinomial NB	2000	word2vec300	31,14	79,19	27,38	32,25	29,62
RandomForest	2000	word2vec300	59,88	92,84	92,87	55,00	69,08
Árvores boosted	2000	word2vec300	81,81	95,82	89,47	80,10	84,53
SVM radial	2000	word2vec300	**89,39**	97,33	92,23	88,71	90,43
SVM polinomial	2000	word2vec300	87,23	96,83	90,93	86,35	88,58
KNN	2000	word2vec300	75,43	94,26	83,36	73,68	78,22
Nearest S. Centroid	2000	word2vec300	40,18	66,13	20,15	53,45	29,27
LDA	2000	word2vec300	81,15	95,36	86,39	79,94	83,04
QDA	2000	word2vec300	83,87	90,59	61,33	92,36	73,71
Regressão Logística	2000	word2vec300	83,40	95,98	88,54	82,21	85,26
Elastic net	Todos	word2vec300	72,48	93,71	82,79	70,29	76,03

Base as_mod bi - com assuntos modificados, bigramas e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	word2vec300	35,24	75,98	24,45	39,61	30,24
Multinomial NB	2000	word2vec300	31,69	78,89	27,26	33,04	29,87
RandomForest	2000	word2vec300	62,04	93,19	93,50	57,22	71,00
Árvores boosted	2000	word2vec300	82,97	96,30	91,90	81,01	86,11
SVM radial	2000	word2vec300	**89,03**	97,26	92,74	88,15	90,39
SVM polinomial	2000	word2vec300	88,11	97,06	92,03	87,18	89,54
KNN	2000	word2vec300	77,02	94,88	86,75	74,92	80,40
Nearest S. Centroid	2000	word2vec300	43,17	67,37	21,83	57,14	31,59
LDA	2000	word2vec300	82,80	95,61	87,34	81,74	84,44
QDA	2000	word2vec300	84,50	90,82	61,99	92,94	74,37
Regressão Logística	2000	word2vec300	85,46	96,43	90,27	84,33	87,20
Elastic net	Todos	word2vec300	73,04	93,85	83,30	70,86	76,58

b. Seleção de atributos por regressão Lasso

Base teor - apenas texto dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	551	Tf-idf	82,68	91,57	64,69	88,85	74,87
Multinomial NB	551	Tf-idf	67,92	93,43	86,65	64,44	73,92
RandomForest	551	Tf-idf	89,07	97,20	92,03	88,36	90,16
Árvores boosted	551	Tf-idf	91,76	97,77	92,97	91,46	92,21
SVM radial	551	Tf-idf	**92,87**	97,96	92,99	92,84	92,92
SVM polinomial	551	Tf-idf	92,63	97,96	93,42	92,43	92,92
KNN	551	Tf-idf	87,90	97,00	91,76	86,98	89,31
Nearest S. Centroid	551	Tf-idf	84,03	90,53	61,20	92,68	73,72
LDA	551	Tf-idf	85,41	96,07	87,65	84,87	86,23
QDA	551	Tf-idf	83,43	88,79	56,50	94,71	70,78
Regressão Logística	551	Tf-idf	88,52	96,89	89,81	88,20	89,00
Elastic net	551	BOW	70,18	94,00	89,12	66,64	76,26
Elastic net	551	Tf-idf	89,37	97,22	91,84	88,77	90,28

Base teor bi - apenas texto dos documentos, com bigramas

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	538	Tf-idf	84,52	92,39	67,60	90,15	77,27
Multinomial NB	538	Tf-idf	69,04	93,63	87,51	65,58	74,98
RandomForest	538	Tf-idf	89,76	97,29	91,80	89,26	90,51
Árvores boosted	538	Tf-idf	91,87	97,78	92,90	91,62	92,26
SVM radial	538	Tf-idf	92,81	98,01	93,36	92,68	93,02
SVM polinomial	538	Tf-idf	**93,21**	98,08	93,39	93,17	93,28
KNN	538	Tf-idf	87,90	96,90	91,07	87,14	89,06
Nearest S. Centroid	538	Tf-idf	82,75	90,22	60,65	91,05	72,80
LDA	538	Tf-idf	86,37	96,28	88,22	85,92	87,06
QDA	538	Tf-idf	84,33	89,24	57,65	95,36	71,86
Regressão Logística	538	Tf-idf	89,10	97,10	90,76	88,69	89,71
Elastic net	538	BOW	72,55	94,36	89,12	69,32	77,99
Elastic net	538	Tf-idf	89,86	97,40	92,34	89,26	90,77

Base as - com assuntos originais e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	533	Tf-idf	77,04	87,43	53,68	86,44	66,23
Multinomial NB	533	Tf-idf	71,82	92,91	76,29	70,78	73,43
RandomForest	533	Tf-idf	90,94	97,57	92,38	90,58	91,48
Árvores boosted	533	Tf-idf	**93,03**	98,04	93,10	93,02	93,06
SVM radial	533	Tf-idf	92,04	97,82	93,01	91,80	92,40
SVM polinomial	533	Tf-idf	92,19	97,86	93,10	91,96	92,53
KNN	533	Tf-idf	83,43	95,78	86,79	82,63	84,66
Nearest S. Centroid	533	Tf-idf	68,96	81,38	40,62	83,52	54,66
LDA	533	Tf-idf	86,08	96,22	87,92	85,63	86,76
QDA	533	Tf-idf	85,34	90,51	60,85	94,89	74,15
Regressão Logística	533	Tf-idf	88,46	96,91	90,41	87,99	89,18
Elastic net	533	BOW	73,18	94,54	90,16	69,89	78,74
Elastic net	533	Tf-idf	90,04	97,38	92,15	89,53	90,82

Base as bi - com assuntos originais, bigramas e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	343	Tf-idf	79,73	87,70	54,64	90,07	68,02
Multinomial NB	343	Tf-idf	75,29	93,80	80,34	74,13	77,11
RandomForest	343	Tf-idf	91,68	97,46	90,62	91,94	91,28
Árvores boosted	343	Tf-idf	92,03	97,75	92,40	91,94	92,17
SVM radial	343	Tf-idf	**92,62**	97,92	93,04	92,51	92,78
SVM polinomial	343	Tf-idf	92,09	97,91	93,68	91,70	92,68
KNN	343	Tf-idf	84,24	96,00	88,12	83,32	85,65
Nearest S. Centroid	343	Tf-idf	68,43	82,40	41,82	81,37	55,25
LDA	343	Tf-idf	87,24	96,52	88,99	86,82	87,89
QDA	343	Tf-idf	85,97	91,57	63,97	94,06	76,15
Regressão Logística	343	Tf-idf	90,52	97,47	92,03	90,15	91,08
Elastic net	343	BOW	72,21	94,38	89,81	68,84	77,94
Elastic net	343	Tf-idf	91,38	97,69	92,71	91,05	91,87

Base as_mod* - com assuntos modificados, sem metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	166	Tf-idf	85,64	91,88	65,38	92,83	76,73
Multinomial NB	166	Tf-idf	66,52	93,43	89,50	62,51	73,61
RandomForest	166	Tf-idf	90,78	97,49	92,04	90,46	91,25
Árvores boosted	166	Tf-idf	92,48	97,95	93,40	92,26	92,82
SVM radial	166	Tf-idf	**93,13**	97,97	92,41	93,32	92,86
SVM polinomial	166	Tf-idf	92,55	97,86	92,43	92,58	92,51
KNN	166	Tf-idf	88,90	97,13	91,17	88,35	89,74
Nearest S. Centroid	166	Tf-idf	85,06	91,92	65,64	91,85	76,56
LDA	166	Tf-idf	85,09	95,76	85,76	84,92	85,34
QDA	166	Tf-idf	84,84	91,02	62,38	93,24	74,75
Regressão Logística	166	Tf-idf	88,85	97,01	90,57	88,43	89,48
Elastic net	166	BOW	74,73	94,78	90,34	71,64	79,91
Elastic net	166	Tf-idf	89,23	97,10	90,83	88,83	89,82

Base as_mod - com assuntos modificados e metadados dos documentos

Modelo	Emb.	Atribs.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	166	Tf-idf	78,66	86,50	52,00	90,22	65,97
Multinomial NB	166	Tf-idf	71,91	93,25	79,80	70,17	74,67
RandomForest	166	Tf-idf	90,89	97,54	92,28	90,55	91,40
Árvores boosted	166	Tf-idf	**92,88**	97,99	93,43	92,75	93,09
SVM radial	166	Tf-idf	89,90	97,20	90,91	89,65	90,27
SVM polinomial	166	Tf-idf	91,74	97,67	92,28	91,61	91,94
KNN	166	Tf-idf	86,41	96,44	88,86	85,82	87,31
Nearest S. Centroid	166	Tf-idf	71,33	80,00	40,16	88,51	55,25
LDA	166	Tf-idf	85,63	95,85	85,85	85,57	85,71
QDA	166	Tf-idf	84,38	89,37	57,90	95,27	72,03
Regressão Logística	166	Tf-idf	88,94	97,04	90,73	88,51	89,60
Elastic net	166	BOW	80,74	95,75	90,77	78,57	84,23
Elastic net	166	Tf-idf	89,43	97,12	90,86	89,08	89,96

Base as_mod bi - com assuntos modificados, bigramas e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	438	Tf-idf	79,73	87,70	54,64	90,07	68,02
Multinomial NB	438	Tf-idf	75,29	93,80	80,34	74,13	77,11
RandomForest	438	Tf-idf	91,68	97,46	90,62	91,94	91,28
Árvores boosted	438	Tf-idf	92,03	97,75	92,40	91,94	92,17
SVM radial	438	Tf-idf	**92,62**	97,92	93,04	92,51	92,78
SVM polinomial	438	Tf-idf	92,09	97,91	93,68	91,70	92,68
KNN	438	Tf-idf	84,24	96,00	88,12	83,32	85,65
Nearest S. Centroid	438	Tf-idf	68,43	82,40	41,82	81,37	55,25
LDA	438	Tf-idf	87,24	96,52	88,99	86,82	87,89
QDA	438	Tf-idf	85,97	91,57	63,97	94,06	76,15
Regressão Logística	438	Tf-idf	90,52	97,47	92,03	90,15	91,08
Elastic net	438	BOW	72,21	94,38	89,81	68,84	77,94
Elastic net	438	Tf-idf	91,38	97,69	92,71	91,05	91,87

c. Sem seleção de atributos, com word2vec calculado sobre atributos representados por Tf-idf

Base as_mod* - com assuntos modificados, sem metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	todos	word2vec100	18,01	71,39	12,93	19,97	15,70
Multinomial NB	todos	word2vec100	7,13	81,96	16,28	6,25	9,03
RandomForest	todos	word2vec100	30,18	88,87	93,26	25,81	40,43
Árvores boosted	todos	word2vec100	50,51	90,57	78,87	46,35	58,38
SVM radial	todos	word2vec100	61,32	92,49	85,87	57,22	68,68
SVM polinomial	todos	word2vec100	40,53	89,84	85,96	35,80	50,54
KNN	todos	word2vec100	61,23	89,03	60,72	61,36	61,04
Nearest S. Centroid	todos	word2vec100	35,71	66,29	18,89	45,94	26,77
LDA	todos	word2vec100	47,75	89,41	70,83	44,16	54,40
QDA	todos	word2vec100	**66,51**	88,45	57,00	69,40	62,59
Regressão Logística	todos	word2vec100	44,82	89,32	72,52	40,91	52,31
Elastic net	todos	word2vec100	43,33	89,31	73,67	39,29	51,24

Base as_mod - com assuntos modificados e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	todos	word2vec100	35,18	69,31	19,86	43,59	27,29
Multinomial NB	todos	word2vec100	23,48	83,51	34,40	21,75	26,65
RandomForest	todos	word2vec100	40,90	90,16	90,97	35,96	51,54
Árvores boosted	todos	word2vec100	57,47	91,79	81,86	53,49	64,70
SVM radial	todos	word2vec100	**65,31**	92,45	79,08	62,58	69,87
SVM polinomial	todos	word2vec100	47,89	90,64	83,67	43,26	57,04
KNN	todos	word2vec100	58,14	88,24	56,64	58,52	57,56
Nearest S. Centroid	todos	word2vec100	34,40	46,89	13,76	55,03	22,02
LDA	todos	word2vec100	53,38	90,85	78,48	49,43	60,66
QDA	todos	word2vec100	44,87	48,24	17,50	73,70	28,28
Regressão Logística	todos	word2vec100	55,79	91,01	77,01	52,19	62,22
Elastic net	todos	word2vec100	56,32	91,12	78,55	52,60	63,00

Base as_mod* - com assuntos modificados, sem metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	todos	word2vec200	18,71	71,54	13,38	20,78	16,28
Multinomial NB	todos	word2vec200	9,35	80,00	14,34	8,60	10,76
RandomForest	todos	word2vec200	34,88	89,40	91,85	30,19	45,45
Árvores boosted	todos	word2vec200	63,99	92,82	84,62	60,31	70,43
SVM radial	todos	word2vec200	**76,03**	94,72	87,46	73,62	79,95
SVM polinomial	todos	word2vec200	62,75	92,74	86,79	58,69	70,02
KNN	todos	word2vec200	70,84	92,02	71,39	70,70	71,04
Nearest S. Centroid	todos	word2vec200	25,30	70,90	16,56	29,14	21,12
LDA	todos	word2vec200	65,47	92,16	78,20	62,91	69,73
QDA	todos	word2vec200	74,63	86,34	51,23	84,25	63,72
Regressão Logística	todos	word2vec200	62,44	92,18	81,32	59,01	68,39
Elastic net	todos	word2vec200	65,02	92,40	80,10	62,09	69,96

Base as_mod - com assuntos modificados e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	todos	word2vec200	34,12	70,16	19,88	41,56	26,89
Multinomial NB	todos	word2vec200	24,43	81,75	29,31	23,46	26,06
RandomForest	todos	word2vec200	40,69	90,30	94,21	35,63	51,71
Árvores boosted	todos	word2vec200	64,63	92,95	85,15	60,96	71,05
SVM radial	todos	word2vec200	**76,27**	94,81	87,26	73,94	80,05
SVM polinomial	todos	word2vec200	51,09	91,23	87,69	46,27	60,57
KNN	todos	word2vec200	64,13	89,65	61,67	64,77	63,18
Nearest S. Centroid	todos	word2vec200	39,18	57,11	17,35	57,14	26,62
LDA	todos	word2vec200	67,74	92,49	78,43	65,50	71,38
QDA	todos	word2vec200	57,02	55,74	23,14	89,94	36,81
Regressão Logística	todos	word2vec200	65,52	92,42	79,61	62,74	70,18
Elastic net	todos	word2vec200	65,02	92,40	80,10	62,09	69,96

Base as_mod* - com assuntos modificados, sem metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	todos	word2vec300	18,59	70,91	13,07	20,78	16,05
Multinomial NB	todos	word2vec300	9,92	78,89	13,22	9,33	10,94
RandomForest	todos	word2vec300	39,75	90,14	93,86	34,74	50,71
Árvores boosted	todos	word2vec300	70,65	94,05	88,84	67,21	76,52
SVM radial	todos	word2vec300	**78,62**	95,41	90,88	76,06	82,81
SVM polinomial	todos	word2vec300	72,95	94,19	87,44	70,05	77,78
KNN	todos	word2vec300	75,45	93,11	74,68	75,65	75,16
Nearest S. Centroid	todos	word2vec300	38,44	58,52	17,49	54,87	26,52
LDA	todos	word2vec300	71,57	93,19	80,93	69,56	74,81
QDA	todos	word2vec300	76,64	84,81	48,44	89,69	62,91
Regressão Logística	todos	word2vec300	69,70	93,31	82,85	67,05	74,11
Elastic net	todos	word2vec300	69,74	93,36	83,10	67,05	74,21

Base as_mod - com assuntos modificados e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	todos	word2vec300	34,96	72,28	21,56	41,40	28,35
Multinomial NB	todos	word2vec300	25,01	80,69	27,21	24,51	25,79
RandomForest	todos	word2vec300	44,32	90,71	92,71	39,20	55,11
Árvores boosted	todos	word2vec300	72,81	94,31	88,47	69,72	77,98
SVM radial	todos	word2vec300	**78,45**	95,27	89,33	76,14	82,21
SVM polinomial	todos	word2vec300	74,46	94,44	87,70	71,75	78,93
KNN	todos	word2vec300	67,57	90,84	66,75	67,78	67,26
Nearest S. Centroid	todos	word2vec300	41,65	63,13	20,02	57,06	29,64
LDA	todos	word2vec300	71,96	93,61	83,92	69,48	76,02
QDA	todos	word2vec300	71,30	78,41	39,11	89,77	54,48
Regressão Logística	todos	word2vec300	73,09	93,85	83,16	70,94	76,57
Elastic net	todos	word2vec300	73,04	93,85	83,30	70,86	76,58

d. Sem seleção de atributos, com word2vec calculado sobre atributos representados por BOW

Base as_mod* - com assuntos modificados, sem metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	todos	word2vec300	12,36	79,12	15,49	11,77	13,38
Multinomial NB	todos	word2vec300	4,80	83,35	17,73	4,06	6,61
RandomForest	todos	word2vec300	67,82	93,78	90,66	63,80	74,89
Árvores boosted	todos	word2vec300	**80,43**	95,61	89,69	78,41	83,67
SVM radial	todos	word2vec300	59,76	92,21	86,04	55,52	67,49
SVM polinomial	todos	word2vec300	16,46	86,95	81,64	13,72	23,49
KNN	todos	word2vec300	79,94	95,07	85,53	78,65	81,95
Nearest S. Centroid	todos	word2vec300	39,51	69,93	21,49	50,00	30,06
LDA	todos	word2vec300	38,96	88,90	75,75	34,74	47,63
QDA	todos	word2vec300	29,69	14,27	9,63	61,93	16,67
Regressão Logística	todos	word2vec300	73,76	94,05	84,86	71,43	77,57
Elastic net	todos	word2vec300	62,17	92,52	85,56	58,20	69,28

Base as_mod - com assuntos modificados e metadados dos documentos

Modelo	Atribs.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	todos	word2vec300	32,46	70,42	19,39	39,04	25,91
Multinomial NB	todos	word2vec300	21,52	84,54	41,00	19,24	26,19
RandomForest	todos	word2vec300	67,47	93,74	90,81	63,39	74,67
Árvores boosted	todos	word2vec300	**81,46**	95,84	90,56	79,46	84,65
SVM radial	todos	word2vec300	56,65	91,84	85,19	52,27	64,79
SVM polinomial	todos	word2vec300	24,53	87,79	82,64	20,86	33,31
KNN	todos	word2vec300	72,79	93,62	80,44	71,10	75,48
Nearest S. Centroid	todos	word2vec300	32,66	38,48	12,23	56,09	20,08
LDA	todos	word2vec300	46,21	89,75	76,51	42,05	54,27
QDA	todos	word2vec300	40,88	60,59	18,97	57,47	28,53
Regressão Logística	todos	word2vec300	75,88	94,38	85,21	73,86	79,13
Elastic net	todos	word2vec300	62,95	92,64	85,93	59,01	69,97

Apêndice 6 – Resultados do segundo experimento

Base as_mod - com assuntos modificados e metadados

NearMiss 1

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	48,41	38,60	17,50	86,69	29,12
Multinomial NB	2000	Tf-idf	47,21	33,38	16,52	88,15	27,82
RandomForest	2000	Tf-idf	49,54	27,40	16,60	98,30	28,40
Árvores boosted	2000	Tf-idf	50,80	30,23	17,24	98,94	29,36
SVM radial	2000	Tf-idf	55,04	42,50	20,04	97,73	33,26
SVM polinomial	2000	Tf-idf	57,04	47,31	21,45	97,48	35,16
KNN	2000	Tf-idf	55,17	50,11	21,44	90,91	34,70
Nearest S. Centroid	2000	Tf-idf	45,45	25,93	15,27	89,85	26,10
LDA	2000	Tf-idf	60,41	55,97	24,41	95,70	38,90
QDA	2000	Tf-idf	46,70	21,11	15,25	96,43	26,33
Regressão Logística	2000	Tf-idf	62,80	58,46	25,80	97,89	40,84
Elastic net	Todos	BOW	60,86	56,36	24,57	96,51	39,16
Elastic net	Todos	Tf-idf	**79,27**	82,25	45,02	97,89	61,67

NearMiss 2

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	54,95	51,64	21,69	89,12	34,88
Multinomial NB	2000	Tf-idf	53,36	48,96	20,66	88,31	33,48
RandomForest	2000	Tf-idf	77,90	80,67	42,94	97,81	59,68
Árvores boosted	2000	Tf-idf	81,75	84,72	48,85	98,30	65,27
SVM radial	2000	Tf-idf	89,28	92,32	65,88	97,97	78,79
SVM polinomial	2000	Tf-idf	89,16	92,21	65,56	97,97	78,56
KNN	2000	Tf-idf	68,05	72,23	33,39	91,88	48,98
Nearest S. Centroid	2000	Tf-idf	81,72	89,88	60,00	89,85	71,95
LDA	2000	Tf-idf	90,23	94,49	74,01	95,45	83,37
QDA	2000	Tf-idf	54,73	42,36	19,90	97,32	33,04
Regressão Logística	2000	Tf-idf	86,66	90,00	59,41	97,89	73,94
Elastic net	Todos	BOW	47,29	22,02	15,48	97,24	26,71
Elastic net	Todos	Tf-idf	**92,59**	95,43	76,90	97,56	86,01

NearMiss 3

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	68,03	73,29	34,03	90,67	49,49
Multinomial NB	2000	Tf-idf	71,69	82,54	44,46	84,66	58,30
RandomForest	2000	Tf-idf	89,56	93,01	68,21	97,16	80,15
Árvores boosted	2000	Tf-idf	92,02	94,84	74,60	97,73	84,61
SVM radial	2000	Tf-idf	92,27	95,94	80,03	95,94	87,26
SVM polinomial	2000	Tf-idf	92,30	95,68	78,59	96,51	86,63
KNN	2000	Tf-idf	74,88	83,72	46,65	88,23	61,03
Nearest S. Centroid	2000	Tf-idf	82,00	93,58	74,84	84,01	79,16
LDA	2000	Tf-idf	**92,63**	96,37	82,10	95,70	88,38
QDA	2000	Tf-idf	71,09	73,07	34,74	96,27	51,05
Regressão Logística	2000	Tf-idf	87,41	92,07	65,53	95,37	77,69
Elastic net	Todos	BOW	89,12	94,74	75,88	93,18	83,64
Elastic net	Todos	Tf-idf	92,27	95,47	77,43	96,92	86,09

One-Sided Selection

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	76,04	85,14	48,96	88,23	62,98
Multinomial NB	2000	Tf-idf	76,70	89,10	58,12	83,36	68,49
RandomForest	2000	Tf-idf	92,95	96,01	79,87	96,92	87,57
Árvores boosted	2000	Tf-idf	93,70	96,77	83,68	96,59	89,68
SVM radial	2000	Tf-idf	**94,33**	98,27	93,42	94,56	93,99
SVM polinomial	2000	Tf-idf	93,68	96,93	84,59	96,27	90,05
KNN	2000	Tf-idf	79,46	89,34	58,69	87,18	70,15
Nearest S. Centroid	2000	Tf-idf	74,13	84,42	46,65	86,93	60,71
LDA	2000	Tf-idf	92,52	96,95	85,95	94,32	89,94
QDA	2000	Tf-idf	83,53	88,08	55,16	95,86	70,03
Regressão Logística	2000	Tf-idf	91,90	96,27	82,13	94,72	87,98
Elastic net	Todos	BOW	83,25	96,22	91,51	81,41	86,17
Elastic net	Todos	Tf-idf	93,37	97,01	85,36	95,62	90,20

Tomek Links

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	80,84	90,39	61,34	87,82	72,23
Multinomial NB	2000	Tf-idf	76,61	93,15	74,69	77,11	75,88
RandomForest	2000	Tf-idf	92,48	97,84	92,60	92,45	92,53
Árvores boosted	2000	Tf-idf	94,05	98,32	94,30	93,99	94,15
SVM radial	2000	Tf-idf	93,94	98,29	94,07	93,91	93,99
SVM polinomial	2000	Tf-idf	92,42	97,88	92,97	92,29	92,63
KNN	2000	Tf-idf	83,78	95,94	87,64	82,87	85,19
Nearest S. Centroid	2000	Tf-idf	63,03	82,20	39,63	73,94	51,60
LDA	2000	Tf-idf	90,44	97,27	90,86	90,34	90,60
QDA	2000	Tf-idf	87,67	93,12	69,43	93,83	79,81
Regressão Logística	2000	Tf-idf	91,75	97,56	91,22	91,88	91,55
Elastic net	Todos	BOW	82,14	96,02	91,39	80,11	85,38
Elastic net	Todos	Tf-idf	**94,07**	98,32	94,07	94,07	94,07

Condensed Nearest Neighbors

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	64,31	66,00	28,88	92,78	44,05
Multinomial NB	2000	Tf-idf	73,29	82,90	44,95	87,01	59,28
RandomForest	2000	Tf-idf	90,61	93,80	70,84	97,40	82,02
Árvores boosted	2000	Tf-idf	91,63	95,05	75,81	96,67	84,98
SVM radial	2000	Tf-idf	92,29	95,55	77,91	96,75	86,31
SVM polinomial	2000	Tf-idf	92,46	95,69	78,52	96,75	86,69
KNN	2000	Tf-idf	70,59	75,10	35,98	92,94	51,88
Nearest S. Centroid	2000	Tf-idf	82,43	93,84	75,88	84,25	79,85
LDA	2000	Tf-idf	92,54	96,13	80,81	96,02	87,76
QDA	2000	Tf-idf	63,24	60,46	26,56	96,59	41,66
Regressão Logística	2000	Tf-idf	89,28	93,60	70,58	95,62	81,21
Elastic net	Todos	BOW	89,41	95,20	78,07	92,78	84,79
Elastic net	Todos	Tf-idf	**94,34**	97,03	84,53	97,16	90,41

Base as_mod* - com assuntos modificados, sem metadados

NearMiss 1

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	50,94	41,53	18,68	89,61	30,92
Multinomial NB	2000	Tf-idf	49,30	36,22	17,47	90,50	29,29
RandomForest	2000	Tf-idf	49,44	27,31	16,56	98,13	28,34
Árvores boosted	2000	Tf-idf	50,06	28,74	16,88	98,46	28,82
SVM radial	2000	Tf-idf	49,68	27,87	16,69	98,21	28,53
SVM polinomial	2000	Tf-idf	49,85	28,39	16,79	98,21	28,67
KNN	2000	Tf-idf	51,96	39,70	18,71	93,51	31,18
Nearest S. Centroid	2000	Tf-idf	45,64	32,05	15,95	85,39	26,88
LDA	2000	Tf-idf	58,67	52,75	23,09	95,45	37,18
QDA	2000	Tf-idf	46,38	19,86	15,06	96,59	26,06
Regressão Logística	2000	Tf-idf	61,68	56,40	24,87	97,89	39,66
Elastic net	Todos	BOW	60,86	56,36	24,57	96,51	39,16
Elastic net	Todos	Tf-idf	**77,71**	80,44	42,58	97,89	59,35

NearMiss 2

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	59,25	58,50	24,78	90,83	38,94
Multinomial NB	2000	Tf-idf	57,52	56,09	23,58	89,85	37,36
RandomForest	2000	Tf-idf	77,03	79,79	41,82	97,56	58,55
Árvores boosted	2000	Tf-idf	82,46	85,63	50,40	98,05	66,57
SVM radial	2000	Tf-idf	89,35	92,19	65,37	98,38	78,55
SVM polinomial	2000	Tf-idf	88,38	91,14	62,33	98,70	76,41
KNN	2000	Tf-idf	70,00	72,43	34,00	95,21	50,11
Nearest S. Centroid	2000	Tf-idf	81,91	93,30	73,48	84,33	78,53
LDA	2000	Tf-idf	89,62	94,41	74,11	94,56	83,10
QDA	2000	Tf-idf	53,66	39,68	19,19	97,40	32,06
Regressão Logística	2000	Tf-idf	87,88	91,12	62,25	97,97	76,13
Elastic net	Todos	BOW	47,29	22,02	15,48	97,24	26,71
Elastic net	Todos	Tf-idf	**92,57**	95,37	76,62	97,65	85,87

NearMiss 3

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	72,14	77,75	38,62	92,13	54,42
Multinomial NB	2000	Tf-idf	75,43	85,67	50,28	86,20	63,52
RandomForest	2000	Tf-idf	90,23	93,47	69,69	97,40	81,25
Árvores boosted	2000	Tf-idf	91,06	94,19	72,25	97,40	82,96
SVM radial	2000	Tf-idf	92,39	96,03	80,46	95,94	87,52
SVM polinomial	2000	Tf-idf	92,29	96,00	80,34	95,86	87,42
KNN	2000	Tf-idf	80,94	87,73	54,60	92,05	68,54
Nearest S. Centroid	2000	Tf-idf	82,04	93,58	74,75	84,09	79,14
LDA	2000	Tf-idf	91,58	96,32	82,79	94,07	88,07
QDA	2000	Tf-idf	74,34	78,12	39,61	95,21	55,95
Regressão Logística	2000	Tf-idf	86,49	91,14	62,74	95,54	75,74
Elastic net	Todos	BOW	89,12	94,74	75,88	93,18	83,64
Elastic net	Todos	Tf-idf	**92,81**	95,66	77,89	97,48	86,59

One-Sided Selection

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	79,04	87,19	53,14	90,02	66,83
Multinomial NB	2000	Tf-idf	78,10	90,76	63,85	82,71	72,07
RandomForest	2000	Tf-idf	93,70	96,43	81,56	97,32	88,75
Árvores boosted	2000	Tf-idf	**94,49**	98,21	92,93	94,89	93,90
SVM radial	2000	Tf-idf	93,78	97,01	85,02	96,27	90,29
SVM polinomial	2000	Tf-idf	93,51	96,95	84,91	95,94	90,09
KNN	2000	Tf-idf	85,63	92,21	66,63	92,21	77,36
Nearest S. Centroid	2000	Tf-idf	85,36	91,33	63,61	93,34	75,66
LDA	2000	Tf-idf	92,35	96,83	85,25	94,32	89,56
QDA	2000	Tf-idf	68,82	81,74	42,33	81,57	55,74
Regressão Logística	2000	Tf-idf	91,67	96,00	80,73	94,89	87,24
Elastic net	Todos	BOW	83,00	96,19	91,65	81,09	86,05
Elastic net	Todos	Tf-idf	94,35	98,38	94,17	94,40	94,28

Tomek Links

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	83,18	92,13	66,75	88,64	76,15
Multinomial NB	2000	Tf-idf	76,75	94,13	81,88	75,57	78,60
RandomForest	2000	Tf-idf	92,31	97,86	93,03	92,13	92,58
Árvores boosted	2000	Tf-idf	94,25	98,37	94,31	94,24	94,28
SVM radial	2000	Tf-idf	94,40	98,39	94,25	94,40	94,32
SVM polinomial	2000	Tf-idf	94,04	98,31	94,22	93,99	94,11
KNN	2000	Tf-idf	89,56	97,30	91,35	89,12	90,22
Nearest S. Centroid	2000	Tf-idf	83,97	91,00	62,90	91,64	74,60
LDA	2000	Tf-idf	89,81	97,09	90,28	89,69	89,98
QDA	2000	Tf-idf	48,63	90,98	89,07	43,67	58,61
Regressão Logística	2000	Tf-idf	90,58	97,40	91,24	90,42	90,83
Elastic net	Todos	BOW	82,08	96,02	91,47	80,03	85,37
Elastic net	Todos	Tf-idf	**94,49**	98,37	93,88	94,64	94,26

Condensed Nearest Neighbors

Modelo	Atrib.	Emb.	MicroF2	Ac.	MicroP	MicroR	MicroF1
Complement NB	2000	Tf-idf	68,97	71,19	32,84	95,13	48,82
Multinomial NB	2000	Tf-idf	77,45	86,25	51,36	88,72	65,06
RandomForest	2000	Tf-idf	90,90	93,96	71,38	97,56	82,44
Árvores boosted	2000	Tf-idf	91,75	95,11	76,02	96,75	85,14
SVM radial	2000	Tf-idf	92,60	95,82	79,05	96,75	87,01
SVM polinomial	2000	Tf-idf	92,46	95,69	78,52	96,75	86,69
KNN	2000	Tf-idf	77,29	81,52	43,68	95,70	59,98
Nearest S. Centroid	2000	Tf-idf	76,96	90,21	61,65	82,06	70,40
LDA	2000	Tf-idf	92,25	96,26	81,81	95,29	88,04
QDA	2000	Tf-idf	30,88	15,47	10,21	62,58	17,55
Regressão Logística	2000	Tf-idf	89,15	93,55	70,35	95,54	81,03
Elastic net	Todos	BOW	89,32	95,21	78,20	92,61	84,80
Elastic net	Todos	Tf-idf	**94,34**	97,03	84,53	97,16	90,41

Apêndice 7 – Faixas de valores testados de hiperparâmetros das redes BERT

Hiperparâmetro	Nome original	Faixa de valores testados
Nº de épocas de treinamento	num train epochs	5, 10, 15, 20, 25, 30, 40
Taxa de aprendizado	learning rate	$5x10^{-5}$ a $5x10^{-3}$
Tamanho do lote de treinamento em nº de contextos	train batch size	8, 12, 16
Tamanho do lote de validação em nº de contextos	evaluation batch size	8, 12, 16
Constante ε do algoritmo de otimização AdamW	adam epsilon	10^{-11} a 10^{-5}
Nº máximo de tokens de um lote de entrada (contexto + pergunta) na rede	maximum sequence length	256, 384, 512
Nº de tokens de contexto sobrepostos entre lotes de entrada, para processar contextos longos	document stride	192, 256, 320, 384, 448, 490
Nº de lotes (*steps*) de tolerância, sem redução da função perda	early stopping patience	2, 3, 4, 5, 6
Proporção de lotes iniciais para aquecer o treinamento	warmup ratio	0,04; 0,05; 0,06; 0,1; 0,2; 0,25
Taxa de aprendizado final de decaimento	polynomial decay schedule lr end	$5x10^{-7}$, $1x10^{-7}$, $5x10^{-8}$, $1x10^{-8}$, $5x10^{-9}$
Expoente da taxa de decaimento da taxa de aprendizado	polynomial decay schedule power	0,5; 0,75; 0,95; 1; 1,1; 1,25; 1,5; 2
Intervalo de cálculo de métricas no conjunto de validação, em nº de lotes durante o treino	evaluate during training steps	120, 180, 250
Nº de lotes de acumulação de gradiente	gradient accumulation steps	1, 2
Nível de corte do gradiente	max gradient norm	0,9; 0,95; 0,99; 1; 1,01; 1,05; 1,1
Coeficiente de regularização L2 da função perda	weight decay	0; 0,1; 0,01; 0,001

Apêndice 8 – Hiperparâmetros das melhores configurações de rede BERT para o problema de extração de informações

Aumento de capital

Config.	Adam epsilon	Doc stride	Early stopping patience	Evaluate during training steps	Eval batch size	Gradient accum. steps	Learning rate	Max grad norm	Max seq length	Num train epochs	Polynom. decay schedule lr end	Polynom. decay sched. power	Train batch size	Warmup ratio	Weight decay
1	9,03E-06	448	2	250	8	1	1,34E-04	0,9	512	10	1E-08	2	12	0,04	0,001
2	1,01E-07	448	2	180	16	1	1,48E-04	1,1	512	15	5E-07	1,25	8	0,06	0,001
3	5,46E-06	384	5	250	8	2	2,00E-04	0,9	512	15	5E-08	0,95	12	0,04	0
5	7,15E-09	490	6	250	16	2	2,01E-04	1,1	512	30	1E-07	1,5	12	0,06	0,001
7	9,95E-09	448	4	180	12	1	8,19E-05	0,95	512	20	5E-08	2	8	0,05	0,01
8	1,23E-06	490	3	250	8	1	7,19E-05	0,95	512	15	1E-08	0,5	8	0,05	0
11	1,85E-06	384	4	180	8	1	8,39E-05	1,01	512	10	1E-08	2	12	0,04	0,1

Incoporação

Config.	Adam epsilon	Doc stride	Early stopping patience	Evaluate during training steps	Eval batch size	Gradient accum. steps	Learning rate	Max grad norm	Max seq length	Num train epochs	Polynom. decay schedule lr end	Polynom. decay sched. power	Train batch size	Warmup ratio	Weight decay
1	3,78E-10	320	5	180	16	2	1,18E-04	1,01	384	40	5E-08	0,5	12	0,06	0
4	8,79E-07	490	2	250	8	2	5,73E-05	0,99	384	40	5E-09	1,25	8	0,05	0
7	1,48E-06	320	5	250	16	2	1,79E-04	0,95	512	30	1E-08	1,5	16	0,05	0,01
8	1,50E-11	192	4	180	16	2	9,24E-05	1,05	384	30	5E-08	2	16	0,1	0,001
10	3,50E-06	384	2	250	12	1	1,15E-04	0,99	384	10	1E-08	0,5	12	0,05	0
11	7,36E-10	256	4	250	8	2	7,72E-05	0,9	256	25	5E-09	1,5	12	0,06	0,1

Hiperparâmetros comuns: manual seed = 1, max answer length = 150, max query length = 15, n best size = 20, optimizer = AdamW, reprocess input data = 0, warmup steps = 0

Apêndice 9 – Exemplo de documento original

CELULOSE IRANI S.A.
CNPJ Nº 92.791.243/0001-03 NIRE Nº 43300002799
COMPANHIA ABERTA

AVISO AOS ACIONISTAS

AUMENTO DE CAPITAL POR SUBSCRIÇÃO PARTICULAR

CELULOSE IRANI S.A. ("Companhia") vem comunicar aos Senhores Acionistas que a Assembleia Geral Extraordinária de Acionistas realizada em 16 de outubro de 2013 aprovou o aumento do capital social da Companhia, de acordo com os termos e condições abaixo descritos:

AUMENTO DO CAPITAL SOCIAL: O capital social da Companhia foi aumentado no montante de R$ 12.918.356,62, passando para R$ 116.894.847,81.

QUANTIDADES/ESPÉCIES/CARACTERÍSTICAS DAS AÇÕES EMITIDAS: Emitidas 4.630.235 ações ordinárias escriturais, todas com características idênticas às atualmente existentes, com participação integral nos dividendos, juros sobre o capital próprio e/ou eventuais remunerações de capital que vierem a ser declaradas pela Companhia, a partir de 16.10.2013.

PREÇO DE EMISSÃO POR AÇÃO: O preço de emissão por ação é de R$ 2,790000209, o qual foi fixado com base no valor de patrimônio líquido por ação da Companhia, nos termos do disposto no artigo 170, §1º, II da Lei das S.A. na data base de 31.08.2013 .

SUBSCRIÇÃO E INTEGRALIZAÇÃO DAS AÇÕES: As ações ordinárias escriturais emitidas foram totalmente subscritas pela acionista Irani Participações S.A. ("Irapar"), e integralizadas com ações da sociedade Wave Participações S.A., avaliadas em R$12.918.356,62.

DIREITO DE PREFERÊNCIA: Os acionistas da Companhia detentores de ações, em 16 de outubro de 2013, exercerão o direito de preferência na subscrição de ações, na proporção de 0,02899080793 por ação detida, na presente data, independente do tipo e espécie da ação que o mesmo possua no capital social da Companhia.

NEGOCIAÇÃO EX-SUBSCRIÇÃO: As ações de emissão da Companhia adquiridas a partir desta data (17/10/2013, inclusive) não exercerão o direito de preferência para subscrição do referido aumento de capital e, serão negociadas ex-subscrição.

FORMA DE INTEGRALIZAÇÃO: As ações subscritas pelos acionistas no aumento de capital deverão ser integralizadas, no ato da subscrição, à vista e em moeda corrente nacional, sendo que tal valor será creditado à Irapar, nos termos do disposto no artigo 171, §2º da Lei n. 6.404/76 ("Lei das S.A.").

PRAZO PARA SUBSCRIÇÕES: O direito de preferência deverá ser exercido no prazo de 30 dias a contar da publicação deste aviso, iniciando-se, em 17 de outubro e