Universidade de Brasília
Faculdade de Tecnologia

Extração de Dados com Aprendizagem de Máquina para Processamento de Informações em Diários Oficiais

Antônio Aureliano de Anicésio Cardoso

TRABALHO DE GRADUAÇÃO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasilía

2022

Universidade de Brasília
Faculdade de Tecnologia

Extração de Dados com Aprendizagem de Máquina para Processamento de Informações em Diários Oficiais

Antônio Aureliano de Anicésio Cardoso

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. Flávio de Barros Vidal

Brasilía
2022

Cardoso, Antônio Aureliano de Anicésio.
C268e Extração de Dados com Aprendizagem de Máquina para Processamento de Informações em Diários Oficiais / Antônio Aureliano de Anicésio Cardoso; orientador Flávio de Barros Vidal. -- Brasília, 2022.
67 p.

Trabalho de Graduação em Engenharia de Controle e Automação -- Universidade de Brasília, 2022.

1. Extração de Dados de bases públicas. 2. Segmentação de Textos. 3. Aprendizado de Máquina. I. Vidal, Flávio de Barros, orient. II. Título

Universidade de Brasília
Faculdade de Tecnologia

Extração de Dados com Aprendizagem de Máquina para Processamento de Informações em Diários Oficiais

Antônio Aureliano de Anicésio Cardoso

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação.

Trabalho aprovado. Brasilía, 20 de Setembro de 2022:

Prof. Flávio de Barros Vidal, UnB/CIC
Orientador

Prof. Dra. Aleteia Patricia Favacho de Araujo - IE/CIC/UnB
Examinador interno

Prof. Dr. Jones Yudi Mori Alves da Silva - FT/ENM/UnB
Examinador interno

Brasilía
2022

Agradecimentos

Gostaria de agradecer a todas as pessoas que estiveram envolvidos de forma direta ou indireta na construção deste trabalho, em especial ao professor Flávio de Barros Vidal pela paciência e apoio durante a realização deste projeto, e aos membros do projeto Deep Vacuity.

Com participação indireta, gostaria de agradecer a minha família pelo suporte e apoio durante a realização deste projeto.

Resumo

A aplicação de técnicas de aprendizagem de máquina em dados textuais tem crescido exponencialmente nos últimos anos, principalmente devido à alta disponibilidade de modelos mais complexos e às enormes quantidades de dados disponíveis para esta tarefa. Para que estes dados tenham utilidade, a grande maioria dos algoritmos de aprendizagem de máquina normalmente necessita que os mesmos sejam pré-processados e anotados de alguma forma para serem utilizados no treinamento de modelos supervisionados. Um fator importante na utilização das informações obtidas de dados textuais a partir deste tipo de abordagem é que os mesmos são normalmente indexados em grandes bases de dados para busca. A correta segmentação de porções de textos a partir dos arquivos brutos pode facilitar muito a busca de informações e mesmo processamentos adicionais destes textos. No entanto, a qualidade dos dados públicos varia muito, e uma grande parte dos dados públicos disponíveis são publicados em formatos que não foram feitos para este tipo de processamento. Por exemplo, no caso dos dados publicados em diários oficiais pelos diversos entes da federação no Brasil, a maioria dos municípios e estados disponibiliza os dados em formato PDF, que apresenta diversas dificuldades, enquanto que o Diário Oficial da União (DOU) já possui protocolos de publicação que facilitam o processamento de dados. Neste trabalho realizou-se uma implementação que fosse capaz de realizar a segmentação de publicações de diários oficiais a partir dos arquivos PDF provenientes dos repositórios oficiais em que estas publicações são disponibilizadas ao público em geral. A implementação realizada fez uso de técnicas de aprendizado supervisionado e testou-se diversos modelos esparsos disponíveis nas principais ferramentas de bibliotecas públicas para verificar e comparar o seu desempenho utilizando um *benchmark* previamente definido.

Palavras-chave: Extração de Dados de bases públicas. Segmentação de Textos. Aprendizado de Máquina.

Abstract

The application of machine learning techniques to textual data has grown exponentially in recent years, mainly due to the high availability of more complex models and the enormous amounts of data available for this task. For these data to be valuable, most machine learning algorithms usually need pre-processed and annotated to be used in supervised training models. An essential factor in the use of information obtained from textual data from this type of approach is that they are usually indexed in large databases for searching. Correctly segmenting text portions from raw files can significantly facilitate the search for information and even further processing of these texts. However, the quality of public data varies widely, and a large part of the shared data is published in formats not designed for this type of processing. For example, in the case of data published in official journals by the various entities of the federation in Brazil, most municipalities and states make the data available in PDF format, which presents several difficulties. In contrast, the Official Gazette of the Union (DOU) already has publication protocols that facilitate data processing. In this work, an implementation was carried out that was able to segment official journal publications from PDF files from the official repositories in which these publications are made available to the general public. The implementation used supervised learning techniques, and several sparse models available in the main tools of public libraries were tested to verify and compare their performance using a previously defined benchmark.

Keywords: Data extraction of public data. Text Segmentation. Machine Learning.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Motivação

O campo de aprendizado de máquina, ou *machine learning*, do inglês, tem ganhado importância cada vez maior nas nossas vidas, na economia, na indústria e na sociedade de uma forma geral. A aplicação das técnicas de aprendizado de máquina teve um crescimento extraordinário nos últimos anos devido à confluência de dois fatores: a disponibilidade de quantidades cada vez maiores de dados para treinamento de modelos computacionais e a disponibilidade do poder computacional cada vez maior para efetivamente treinar e utilizar estes modelos (POTHEN, 2022), (LIU et al., 2018).

Dentro do contexto do aprendizado de máquina, e da inteligência artificial em geral, a criação de sistemas capazes de obter informações a partir de dados textuais, e usar estas informações para fornecer serviços úteis, constitui uma das áreas de aplicação mais promissoras. Isso se deve a seu potencial para agilizar serviços e facilitar a busca e integração de grandes quantidades de informação. Várias disciplinas tem como foco total ou parcial a utilização de métodos computacionais para compreensão textual. Dentre estas pode-se citar o Processamento de Linguagem Natural (PLN, ou NLP na sigla em inglês - *Natural Language Processing*), Linguística Computacional e de certa forma até a psicologia e a neurociência. Dentre as aplicações mais comuns do aprendizado de máquinas voltado à utilização de dados textuais pode-se citar a checagem e correção gramaticais, a tradução de textos de uma linguagem a outra, a busca de informações em bases de dados, a criação de *chatbots* e assistentes virtuais, dentre outras.

Um conjunto de áreas específicas no qual a aplicação destas técnicas pode ter um grande impacto para a sociedade são as áreas jurídica e governamental. Atualmente, tem crescido muito o número de tribunais e órgãos governamentais que utilizam algum tipo de inteligência artificial em seus trabalhos, e o campo do processamento de linguagem natural tem sido o mais importante neste contexto. Aplicações como indexação e busca inteligente de informações em bases de dados de processos jurídicos, criação de contratos inteligentes e detecção e prevenção de fraudes em processos públicos são cada vez mais comuns nestes cenários e apresentam um grande potencial para agilizar processos judiciais e ajudar a prevenir a corrupção no poder público.

1.2 Justificativa

Para que as decisões dos órgãos do governo entrem em vigor, é necessário que as mesmas sejam comunicadas à população de uma maneira acessível. No Brasil, os atos oficiais da administração pública executiva, legislativa e judiciária são publicados em jornais denominados diários oficiais. A União, os Estados, o Distrito Federal e cada município do país (além de tribunais e outros órgãos) mantém diários oficiais para publicação de seus atos. Dependendo do tamanho da entidade e dos recursos disponíveis, estes diários podem ser mantidos com recursos próprios e contar com diversas seções, como é o caso do DOU e dos diários dos estados e dos maiores municípios do país ou, como acontece com a maioria dos municípios, serem diários breves e muitas vezes publicados em conjunto com outros municípios na forma de associações de municípios estaduais.

Os tipos de publicações contidas nos diários oficiais podem variar muito, mas as mais comuns e, para os objetivos deste trabalho as mais informativas em termos de possíveis atividades ilegais, são licitações, convênios e contratos. A seguir são descritas brevemente estas modalidades:

- **Licitações** são processos públicos abertos para a seleção de empresas a serem contratadas para a realização de obras públicas. As licitações possuem três modalidades básicas: Concorrência, Tomada de Preços e Convite. A modalidade de concorrência normalmente envolve os maiores recursos e, por isso, envolve mais etapas de controle e mais publicações são realizadas;
- **Convênios** são assinados quando há a passagem de recursos de um ente federado, normalmente a União, para outro para a realização de alguma obra ou projeto. Os termos do acordo normalmente precisam incluir cláusulas de compromisso, prazos e valores;
- **Contratos** são firmados após a finalização de um processo licitatório entre o ente responsável pela licitação e a empresa vencedora. Deve mencionar os nomes das partes e os de seus representantes, a finalidade, o ato que autorizou a sua lavratura, o número do processo da licitação, da dispensa ou da inexigibilidade.

Por meio da extração de informações deste tipo de publicação e sua comparação entre publicações e ao longo do tempo, é possível detectar comportamentos como conluio ou formação de cartel, como por exemplo fixação de preços e divisão de mercado. Estes ocorrem quando os principais concorrentes nos processos licitatórios entram em acordo para cobrar preços mais elevados, prejudicando o processo e a sociedade.

1.3 Objetivos

Neste trabalho, se buscou atingir os seguintes objetivos geral e específicos, descritos a seguir.

1.3.1 Objetivo Geral

Implementar as partes constituintes de um sistema capaz de, a partir dos arquivos dos diários oficiais dos estados e municípios do Brasil, segmentar as publicações individuais contidas nestes diários para posterior indexação. Para isso, será implementado um módulo capaz de realizar refluxo de textos de arquivos pdf de duas colunas para uma coluna, seguido pelo reconhecimento óptico de caracteres destes para obtenção do texto final. Em seguida será treinado um classificador para separar cada publicação a partir de textos anotados manualmente.

1.3.2 Objetivos Específicos

Para cumprir o objetivo geral, faz-se necessário atingir os seguintes objetivos específicos:

- Obter os arquivos brutos dos diários de uma parcela representativa dos municípios e estados brasileiros;
- Implementar um módulo de processamento capaz de extrair os textos destes arquivos utilizando um protocolo previamente definido;
- Montar uma base de dados a partir dos textos extraídos e realizar a anotação desta base para treinamento do sistema de segmentação de publicações, por meio do treinamento de modelos de aprendizagem de máquina supervisionados;
- Realização do treinamento e validação do módulo que realize, a partir dos textos extraídos dos diários oficiais de estados e municípios, a segmentação destes gerando publicações individuais.

1.4 Organização do Trabalho

Este trabalho encontra-se dividido da seguinte maneira: O capítulo 2 apresenta aspectos teóricos das diversas partes necessárias para realização deste trabalho, como extração de dados automatizadas de bases de dados em sítios web, processamento de textos a partir de arquivos diversos, aprendizado de máquina e processamento de linguagem natural. O capítulo 3 apresenta a descrição de alguns trabalhos relacionados com este trabalho: o projeto

Querido Diário[1], para extração de diários oficiais dos municípios brasileiros e sua indexação, e o projeto **Deep-Vacuity**[2], que buscam utilizar as informações contidas nos diários oficiais dos diversos entes da Federação para detectar e prevenir atividades de conluio e cartel das empresas prestadoras de serviço público. O capítulo 4 apresenta em mais detalhes a metodologia específica utilizada para desenvolver cada uma das etapas do trabalho, enquanto o capítulo 5 apresenta os resultados obtidos. No capítulo 6 são apresentadas as conclusões do trabalho realizado e as propostas futuras.

[1] Disponível em https://queridodiario.ok.org.br, acessado em 31/08/2022.
[2] Disponível em https://deepvacuity.cic.unb.br, acessado em 31/08/2022.

2 Aprendizagem de Máquina para Dados Textuais

Neste capítulo são apresentados conceitos teóricos a respeito das técnicas utilizadas nas diversas etapas desenvolvidas ao longo deste trabalho. Começa-se por uma descrição geral do fluxo de trabalho padrão que costuma ser seguido em trabalhos de aprendizagem de máquina. Em seguida são apresentadas as técnicas de *web crawling* e *web scraping*, importantes para a obtenção de grandes quantidades de dados da Internet para a construção de bases de dados. Depois disso é feita uma apresentação geral sobre métodos de processamento de texto, em especial de arquivos PDF. Em seguida são abordados os conceitos de aprendizagem de máquinas em si e os diversos tipos de aprendizagem utilizados. Por fim são discutidos mais especificamente as técnicas utilizadas em Processamento de Linguagem Natural (PLN).

2.1 Fluxo de Trabalho Padrão

Trabalhos de aplicação de aprendizagem de máquina para classificação de textos normalmente seguem um fluxo bem definido de etapas para sua implementação, embora os detalhes envolvidos em cada etapa possam diferir bastante entre um projeto e outro. Uma referência muito utilizada é o fluxo de trabalho com dados textuais publicado pelo Google (GOOGLE, 2022). A Figura 1 apresenta o esquema geral deste fluxo. As etapas descritas na Figura 1 consistem em:

Figura 1 – Fluxo do trabalho com dados.

Fonte: https://developers.google.com/machine-learning/guides/text-classification/

- **Coleta de dados:** todo projeto de aprendizagem de máquina, e em especial aqueles que lidam com dados textuais, são limitados pela qualidade dos dados utilizados. Por isso é de fundamental importância garantir que os dados coletados sejam representativos das classes que se deseja classificar. Esta etapa normalmente envolve processos de *web crawling* e *web scraping*;

- **Exploração dos dados:** consiste em tentar entender melhor as características dos dados para a construção de um modelo melhor;
- **Preparação dos dados:** consiste em deixar os dados em formato padrão para treinamento dos modelos selecionados. Um exemplo seria a anotação de textos com os rótulos que o sistema precisará identificar. Esta é normalmente a parte mais demorada do trabalho;
- **Criação, treinamento e validação do modelo:** consiste no processo de treinamento e validação dos modelos, parte central de um trabalho com aprendizagem de máquina. É importante nesta parte tentar garantir que os dados não estejam enviesados. Uma forma de se fazer isso é selecionar aleatoriamente os conjuntos de treinamento e de testes e realizar procedimentos de validação cruzada;
- **Ajuste de hiper-parâmetros:** consiste em explorar diferentes configurações de hiper-parâmetros dos modelos selecionados com o objetivo de definir qual configuração fornece melhores resultados;
- **Implantação do modelo:** implanta-se o modelo treinado em um ambiente de produção.

A seguir serão apresentados mais detalhes das tecnologias mais importantes para esse trabalho dentro do contexto do aprendizado de máquinas para dados textuais. Primeiro serão apresentados os processos de *web scraping*. Em seguida os métodos de processamento de textos são descritos, especificamente para arquivos PDF. Por fim será apresentada uma introdução geral sobre conceitos de Aprendizado de Máquinas e Processamento de Linguagem Natural (PLN).

2.2 Extração de Dados: *Web Crawling* e *Web Scraping*

Web crawlers são sistemas utilizados para o *download* em massa de páginas da *web* (KHDER, 2021). Dentre suas principais aplicações, é possível citar a construção de mecanismos de busca. Neste caso, são utilizados para reunir um banco de páginas da *web*, indexá-las e permitir que usuários façam buscas no índice para encontrar as páginas mais relevantes. Outra aplicação comum é a construção de arquivos web, onde são utilizados para coletar periodicamente grandes conjuntos de páginas e salvá-las para a posteridade. Também pode-se citar a mineração de dados como aplicação, onde páginas são analisadas para extrair estatísticas delas (RAMAGERI, 2010). Alguns exemplos de web crawlers utilizados por grandes empresas são:

- Amazonbot é o *web crawler* da Amazon para identificação de conteúdo da web e descoberta de backlinks[1];
- Baiduspider é o principal *web crawler* para o mecanismo de busca da Baidu[2];
- Bingbot da Bing[3];
- DuckDuckBot da DuckDuckGo[4];
- Googlebot é o *web crawler* do Google[5];
- Yahoo! Slurp da Yahoo[6];

Um *web crawler* é definido como um programa que atravessa a web e faz o *download* de documentos de uma maneira metódica e automatizada (ABU KAUSAR; DHAKA; SINGH, 2013). Os termos *web crawling* e *web scraping* são frequentemente confundidos. Aqui *web crawling* será tratado como apenas o acesso e percorrimento automático de páginas web, enquanto que *web scraping* será definido como a extração ou *download* de informações específicas de um conjunto de páginas de interesse. Neste sentido, *web crawling* normalmente é um primeiro passo para a realização de *web scraping*.

A maioria dos *crawlers* funcionam a partir de uma lista pré-definida de páginas alvo e procedem extraindo todos os links e informações contidas nestas páginas. A partir daí, os *crawlers* normalmente procedem seguindo todos os *links* extraídos em uma determinada ordem e seguindo o mesmo processo para estas novas páginas.

Enquanto *crawlers* podem apresentar um certo grau de generalidade, no sentido de que um *crawler* não necessariamente precisa ser construído para percorrer uma página ou conjunto específico de páginas, o mesmo não pode ser dito a respeito dos *scrapers*. A maioria das páginas da *web* têm uma estrutura extremamente específica e ainda não é possível criar um sistema que seja capaz de extrair os dados desejados de um conjunto muito heterogêneo de páginas. Por isso a construção de *scrapers* é normalmente direcionada para páginas específicas e, em consequência, pode ser um processo demorado e trabalhoso. Por isso, a utilização de *frameworks* para construção de *scrapers* apresenta enormes vantagens. A seguir será apresentado o *framework Scrapy*, um dos mais populares para a criação de scrapers em linguagem Python.

[1] https://developer.amazon.com/support/amazonbot
[2] https://qpsoftware.net/blog/baiduspider
[3] https://rockcontent.com/br/blog/bingbot/
[4] https://help.duckduckgo.com/duckduckgo-help-pages/results/duckduckbot/
[5] https://developers.google.com/search/docs/advanced/crawling/googlebot
[6] https://help.yahoo.com/kb/SLN22600.html

2.2.1 O Módulo *Scrapy*

Scrapy[7] é um *framework* completo para implementação de projetos de *web crawling* e raspagem de dados em linguagem Python. É atualmente a biblioteca de código aberto mais completa para implementação de projetos de extração de dados da web, e permite o controle sobre cada etapa do processo. Este é o *framework* utilizado no projeto Querido Diário, que foi usado neste projeto para extração de dados. A seguir é apresentado, em detalhes, o funcionamento do sistema. A Figura 2 apresenta a arquitetura básica do *framework*.

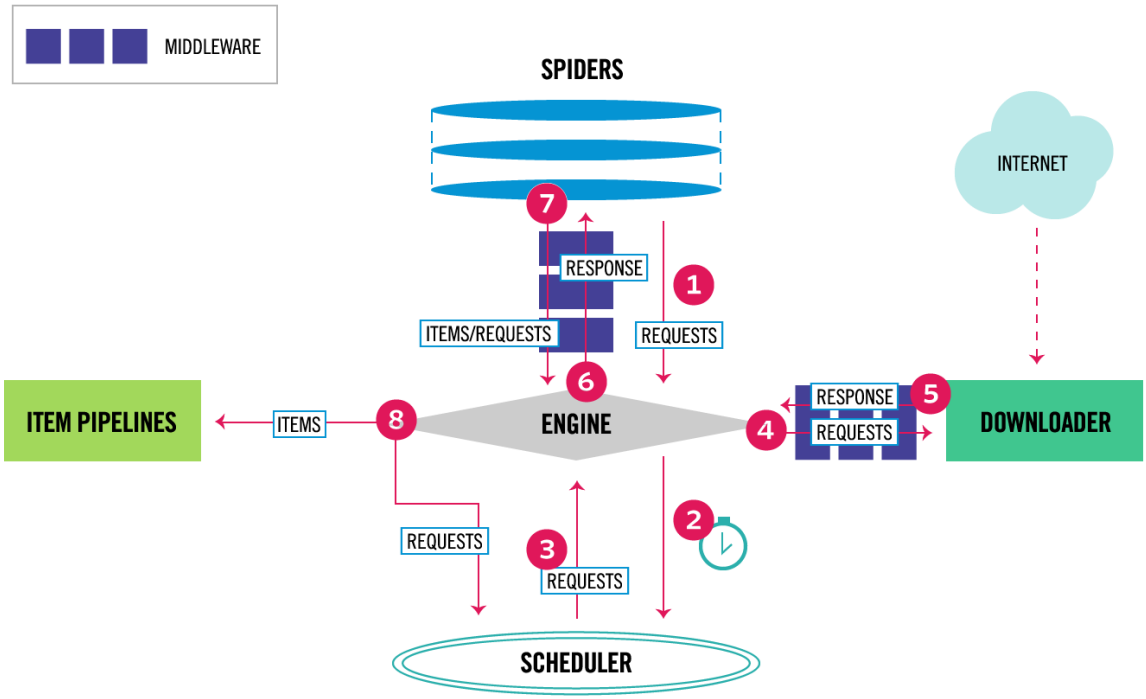


Figura 2 – Arquitetura do *framework Scrapy*.

Fonte: https://docs.scrapy.org

A arquitetura é razoavelmente complexa, mas o processo básico de extração de dados com o *Scrapy* tem início com a programação de extratores de dados, ou *spiders*, específicos para cada projeto criado com o framework. Estes *spiders* definem um conjunto de URLs iniciais. A partir destas URLs a *Engine* do Scrapy irá obter um conjunto de requisiçõess (1). Estas requisições são transferidas para um *scheduler* (2) para inserção na fila de requisições e a requisição no início da fila (3) é trasferida de volta à *Engine* e passado para um *Downloader* (4). Este realizará o *download* da página especificada e retornará o conteúdo (5) para a *Engine*. A *Engine* retorna o conteúdo para os *spiders* (6) para que estes realizem a seleção das informações que desejam extrair da página. O processo pode acabar aqui, para a extração de dados mais simples que estejam presentes no código fonte da página, ou pode continuar

[7] https://docs.scrapy.org/en/latest/

para a extração de dados mais complexos, como imagens, arquivos, etc. Neste caso, o *spider* retorna à *Engine* uma descrição dos itens que deseja extrair da página (7). Estes Itens, que serão processados no *pipeline* de itens do *Scrapy* (8) são estruturas de dados semelhantes a dicionários que podem conter campos arbitrários com dados da página ou URLs para extração de imagens ou arquivos. O *spider* também pode solicitar à *Engine* a realização de novas requisições (8), caso comum na extração de páginas sequenciais em um servidor.

2.2.2 Aspectos Legais

É importante ressaltar que extratores de dados podem direcionar uma grande quantidade de tráfego simultaneamente para as páginas web das quais desejam extrair informações. Isto pode causar problemas para o servidor ou pode ser tratado como um ataque do tipo negação de serviço. Por isso, nem todos os sites permitem que seu conteúdo seja extraído desta forma, e a violação destas regras pode resultar em sanções legais em certas jurisdições (KROTOV; SILVA, 2018).

Para saber se um *site* permite ou não a extração de dados, é importante checar o arquivo "robots.txt"do *site*. Este arquivo consiste em um mecanismo padronizado por meio do qual os sites informam se permitem ou não o uso de *web crawlers* e *scrapers* e, se sim, quais partes do *site* podem ser acessadas por estes sistemas.

Além de garantir que seu sistema obedeça as regras do arquivo "robost.txt", é importante checar outros detalhes antes de implementar um projeto de *web scraping* em larga escala. Por exemplo, consultar os termos de serviço do *site*, consultar as leis do local em que o *site* está hospedado, principalmente no tocante a proteção de dados, e ajustar as taxas de envio de requisições dos projetos implementados para não inundar o servidor com requisições. Outro ponto importante a se tomar cuidado é na extração de conteúdos que possuam direitos autorais (KIENLE; MÜLLER, 2013).

2.3 Processamento de Textos

De posse dos arquivos brutos com os textos que deseja-se processar, a próxima etapa do fluxo de trabalho envolve o tratamento destes arquivos para se obter o texto desejado em um formato que facilite os trabalhos subsequentes. O trabalho necessário para esta etapa pode variar bastante dependendo do formato dos arquivos originais. Para arquivos obtidos da Internet em formato html, por exemplo, os textos desejados podem ser extraídos do documento de maneira relativamente simples com o auxílio de *parsers* para a linguagem html. No caso da linguagem python, há um parser de html na biblioteca padrão da linguagem, no módulo HTML. Outras opções de bibliotecas para esta tarefa também são muito populares,

como *BeautifulSoup*[8] e *lxml*[9].

Existem vários outros formatos de arquivos que podem ser utilizados para obter dados para aprendizagem de máquina aplicado a textos. Alguns dos mais comuns são o formato CSV, para dados tabulares, o formato XML para dados semi-estruturados e o formato JSON, utilizado para serialização e armazenamento de dados como coleções de objetos (CARR, 2008).

Entretanto, um dos formatos mais utilizados para armazenamento e transmissão de textos (e documentos de uma forma geral) é o formato PDF. Uma das surpresas encontradas no decorrer deste trabalho foi a relativa dificuldade em se obter os textos a partir de arquivos PDF, principalmente quando os arquivos obtidos continham texto formatado com mais de uma coluna ao longo do documento. Mas, mesmo para os arquivos com apenas uma coluna de texto, o trabalho de extrair este texto se mostrou desafiador. Isso se deve, principalmente, à estrutura do formato PDF, apresentada a seguir, que armazena informações sobre o conteúdo do arquivo como objetos para apresentação padronizada nas telas de diversos dispositivos, e não distingue muito bem sobre a noção de texto (ADOBE, 2004).

As próximas seções apresentam brevemente a estrutura básica dos arquivos PDF, bem como algumas das principais ferramentas disponíveis para trabalhar com arquivos deste formato em código. A seguir são apresentados alguns dos resultados obtidos ao utilizar estas ferramentas para tentar extrair textos dos arquivos PDF. Tendo em vista que os resultados obtidos com estas ferramentas para extração direta do texto não foram satisfatórios, principalmente para textos de duas ou mais colunas, optou-se por utilizar uma ferramenta de refluxo de texto em conjunto com uma ferramente de OCR para obtenção dos textos dos diários.

2.3.1 Formato PDF

O Portable Document Format (PDF) (ADOBE, 2004) foi criado pela Adobe na década de 1990 com o intuito de servir como um padrão para a visualização de documentos e materiais de referência, independente de software, hardware ou sistema operacional. O formato é capaz de conter dados como textos, imagens, *hyperlinks*, formulários, assinaturas digitais, anexos, metadados, dados geoespaciais, figuras 3D, entre outros.

A estrutura básica de um arquivo PDF é mostrada na Figura 3. O arquivo é composto de um cabeçalho, o corpo do arquivo, uma tabela de referências e uma seção trailer. O cabeçalho, ou *header*, é a primeira linha de um arquivo PDF e contém o número da versão da especificação do formato PDF utilizada naquele arquivo. Quando visualizada em um editor de texto, esta linha tem o formato "%PDF-1.x", onde x pode variar entre 1 e 7. O corpo do arquivo PDF contém uma sequência de objetos, os quais podem representar todos os tipos

[8] https://beautiful-soup-4.readthedocs.io/en/latest/
[9] https://lxml.de/

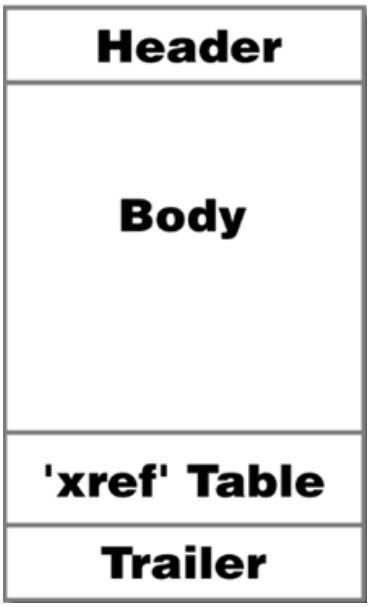


Figura 3 – Formato de um arquivo PDF.

Fonte: https://resources.infosecinstitute.com/topic/pdf-file-format-basic-structure/

de dados citados anteriormente, como texto, imagens, etc. A tabela de referências permite o acesso randômico aos objetos do arquivo, de modo que o arquivo inteiro não precise ser atravessado para se localizar um objeto em particular. A tabela contém entradas de uma linha para cada objeto no arquivo, especificando o offset em bytes para aquele objeto. A seção trailer permite que o programa leitor de arquivos PDF acesse a tabela de referências e certos objetos especiais. Por este motivo, leitores de PDF normalmente leem o arquivo começando pelo final.

Pode-se notar pela exposição acima que em um arquivo PDF, todos os elementos que são visualizados na tela, quando se abre um arquivo PDF, são armazenados como objetos (ADOBE, 2004). Estes podem ser de um tos seguintes tipos:

- Valores booleanos - representam verdadeiro ou falso;
- Números - valores inteiros ou reais;
- *Strings* - contêm caracteres entre parênteses;
- Nomes - valores que iniciam com "/";
- *Arrays* - arrays de uma dimensão;
- Dicionários - Coleções de objetos como pares chave-valor;
- *Streams* - representam sequências de bytes;

- Objeto nulo - representam um valor nulo.

Desta forma fica mais fácil entender a dificuldade de extrair textos diretamente de arquivos PDF. Embora o formato PDF possua um objeto do tipo string, dados textuais não são armazenados como parágrafos ou palavras, mas como caracteres a serem desenhados em certas coordenadas da tela, de forma que muitas vezes, ao tentar extrair diretamente esse texto, o que se obtém é uma emaranhado de palavras. Isso ocorre porque o formato PDF nunca foi projetado para ser um formato de entrada de dados, mas principalmente um formato de armazenamento e visualização.

2.3.2 Bibliotecas para Processamento de Arquivos PDF

Inúmeras bibliotecas estão disponíveis para trabalhar com arquivos PDF a partir de código de programação. Dentre as principais bibliotecas pesquisadas para este trabalho pode-se citar: *Ghostscript, xpdf, Apache Tika, PyPDF2* e *PDFMiner*.

Ghostscript[10] é um conjunto de ferramentas para manipulação, edição e criação de arquivos nos formatos *Postscript* e PDF. Trata-se de uma das ferramentas mais antigas e mais completas para manipulação de arquivos nestes formatos. O software atualmente é distribuído em duas licenças, uma na versão GPL e uma versão comercial. Neste trabalho foi testado o recurso de extração de textos de arquivos PDF através do comando

```
gswin64 -sDEVICE=txtwrite -ooutput2.txt test.pdf
```

onde gswin64 é o comando, test.pdf é o arquivo pdf e output.txt é o arquivo de saída.

XPDF[11] é um software de código aberto que conta com um leitor de arquivos PDF e diversas ferramentas para conversão de arquivos PDF para outros formatos e extração de conteúdos específicos dos arquivos. Aquela testada neste trabalho foi a ferramenta *pdftotext*, para extração de textos, através do comando

```
"pdftotext test.pdf output.txt"
```

onde novamente test.pdf é o arquivo PDF original e output.txt é o arquivo com o texto extraído.

Apache Tika[12] é um framework que apresenta um conjunto de ferramentas para detecção e extração de de texto e metadado de mais de 1000 formatos de arquivo diferentes. O projeto foi iniciado em 2007 na Fundação Apache e sua flexibilidade de formatos suportados o tornam atraente para aplicações em indexação para mecanismos de busca, análise de

[10] https://ghostscript.com/doc/current/Readme.htm
[11] https://www.xpdfreader.com/support.html
[12] https://tika.apache.org/

conteúdo, tradução, entre outros. As ferramentas podem ser utilizadas de diversas formas, incluindo uma API (*Application Programming Interface*) em Java e na linha de comando, através de um arquivo .jar incluído no projeto. O segundo método foi testado neste trabalho, através do comando

```
"java -jar tika-app-2.4.1.jar -text test.pdf > output.txt"
```

PyPDF2[13] é uma biblioteca de código aberto escrita em linguagem Python, capaz de realizar diversas operações com arquivos PDF, como por exemplo separar um arquivo em múltiplas partes, mesclar diferentes arquivos em um só, realizar manipulação de páginas, dentre outros. Neste trabalho foram testadas as funções da biblioteca referentes a extração de texto.

PDFMiner[14], assim como PyPDF2, é uma biblioteca de código aberto em Python para manipulação de PDFs. As duas bibliotecas são muito similares em relação a suas funcionalidades, por isso foram testadas neste trabalho para comparação. Para extração de textos, *PDFMiner* disponibiliza um *script* em Python, "pdf2txt.py", e o mesmo foi testado através do comando

```
"python3 pdf2txt.py -o output.txt test.pdf"
```

Após todos os testes descritos, foi constatado neste trabalho que todas as ferramentas eram capazes de realizar razoavelmente bem a extração completa de textos de uma coluna. Porém, nenhuma das ferramentas foi capaz de extrair satisfatoriamente textos de duas colunas. Mais detalhes serão discutidos no Capítulo 5.

2.3.3 Refluxo de Textos

O termo refluxo de texto refere-se a um meio de mudar automaticamente o leiaute de um texto para que se ajuste às dimensões da página, tela ou janela onde está sendo visualizado (PANJWANI; UPPAL; CUTRELL, 2011). É o que acontece na maiorias das páginas da web quando uma janela de um navegador tem seu tamanho reajustado. O refluxo de texto é uma tarefa particularmente importante para dispositivos *e-readers*, situação na qual cada dispositivo possui um tamanho diferente de tela.

O refluxo de texto trata-se de uma tarefa fácil para formatos de arquivo como epub[15], projetados para visualização em diversos tamanhos de dispositivo, mas é consideravelmente mais complicado para um formato estático como PDF. Neste caso, os métodos empregados para refluxo de texto normalmente fazem uso de processamento gráfico das partes individuais

[13] https://pypdf2.readthedocs.io/en/latest/index.html
[14] https://pdfminer-docs.readthedocs.io/pdfminer_index.html
[15] https://www.w3.org/publishing/epub3/epub-spec.html

do arquivo em conjunto com outros métodos de análise, seguidos pela remontagem do arquivo no formato desejado, para obtenção do resultado final.

2.3.4 Reconhecimento Óptico de Caracteres

O reconhecimento óptico de caracteres refere-se à conversão automática de caracteres, manuscritos ou impressos, a partir de imagens, em strings de texto (CHAUDHURI et al., 2017). Esta tecnologia possui aplicações em diversas áreas, das quais pode-se citar a entrada de dados a partir de registros impressos (como passaportes, recibos, extratos bancários, correspondências, entre outros), digitalização de textos escaneados de documentos legais e livros para armazenamento em arquivos digitais, leitura de placas de veículos em imagens de vigilância, tradução de textos em imagens e vídeo, mineração de dados textuais, dentre outras aplicações.

A tecnologia para reconhecimento de caracteres em imagens tem sido desenvolvida desde a década de 1970 (SCHANTZ; ASSOCIATION, 1982), com aplicações de leitura automática de texto para deficientes visuais, e tem progredido rapidamente nos últimos anos, principalmente com o surgimento de bibliotecas de código aberto. Dentre estas pode-se destacar a biblioteca *Tesseract* (S., 2016). Originalmente desenvolvida como um software proprietário pela *Hewlett Packard* na Inglaterra e nos Estados Unidos entre os anos de 1985 e 1994 (SMITH, 2013), a biblioteca teve seu código aberto em 2005, e a partir de 2006 o projeto continuou a ser desenvolvido pelo Google. Atualmente, o *Tesseract* é capaz de reconhecer textos em mais de 120 idiomas diferentes.

2.4 Aprendizado de Máquina

O aprendizado de máquina é um ramo da Inteligência Artificial que busca criar sistemas que aprendam a realizar tarefas a partir de dados de exemplo, ao invés de depender da programação individual de cada sistema para a tarefa em questão (ZAKI; MEIRA JR, 2020). O aprendizado de máquina não é uma área nova de pesquisa, tendo surgido e se desenvolvido paralelamente ao campo da inteligência artificial em si. Porém o campo do aprendizado de máquina começou realmente a ser desenvolvido separadamente a partir da década de 1970, quando a comunidade de pesquisadores em inteligência artificial praticamente abandonou o uso de modelos baseados em redes neurais por quase uma década (FRADKOV, 2020). Em consequência, o campo do aprendizado de máquina foi reorganizado como uma área de pesquisa separada e teve seu foco redirecionado para a solução de aplicações mais práticas com base em métodos probabilísticos e estatísticos (FRADKOV, 2020). A área teve progressos contínuos durante as próximas décadas, mas sua aplicação esteve limitada pela falta de um volume significativo de dados para treino dos modelos. Com a explosão da disponibilidade de dados nas duas primeiras décadas do século XXI, as técnicas

de aprendizado de máquinas obtiveram um crescimento exponencial em sua utilização e também no montante de financiamentos direcionados para pesquisas na área.

De maneira geral, pode-se dividir as técnicas de aprendizado de máquina em quatro tipos, baseado no grau de supervisão humana necessário para sua implementação:

- Aprendizado não-supervisionado
- Aprendizado supervisionado
- Aprendizado semi-supervisionado
- Aprendizado com reforço

Aprendizado supervisionado é o processo de treinar um computador utilizando uma base de dados anotada com rótulos que servem de exemplo para o resultado esperado que o sistema deve produzir para aquele dado de entrada específico (ZAKI; MEIRA JR, 2020). Após divisão da base de dados em conjuntos de treinamento e de teste, o modelo em questão é treinado no conjunto de treinamento e os resultados do treinamento são avaliados usando o conjunto de testes.

Aplicações de aprendizado supervisionado podem ser divididas em basicamente duas categorias, regressão e classificação (ZAKI; MEIRA JR, 2020). Em problemas de regressão, o resultado esperado do modelo é um valor numérico que pode variar continuamente, já no caso de problemas de classificação, o resultado esperado é uma classe.

No aprendizado não-supervisionado, o sistema é treinado usando uma base de dados sem nenhum tipo de anotações ou rótulos (ZAKI; MEIRA JR, 2020). Algoritmos de aprendizado não supervisionado analisam os dados fornecidos e tentam agrupa-los da melhor maneira possível de acordo com simetrias, padrões ou diferenças. Estes agrupamentos são chamados de *clusters* e, por isso, algoritmos de aprendizado não-supervisionado são também chamados de algoritmos de clusterização. Muitas vezes algoritmos de clusterização são utilizados nas primeiras etapas de exploração dos dados disponíveis para se ter uma melhor ideia das características dos dados (ZAKI; MEIRA JR, 2020).

O aprendizado semi-supervisionado é um tipo de meio termo entre os aprendizados supervisionado e não supervisionado. Nele, o sistema normalmente recebe dados tanto anotados quanto não anotados, sendo que a quantidade de dados não anotados costuma ser muito maior. Sua principal motivação é minimizar o trabalhoso processo de anotar bases de dados e tentar utilizar ao máximo os dados disponíveis (ZAKI; MEIRA JR, 2020).

O aprendizado com reforço é baseado no método de tentativa e erro. Nele o sistema tenta diversas linhas de ação e procura escolher aquela que fornece a maior recompensa ou o menor erro (ZAKI; MEIRA JR, 2020). Este tipo de aprendizado é muito comum em

aplicações de robótica móvel, onde um robô aprende a atravessar um labirinto ou pista de obstáculos por tentativa erro.

2.4.1 Ferramenta *Scikit-Learn*

Scikit-Learn[16] é uma biblioteca de código aberto em linguagem Python para aplicações de aprendizagem de máquina. A biblioteca conta com inúmeros modelos para tarefas como classificação, regressão e clusterização, além de inúmeras ferramentas para pré-processamento de dados, como extração de *features* e redução de dimensionalidade.

A interface padrão utilizada pelos modelos do Scikit-Learn foi projetada para ser intuitiva e fácil de usar. Por exemplo, um método "*fit*"é utilizado por todos os modelos para treinar o modelo nos dados e obter os parâmetros que governam o comportamento do modelo treinado, enquanto um método "*predict*"retorna os resultados do modelo treinado em um conjunto dados de teste.

2.5 Processamento de Linguagem natural

O processamento de linguagem natural (NLP) é uma subárea da Inteligência Artificial que busca obter automaticamente informações úteis sobre dados textuais, estruturados ou não (JURAFSKY; MARTIN, 2000). O NLP envolve uma ampla gama de técnicas que englobam métodos de pré-processamento de textos, vetorização e engenharia de features, métodos de classificação, entre outros. A seguir será apresentado um breve histórico do NLP, assim como as principais bibliotecas e técnicas empregadas em aplicações da área.

2.5.1 Histórico e Aplicações

As origens do NLP se remontam às primeiras tentativas de construir máquinas capazes de traduzir automaticamente entre uma língua e outra (DAHL, 2013). Em 1950, Alan Turing (TURING, 1950) propôs o Teste de Turing como um meio de julgar a presença de inteligência. Turing argumentou que se uma máquina pudesse engajar em conversação com um humano a partir de um dispositivo de texto, e se essa máquina fosse capaz de imitar um ser humano tão bem que a pessoa do outro lado do dispositivo fosse incapaz de reconhecer a diferença, então poderia-se dizer que a máquina possuía inteligência e pensamento próprio.

Em 1957, a teoria linguística foi revolucionada com a publicação do livro *Syntactic Structures*, de Noam Chomsky (CHOMSKY, 1957), o qual argumentava pela existência de uma "gramática universal"subjacente a todas as línguas humanas. Na década de 1960 e início da década de 1970, após alguns avanços promissores, como a criação de programas de computador como ELIZA (WEIZENBAUM, 1983), em 1966, e PARRY, em 1972, o campo do

[16] https://scikit-learn.org/stable/user_guide.html#user-guide

processamento de linguagem natural teve seu progresso quase que completamente parado devido a cortes de financiamento para pesquisa.

Foi apenas a partir de 1980 que o campo do processamento de linguagem natural, e a inteligência artificial como um todo, se recuperaram e retomaram o progresso (DAHL, 2013). A partir de então, após alguns progressos utilizando abordagens simbólicas, os algoritmos utilizados assumiram um caráter decididamente mais estatístico, auxiliados pela disponibilidade de maior poder computacional e pelo adoção de técnicas de aprendizagem de máquina.

A partir dos anos 1990, a popularidade de modelos estatísticos para o processamento de linguagem natural cresceu dramaticamente (DAHL, 2013), e a partir do final daquela década e início dos anos 2000, modelos baseados em redes neurais, como *Long-Short Term Memory* (LSTM) (DAHL, 2013) se tornaram o estado-da-arte no campo. A partir da década de 2010, o campo havia progredido tanto que foi possível a introdução comercial de assistentes virtuais em *smartphones* e computadores.

2.5.2 Pré-Processamento de Dados Textuais

Embora seja fácil para humanos interpretar textos contendo grandes quantidades de ruído, sentenças complexas e estruturas gramaticais elaboradas, o mesmo não é verdade dos programas de computador utilizados para processamento de linguagem natural (JURAFSKY; MARTIN, 2000). O pré-processamento de textos consiste nas etapas aplicadas aos dados textuais antes da aplicação de modelos para interpretação dos dados e seu objetivo é deixar o texto em um formato o mais regular possível para facilitar o trabalho dos algoritmos de NLP e aprimorar seu desempenho nos dados em questão.

Pode-se dividir o pré-processamento de textos basicamente em três tipos distintos (JURAFSKY; MARTIN, 2000), que normalmente são aplicados sequencialmente: Limpeza dos dados, tokenização e normalização. A seguir são apresentados estes três tipos de pré-processamento.

2.5.2.1 Limpeza de Dados

A limpeza de dados consiste em remover ruídos dos dados, na forma de caracteres ou palavras que não contribuem para o significado do texto e podem dificultar as tarefas de processamento pois podem aumentar a dimensionalidade dos dados (JURAFSKY; MARTIN, 2000). Alguns dos passos de limpeza de dados mais comuns para dados textuais são:

- Remoção de caracteres especiais, como *emojis* e outros símbolos.
- Remoção de *stopwords*. Trata-se de palavras que não contribuem para o sentido da frase, como artigos ou conjunções.

- Remoção de URLs
- Remoção de *tags* HTML
- Remoção de espaços extras

2.5.2.2 Tokenização

Tokenização consiste no processo de converter um texto em pedaços menores, denominados *tokens*. *Tokens* podem consistir em palavras, números, símbolos ou caracteres (JURAFSKY; MARTIN, 2000) . O tipo mais comum de tokenização é a tokenização por espaços em branco, onde considera-se um *token* qualquer conjunto de caracteres entre dois espaços. Por exemplo, a frase "Eu nasci em Brasília, em 1994."seria separada nos *tokens* "Eu", "nasci", "em", "Brasília", "em"e "1994".

2.5.2.3 Normalização

O objetivo da normalização é basicamente garantir a consistência dos dados que passarão por análise ou por mais pré-processamento (JURAFSKY; MARTIN, 2000). Pode-se citar três técnicas principais de normalização de dados utilizadas em NLP: normalização do caso de todas as palavras do texto, stemização e lematização. A redução de caso consiste basicamente na conversão de todos os caracteres do texto em letras minúsculas ou maiúsculas. Esta etapa pode ajudar a reduzir a dimensionalidade dos dados pois evita que a mesma palavra, com letras com caso diferente, possa ser contabilizada mais de uma vez durante a vetorização do texto.

Stemização é o processo de extrair a raiz de um *token* através da remoção de sufixos ou conjugações (JURAFSKY; MARTIN, 2000). Por exemplo, os *tokens* "trabalham", "trabalhando", "trabalhar", "trabalho"e "trabalhos"podem ser convertidos na mesma raiz: "trabalh".

A lematização é o processo de reduzir uma palavra ao seu lema básico baseado no sentido pretendido da palavra (JURAFSKY; MARTIN, 2000). Lematização é bastante parecida com a stemização, porém a stemização leva em consideração apenas a palavra em si e não seu contexto.

2.5.2.4 *POS-Tagging*

Part-of-Speech (POS) tagging refere-se ao processo de anotar um texto atribuindo a cada *token* sua função, como adjetivo, verbo, preposição, etc. Trata-se de um processo que é muito utilizado como etapa inicial para reconhecimento de entidades nomeadas (JURAFSKY; MARTIN, 2000).

2.5.3 Vetorização e Extração de *Features*

Vetorização refere-se ao processo de converter um conjunto de dados em um vetor de *features* de maneira que este vetor possa ser utilizado para treinar um modelo de aprendizagem de máquina (JURAFSKY; MARTIN, 2000). Em dados textuais, é comum que as features utilizadas sejam as palavras do vocabulário total contido no conjunto de textos. A seguir são descritos dois dos principais métodos de vetorização de textos para aprendizagem de máquina.

2.5.3.1 *Bag of Words*

A vetorização de textos pelo método *Bag of Words (BoW)* ou cesto de palavras, funciona lista todos os *tokens* contidos no conjunto total de textos sob análise e representa cada texto como um vetor onde cada palavra da lista corresponde a uma dimensão do vetor, e cada valor armazenado no vetor corresponde ao número de ocorrências daquele *token* no documento em questão (JURAFSKY; MARTIN, 2000).

Embora seja fácil de implementar e rápido de executar, uma desvantagem deste método é que cada palavra de um texto é representada sem informações de seu contexto.

2.5.3.2 *Term Frequency-Inverse Document Frequency* (TF-IDF)

Para entender o método TF-IDF, vale a pena analisar o método termo a termo. *Term frequency*, ou frequência de termo (JURAFSKY; MARTIN, 2000), descreve o a frequência relativa de um termo, ou *token*, no documento em questão, ou seja, é o número de ocorrências do termo no documento dividido pelo numero total de termos no documento:

$$tf(t,d) = \frac{count(t)}{n(d)} \tag{2.1}$$

onde t é o termo em questão, d é o documento, count(t) é o número de ocorrências do termo e n(d) é o total de termos no documento d.

Document frequency (DF), ou frequência em documentos, refere-se ao número total de ocorrências de um termo em todo o conjunto de textos em questão:

$$DF = \frac{df(t)}{N} \tag{2.2}$$

onde df(t) é o número de ocorrências do termo t e N é número total de documentos.

IDF então refere-se ao inverso desta quantidade,

$$idf(t) = \frac{N}{df(t)} \tag{2.3}$$

. Costuma-se usar o logaritmo desta quantidade para suavizar o crescimento da função em grandes corpos de textos:

$$idf(t) = log(\frac{N}{df(t)+1}) \tag{2.4}$$

onde o 1 no denominador é acrescentado para evitar divisão por zero.

Assim, o TF-IDF é calculado como

$$tfidf(t,d) = tf(t,d) * log(\frac{N}{df(t)+1}) \tag{2.5}$$

2.5.4 Avaliação de Modelos

De posse dos resultados fornecidos por um modelo treinado em um conjunto de testes e dos rótulos verdadeiros para este conjunto de dados (ZAKI; MEIRA JR, 2020), pode-se obter os seguintes valores:

- **Verdadeiros Positivos(True Positives ou TP):** quantidade de instâncias do resultado em que o modelo prevê corretamente a classe positiva;
- **Falsos Positivos (False Positives ou FP):** quantidade de instâncias do resultado em que o modelo prevê incorretamente a classe positiva;
- **Verdadeiros Negativos (True Negatives ou TN):** quantidade de instâncias do resultado em que o modelo prevê corretamente a classe negativa;
- **Falsos Negativos (False Negatives ou FN):** quantidade de instâncias do resultado em que o modelo prevê incorretamente a classe positiva.

De posse destes valores, é possível obter as seguintes métricas para avaliação do desempenho de um modelo.

- **Acurácia:** indica a fração de previsões, verdadeiras ou falsas, que o modelo realmente acertou. A acurácia (A) pode ser definida como a soma da quantidade de Verdadeiros Positivos (TP) com a quantidade de Verdadeiros Negativos (TN) divida pela soma da quantidade de Verdadeiros Positivos (TP) com a quantidade de Verdadeiros Negativos (TN), Falsos Positivos (FP).

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

- **Precisão:** representa a proporção de identificações positivas que foi realmente correta. A precisão (P) pode ser definida como a quantidade de Verdadeiros Positivos (TP) dividida pela soma da quantidade Verdadeiros Positivos com a quantidade de Falsos Positivos (FP)

$$P = \frac{TP}{TP + FP} \tag{2.7}$$

- **Recall:** representa a proporção de positivos reais foi identificada corretamente, ou seja, de todos os positivos nos dados, qual proporção o modelo conseguiu identificar. O

recall (R) pode ser definido como a quantidade de Verdadeiros Positivos (TP) dividida pela soma da quantidade de Verdadeiros Positivos (TP) com a quantidade de Falsos Negativos (FN)

$$R = \frac{TP}{TP + FN} \tag{2.8}$$

- **F1-Score:** é uma métrica que foi introduzida para compensar pela baixa informatividade das duas métricas anteriores no caso de bases de dados altamente enviesadas. O F1-Score é calculado a partir da média harmônica da precisão e do recall. Um modelo com desempenho perfeito teria um F1-Score igual a 1.

- **Matriz de confusão:** A matriz de confusão é uma tabela que permite a visualização direta do desempenho do sistema. Cada linha da matriz representa instâncias de uma classe prevista enquanto cada coluna representa instâncias da classe atual. A Tabela 1 mostra um exemplo.

	Predição positiva	Predição negativa
Classe positiva	Positivos Verdadeiros (TP)	Negativos Verdadeiros (TN)
Classe Negativa	Positivos Falsos (FP)	Negativos Falsos (FN)

Tabela 1 – Matriz de Confusão Genérica.

No próximo capítulo será apresentada a metodologia utilizada para implementar as diversas técnicas apresentadas para construção do sistema de segmentação de publicações que é o objetivo deste trabalho.

3 Trabalhos Relacionados

Neste capítulo serão apresentados dois projetos que visam realizar a extração dos diários oficiais dos diversos entes da federação, e utilizar as informações extraídas para melhorar a fiscalização do poder público e das empresas que prestam serviços ao mesmo.

Os projetos Querido Diário e Deep-vacuity são dois projetos que visam utilizar técnicas modernas de inteligência artificial e aprendizado de máquinas para facilitar o trabalho de monitoramento do poder público pela população. O principal meio que ambos os projetos utilizam para esse fim é o monitoramento de informações publicadas em diários oficiais, como o Diário Oficial da União (DOU) no caso do projeto Deep-Vacuity e os diários oficiais municipais no caso do projeto Querido Diário. Neste capítulo serão descritos estes dois projetos, assim como seus históricos e o funcionamento de cada um.

3.1 Projeto Querido Diário

O projeto Querido Diário (OKBR, 2022) foi iniciado pela fundação Open Knowledge Brasil[1] em 2018 com o objetivo de criar um sistema que extraísse e indexasse informações dos diários oficiais de todos os municípios do Brasil para que seja possível o monitoramento automatizado dos atos dos poderes em todo o território nacional e assim facilitar a fiscalização do poder público. Atualmente o projeto conta com extratores de texto para cerca de 600 municípios e disponibiliza um website online para buscas nos diários oficiais por palavras-chave, municípios ou datas.

A extração de dados dos diários municipais apresenta desafios principalmente por causa da dificuldade em rastrear as fontes para cada um destes diários e pela heterogeneidade das formas de publicação e disponibilização dos dados. Alguns municípios publicam em sites de associações, empresas privadas ou até mesmo site de outros entes da federação, e tudo isso sem nenhuma gestão de fluxo da informação.

3.1.1 Histórico

O Querido Diário nasceu da inspiração obtida de dois projetos anteriores, que tinham objetivos semelhantes. O primeiro deles, o Projeto Nosso Querido Diário Oficial (QueriDO) foi também uma iniciativa da Open Knowledge Brasil e funcionava com a participação ativa da comunidade. Um grupo de pessoas, chamadas de curadoras, definiam os "alvos", estabelecendo para quais assuntos e diários oficiais desejavam direcionar sua atenção, enquanto um segundo grupo de pessoas, chamadas de experts, codificariam os extratores de

[1] https://ok.org.br/sobre/

dados utilizados para extrair informações dos diários. O coletivo responsável pelo projeto se responsabilizava pela tutela das informações e por manifestar publicamente o seu parecer em relatórios fundamentados nos diários oficiais. O segundo projeto que inspirou a criação do Querido Diário foi a Operação Serenata de Amor (OKBR, 2016), um projeto aberto que utiliza uma inteligência artificial, chamada Rosie[2], para auditar as contas públicas e detectar suspeitas de mal uso do dinheiro público por parlamentares. Esse projeto monitora principalmente os reembolsos efetuados pela Cota para Exercício da Atividade Parlamentar (CEAP), uma verba que custeia alimentação, transporte, hospedagem, entre outras despesas dos parlamentares. O projeto foi posteriormente englobado na estrutura da Open Knowledge Brasil. O projeto Serenata de Amor ganhou grande destaque na mídia, sendo abordado nos maiores jornais do país e gerando uma forte demanda da população por mais iniciativas deste tipo. Como a operação Serenata de Amor focava-se no Congresso Nacional e no âmbito federal, buscou-se criar iniciativas que realizassem algo semelhante nas esferas estadual e municipal. Foi criado então em 2018 o projeto Diário Oficial da Serenata, com o objetivo de levar as análises para a esfera municipal. Este projeto tinha inicialmente o intuito de coletar os arquivos de diários oficiais dos municípios e criar mecanismos para analisar a dispensa de licitações, mas a ideia enfrentou muitos desafios e não seguiu adiante. Posteriormente, quando o Serenata de Amor já estava englobado no âmbito da fundação Open Knowledge Brasil, o projeto Nosso Querido Diário Oficial foi renomeado Querido Diário, e os esforços do projeto foram redirecionados para escrever o maior número possível de raspadores para os diários municipais. Em menos de dois anos, o projeto passou a contar com uma cobertura de mais de 2200 municípios e cerca de 600 raspadores ativos, e a plataforma oficial do Querido Diário foi lançada em julho de 2021.

3.1.2 Funcionamento do Sistema

O escopo atual do projeto está em coletar os atos do poder executivo dos 5.570 municípios brasileiros. O sistema realiza a extração diária dos arquivos publicados naqueles diários para os quais possui extratores e, após tentar separar metadados e textos dos diários, então indexa as publicações dos diários em um banco de dados. Porém os resultados retornados pelo sistema ainda não são satisfatórios, pois devido à grande heterogeneidade dos dados e ao fato de que a maioria dos arquivos extraídos encontram-se em formato de arquivo portável (*Portable Document Format* ou PDF), de difícil tratamento automatizado, as publicações individuais contidas nos diários ainda não estão claramente segmentadas, o que dificulta sua correta indexação e posteriores buscas.

A seguir são descritas as etapas realizadas pelo sistema do Querido Diário desde a localização das fontes de dados e criação de extratores até a incorporação dos mesmos no sistema e indexação dos diários extraídos para busca.

[2] https://medium.com/serenata/como-a-rosie-usa-machine-learning-na-serenata-de-amor-9168e0f1909d

3.1.2.1 Criação de Extratores de Dados

O primeiro passo é a reunião de informações a respeito das páginas da Internet onde estão disponibilizados os arquivos ou textos dos diários oficiais dos municípios. A coleta destas informações conta com o auxílio de voluntários que podem cadastrar as fontes para os diários de um determinado município através do portal do Censo do Querido Diário. [3] Para facilitar o controle do andamento do projeto e monitorar o progresso da criação dos extratores de dados, a equipe do projeto Querido Diário classificou cada município de acordo com os seguintes níveis de acesso:

0 - O projeto não possui a fonte de publicação do diário oficial deste município 1 - O projeto possui a fonte de publicação do diário oficial deste município 2 - O projeto possui o script para coletar os arquivos e armazená-los em nossa base 3 - O conteúdo dos diários oficiais deste município está disponível na plataforma Querido Diário

De posse das páginas que contém as informações dos diários municipais, a equipe do Querido Diário desenvolve os extratores de dados, utilizando a ferramenta Scrapy. Este desenvolvimento também é fundamentado na participação de programadores voluntários ligados à fundação Open Knowledge Brasil, que contribuem seu tempo e experiência para a desenvolvimento das iniciativas propostas pela fundação. Os extratores são criados utilizando a linguagem Python por meio do framework Scrapy, que será apresentado com mais detalhes no próximo capítulo.

Uma vez em posse dos extratores de dados, estes são incluídos no repositório do projeto no GitHub e na implementação do sistema e os diários passam a ser indexados na base de dados do projeto.

3.1.2.2 Arquitetura do Sistema

O sistema é organizado como mostrado na Figura 1.

O projeto é fundamentado em três bases principais: Os algoritmos de extração e indexação dos dados dos diários oficiais, uma plataforma online para visualização das informações extraídas e uma API aberta para integração com outros sistemas e aplicações.

Já foi discutido acima o processo de criação dos extratores. Uma vez que estes extratores para um município estejam prontos, os arquivos brutos são extraídos juntamente com metadados referentes a cada diário. O sistema também tenta extrair e indexar o texto dos arquivos brutos e indexá-lo juntamente com os metadados, utilizando a ferramenta elasticsearch.[4] Os dados indexados são disponibilizados para consultas públicas através da página do Querido Diário, com uma API própria.

[3] Disponível em https://censo.ok.org.br/sobre/
[4] Disponível em https://www.elastic.co/elasticsearch/

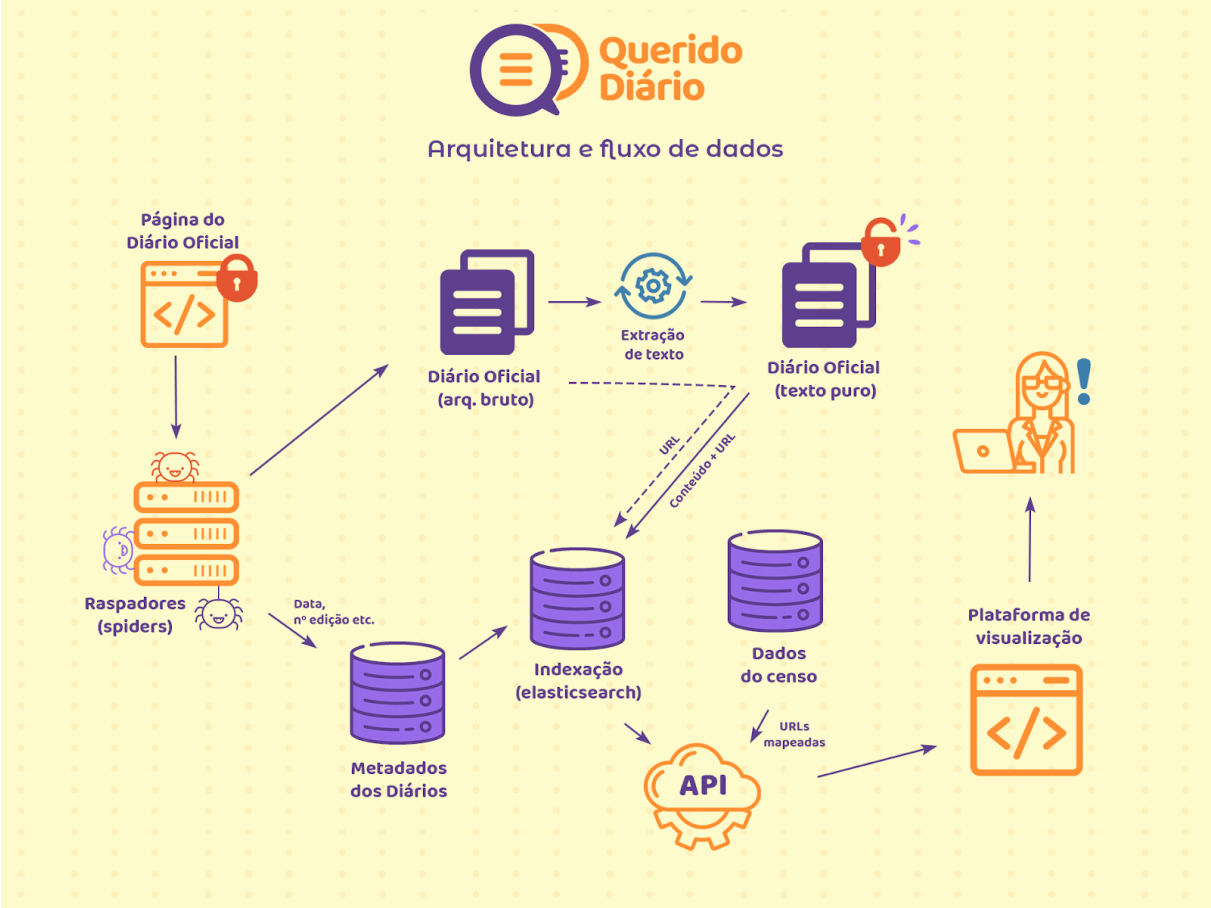


Figura 4 – Arquitetura do projeto Querido Diário.

Fonte: https://queridodiario.ok.org.br/tecnologia

3.2 Projeto Deep-Vacuity

As práticas de conluio e formação de cartel em obras públicas são um fato muito comum no Brasil. Essas práticas se dão principalmente quando grandes empreiteiras, que dominam esse mercado, combinam entre si como vão dividir as obras e as licitações. Essas práticas facilitam a ocorrência de superfaturamento em obras e contribuem para a continuidade da corrupção no país.

O projeto Deep-Vacuity (LIMA, 2021), atualmente desenvolvido de maneira conjunta por equipes da Universidade de Brasília e do Instituto Nacional de Criminalística (INC) da polícia Federal, busca utilizar técnicas de aprendizado de máquina e inteligência artificial para identificar comportamento de cartéis de empresas em obras públicas. O projeto é semelhante ao projeto Querido Diário no sentido de que utiliza principalmente documentos publicados nos diversos diários oficiais do país.

3.2.1 Histórico e Motivação

O projeto Deep-Vacuity surgiu em um contexto em que cada vez mais se busca utilizar a tecnologia para monitorar os poderes públicos e reduzir a incidência da corrupção. Casos notórios de corrupção, como o revelado pela Operação Lava-Jato, mostraram que os meios utilizados pelos criminosos estão se tornando cada vez mais sofisticados e fizeram aumentar a demanda da população por maneiras mais modernas de fiscalizar o poder público e diversas iniciativas surgiram nos últimos anos, tais como:

- **Operação Serenata de Amor**[5]

  já mencionada anteriormente, a Operação Serenata de Amor utiliza uma inteligência artificial, chamada Rosie, para auditar as contas públicas e detectar suspeitas de mal uso do dinheiro público. Inicialmente focado em despesas dos deputados da Câmara dos Deputados, o projeto foi desde então expandido para incluir gastos de senadores.

- **Operação Política Supervisionada**[6]

  Monitora os gastos realizados via Cota para Exercício da Atividade Parlamentar (CEAP). Também focado no monitoramento da Câmara dos Deputados e do Senado Federal, a fiscalização dos gastos é realizada por uma rede de voluntários espalhados pelo Brasil e qualquer pessoa pode participar do processo.

- **Observatório da Despesa Pública**[7]

  Criado em 2008 por iniciativa da Controladoria-Geral da União (CGU), tem o objetivo de acompanhar os gastos governamentais e aprimorar o controle interno e externo. Inicialmente voltado à esfera federal, o projeto foi recentemente expandido para englobar uma rede de observatórios em todos os estados do Brasil.

- **Sistema de Análise de Licitações e Editais (Alice)**[8]

  Desenvolvido pelo TCU, o robô analisa diariamente o Diário Oficial da União e todos os editais e atas inseridos no Comprasnet em busca de inconsistências nos editais de licitação e nos resultados de atas de pregão eletrônico publicados, diariamente, no Portal de Compras do Governo Federal.

Dentro deste contexto, o projeto Deep-Vacuity busca ampliar as capacidades de monitoramento do poder público pelos agentes da Polícia Federal e agilizar a detecção e investigação de comportamentos suspeitos por empresas e membros do poder público.

[5] https://serenata.ai/about/
[6] https://www.ops.net.br/sobre
[7] https://tinyurl.com/senadoobsdespesas
[8] https://tinyurl.com/alicecgu

3.2.2 Funcionamento do Sistema

O sistema do projeto Deep-Vacuity está organizado de forma a seguir um fluxo de trabalho que envolve a captura, o pré-processamento, a formatação, o processamento efetivo e a análise dos dados dos diários.

3.2.2.1 Arquitetura

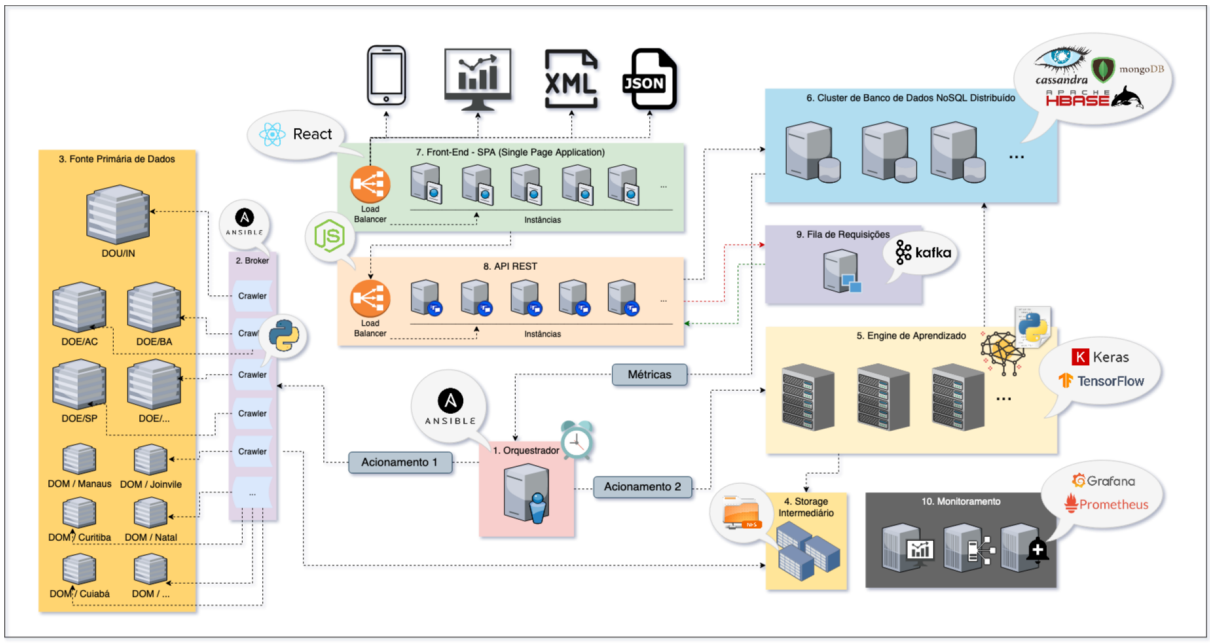


Figura 5 – Arquitetura do projeto Deep-Vacuity.

Fonte: https://repositorio.unb.br/handle/10482/42026

A captura ou extração dos dados é feita com métodos de web scraping para a obtenção de dados brutos em formatos como PDF, HTML ou XML. Os dados são então extraídos para um padrão textual puro. Os dados brutos, dados processados e metadados são então formatados em padrões semi-estruturados como CSV ou JSON ou inseridos em bancos de dados. A partir destes formatos semi-estruturados ou estruturados, os dados podem ser então analisados com técnicas como Extração de Entidades Nomeadas ou outras técnicas de processamento de linguagem natural. Estes dados podem também ser anotados para treinamento de sistemas supervisionados de aprendizado de máquina para aplicações específicas.

4 Metodologia

Neste capítulo são descritos os métodos utilizados para criação do sistema de segmentação de publicações. Primeiro é explicado como foi feita a extração dos arquivos dos diários oficiais de seus respectivos domínios na Internet, por meio de técnicas de *web scraping*. A seguir é descrita a extração dos textos desses diários por meio de ferramentas de refluxo de textos e reconhecimento óptico de caracteres. Por fim é descrito o processo de anotação de uma base de dados para treinamento do sistema e o processo de treinamento em si e avaliação de resultados. A figura 6 fornece uma visão geral do fluxo de trabalho que foi seguido.

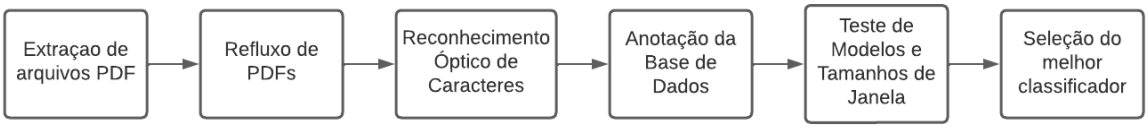


Figura 6 – Fluxograma das etapas seguidas durante o trabalho.

Fonte: Autor

4.1 Extração dos Arquivos dos Diários Oficiais Estaduais e Municipais

A primeira etapa do projeto envolveu a extração dos arquivos dos diários oficiais municipais e estaduais dos domínios onde estão localizados na Internet. Para isso foi utilizado o código do projeto Querido Diário[1] (Capítulo 3). O projeto disponibiliza extratores de dados para cerca de 600 municípios brasileiros, implementados por meio da ferramenta Scrapy (Capítulo 2).

A interface do Querido Diário permite passar o parâmetro da cidade de onde se deseja extrair diários e das datas desejadas por meio de uma opção na linha de comando, como no exemplo a seguir:

```
scrapy crawl sc\_florianopolis -a start\_date=2020-09-01
```

Para se gerar uma amostra representativa de diários municipais, primeiro foi gerada uma lista textual com todos os *crawlers* disponíveis no Querido Diário. Em seguida, utilizando a linguagem Python e suas funções para números aleatórios, foi selecionado um subconjunto

[1] Disponível em https://github.com/okfn-brasil/querido-diario

desses crawlers para serem executados. Procedeu-se então à rodar o código do Querido Diário em *loop* de acordo com esta lista aleatória de *crawlers* para captar diários oficiais das cidades selecionadas. Para que a extração de dados não se prolongasse excessivamente, o escopo de datas dos diários extraídos foi limitado para o ano de 2022.

O Querido Diário ainda não possui extratores para diários estaduais. Por isso foi feita também uma busca e seleção manual de diários oficiais estaduais. Buscou-se encontrar diários que não fossem muito extensos para facilitar o trabalho de anotação dos dados e também buscando abranger todos os estados do Brasil.

De posse dos arquivos PDF extraídos dos sites oficiais, procedeu-se a inspecionar estes arquivos para selecionar uma amostra representativa da maioria dos estados brasileiros, montando uma base de dados com os diários que apresentassem dados claros e que não fossem muito extensos, para facilitar o trabalho de anotação. Ao todo, foram selecionados cerca de 250 diários estaduais e municipais, cobrindo todos os estados do páis.

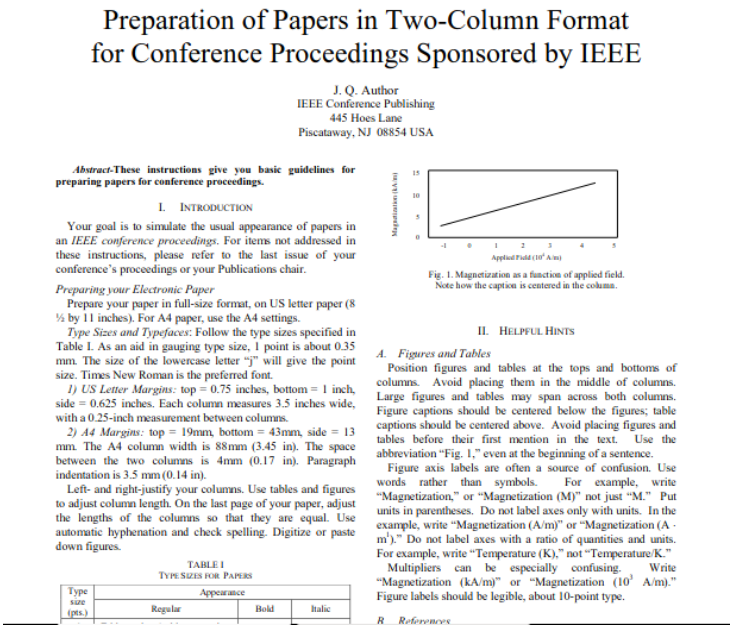
4.2 Extração de Texto dos Arquivos PDF

Como mencionado no Capítulo 2, as tentativas de extração direta do texto dos arquivos PDF não tiveram resultados satisfatórios, principalmente para textos de duas ou mais colunas. Por isso, optou-se por utilizar uma ferramenta de refluxo de texto para a transformação do texto de duas colunas em uma única coluna de texto. Adicionalmente, como a ferramenta utilizada (única ferramenta gratuita encontrada para esta tarefa) só fornecia a saída como arquivos PDF compostos de imagens, foi necessário utilizar uma ferramenta de reconhecimento óptico de caracteres para a extração final do texto.

A ferramenta utilizada para o refluxo de textos foi a biblioteca k2pdfopt[2]. Esta ferramenta foi originalmente projetada para refluxo de arquivos PDF para facilitar sua visualização em dispositivos *e-Readers*. Ela apresenta uma vasta gama de opções de processamento, incluindo opções para detecção do número de colunas no texto, ajuste da resolução das imagens de saída, entre outras. As Figuras 7 e 8 apresentam um exemplo do refluxo de um arquivo PDF de duas colunas para um de uma coluna utilizando esta ferramenta.

A ferramenta escolhida para o reconhecimento óptico de caracteres foi a biblioteca *Tesseract*, por ser gratuita e apresentar um bom desempenho para a maioria dos cenários de OCR. Para confirmar a adequação do *Tesseract* à tarefa, foi feito um *benchmark*, no qual selecionou-se um conjunto adicional de 20 diários de uma coluna, cujo texto pudesse ser extraído facilmente por extratores de texto convencionais. Procedeu-se então em extrair os mesmos textos utilizando o *Tesseract*, e comparou-se a similaridade dos textos extraídos por dois métodos: primeiro pela matriz de correlação dos vetores dos textos pelo método TF-IDF

[2] https://www.willus.com/k2pdfopt/



Preparation of Papers in Two-Column Format
for Conference Proceedings Sponsored by IEEE

J. Q. Author
IEEE Conference Publishing
445 Hoes Lane
Piscataway, NJ 08854 USA

***Abstract*-These instructions give you basic guidelines for preparing papers for conference proceedings.**

I. Introduction

Your goal is to simulate the usual appearance of papers in an *IEEE conference proceedings*. For items not addressed in these instructions, please refer to the last issue of your conference's proceedings or your Publications chair.

Preparing your Electronic Paper

Prepare your paper in full-size format, on US letter paper (8 ½ by 11 inches). For A4 paper, use the A4 settings.

Type Sizes and Typefaces: Follow the type sizes specified in Table 1. As an aid in gauging type size, 1 point is about 0.35 mm. The size of the lowercase letter "j" will give the point size. Times New Roman is the preferred font.

1) US Letter Margins: top = 0.75 inches, bottom = 1 inch, side = 0.625 inches. Each column measures 3.5 inches wide, with a 0.25-inch measurement between columns.

2) A4 Margins: top = 19mm, bottom = 43mm, side = 13 mm. The A4 column width is 88mm (3.45 in). The space between the two columns is 4mm (0.17 in). Paragraph indentation is 3.5 mm (0.14 in).

Left- and right-justify your columns. Use tables and figures to adjust column length. On the last page of your paper, adjust the lengths of the columns so that they are equal. Use automatic hyphenation and check spelling. Digitize or paste down figures.

TABLE I
Type Sizes for Papers

Type size (pts.)	Appearance		
	Regular	Bold	Italic

Fig. 1. Magnetization as a function of applied field.
Note how the caption is centered in the column.

II. Helpful Hints

A. Figures and Tables

Position figures and tables at the tops and bottoms of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be centered below the figures; table captions should be centered above. Avoid placing figures and tables before their first mention in the text. Use the abbreviation "Fig. 1," even at the beginning of a sentence.

Figure axis labels are often a source of confusion. Use words rather than symbols. For example, write "Magnetization," or "Magnetization (M)" not just "M." Put units in parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization (A · m^{1})." Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)," not "Temperature/K."

Multipliers can be especially confusing. Write "Magnetization (kA/m)" or "Magnetization (10^{3} A/m)." Figure labels should be legible, about 10-point type.

B. References

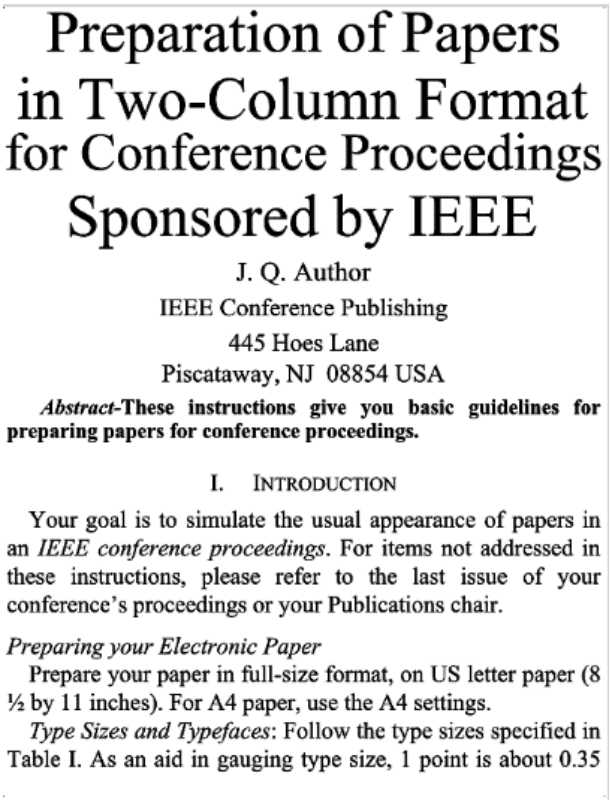
Figura 7 – Exemplo de texto de duas colunas em PDF.

Fonte: https://www.willus.com/k2pdfopt/

e segundo utilizando o método de cálculo de similaridade de documentos da biblioteca *Spacy*. Os resultados são apresentados no Capítulo 5.

4.3 Anotação da Base de Dados

A segmentação das publicações dos diários foi modelada como um problema de classificação binária de linhas de texto, sendo que a primeira linha que marca o início de uma nova publicação recebe a classificação 1, e todas as demais linhas a classificação 0. Procedeu-se da seguinte forma para a anotação dos textos para treinamento: o primeiro passo foi gerar, para cada arquivo de texto extraído dos PDFs, um segundo arquivo de texto, que conteria os *targets* para classificação, contendo o mesmo número de linhas, cada uma preenchida com o número 0. A partir daí foi feita a anotação manual, com auxílio de um editor de texto simples, de todas as linhas no texto que correspondiam ao início de uma nova publicação, mudando-se a linha correspondente no arquivo de *targets* de 0 para 1. Esta foi a etapa mais demorada e trabalhosa do projeto, pois toda a anotação foi feita à mão. O processo de anotação levou cerca de 2 meses para ser concluído e a base de dados final contava com 191 diários anotados.



Preparation of Papers
in Two-Column Format
for Conference Proceedings
Sponsored by IEEE

J. Q. Author
IEEE Conference Publishing
445 Hoes Lane
Piscataway, NJ 08854 USA

***Abstract*-These instructions give you basic guidelines for preparing papers for conference proceedings.**

I. INTRODUCTION

Your goal is to simulate the usual appearance of papers in an *IEEE conference proceedings*. For items not addressed in these instructions, please refer to the last issue of your conference's proceedings or your Publications chair.

Preparing your Electronic Paper

Prepare your paper in full-size format, on US letter paper (8 ½ by 11 inches). For A4 paper, use the A4 settings.

Type Sizes and Typefaces: Follow the type sizes specified in Table I. As an aid in gauging type size, 1 point is about 0.35

Figura 8 – Refluxo de texto da imagem anterior.

Fonte: https://www.willus.com/k2pdfopt/

4.4 Vetorização de Textos e Treinamento de Classificadores

A vetorização dos textos e o treinamento de classificadores foram realizados com ferramentas da biblioteca *Scikit-Learn*. Para a vetorização foram testados dois vetorizadores: *Bag-of-Words*, correspondente ao vetorizador *CountVectorizer* do *Scikit-Learn*, e TF-IDF, implementado por meio do vetorizador *tfidfvectorizer*.

Diferente de uma aplicação de classificação de documentos inteiros, neste trabalho o que se está classificando são as linhas de texto extraídas de cada diário. Desta forma, para a aplicação dos vetorizadores, as linhas de texto dos diários e as linhas correspondentes nos arquivos de *targets* foram primeiro lidas para duas listas. Estas listas de linhas e *targets* foram divididas em conjuntos de treinamento e teste e em seguida os algoritmos de vetorização foram aplicados nos elementos da lista de linhas de texto de teste. O resultado desta vetorização é uma matriz de vetores cuja dimensão corresponde ao tamanho do vocabulário presente no conjunto de treinamento.

Com os conjuntos de treinamento e teste devidamente vetorizados, é possível utilizar os modelos do *Scikit-Learn* para treinamento e classificação, em que cada modelo é treinado no conjunto de treinamento por meio de seu método *fit*, e resultados de classificação são

obtidos usando o conjunto de teste. De posse das predições do modelo para o conjunto de teste e dos *targets* reais de cada linha, é possível obter as métricas de avaliação para aquele modelo.

O procedimento descrito nos parágrafos anteriores realizava o treinamento de cada modelo nos vetores obtidos apenas a partir de cada linha de texto individual, de modo que os classificadores não têm informações do contexto de cada linha, ou seja, das linhas que vem antes ou depois da linha em questão. Para adicionar informações contextuais no sistema de classificação, a lista de linhas de texto foi expandida de modo a ser transformada em três listas: *before*, *line*, *after*, onde *before* corresponde a uma lista de strings contendo um determinado número de linhas antes da linha em questão e *after* corresponde ao mesmo número de linhas após a linha. A quantidade de linhas a ser incorporada, ou o tamanho da janela, foi controlada em código por um parâmetro, "*WINDOW_SIZE*", e este parâmetro foi variado entre 0, 5 e 10 linhas para determinar um tamanho de janela que fornecesse melhores resultados.

5 Resultados

Neste capítulo são apresentados os resultados das etapas descritas no capítulo anterior. Primeiramente são apresentados os resultados da extração de diários, seguidos pelo refluxo de textos, reconhecimento óptico de caracteres e a criação da base de dados anotada. Por fim, são apresentados os resultados do treinamento e avaliação de diversos modelos nos dados anotados para seleção do melhor modelo a ser utilizado.

5.1 Extração de Dados

Utilizando os extratores de texto do projeto Querido Diário, foi possível fazer o *download* de cerca de 250 diários de diversos municípios do Brasil, englobando todos os estados do país. A seleção destes municípios foi feita manualmente gerando-se uma lista de municípios a partir da lista de extratores disponíveis no projeto Querido Diário, buscando-se englobar cerca de 4 municípios de cada estado, quando possível. Além disso, cerca de 50 diários estaduais foram selecionados manualmente buscando englobar cerca de 2 diários por estado. Após seleção manual de diários que possuíam uma extensão apropriada para anotação manual, um total de 200 diários foram selecionados para inclusão na base de dados.

5.2 Extração de Texto

As Figuras 9 e 10 mostram um exemplo de uma página de um dos diários de duas colunas da base de dados, e a mesma página após o refluxo de texto com a ferramenta k2pdfopt. A extração de texto foi realizada com a biblioteca *Tesseract OCR*. A Figura 11 mostra o texto extraído do documento da Figura 10. Como pode-se observar, a extração de texto por meio de OCR foi bastante precisa.

Para testar a confiabilidade dos resultados obtidos pela biblioteca *Tesseract*, foi feito um *benchmarking* da ferramenta, como descrito no capítulo 4, comparando-se o resultado da extração de 20 textos com extratores e com a biblioteca. A similaridade entre cada par de textos iguais foi calculada pelos métodos da matriz de coocorrências e utilizando-se uma função da biblioteca *Spacy*. A similaridade média entre os textos foi então calculada para cada método. As similaridades entre os textos foram de 97,26% para o primeiro método e 92,99% para o segundo. A principal fonte de erros na extração dos textos pelo método de ocorreu na detecção de tabelas presentes nos documentos.

SIGPUB - Sistema Gerenciador de Publicações Legais

DIÁRIO OFICIAL
DOS MUNICÍPIOS DO ESTADO DO PARÁ

FAMEP

Pará , 03 de Janeiro de 2022 • Diário Oficial dos Municípios do Estado do Pará • ANO XIII | Nº 2900

Expediente:
Federação das Associações de Municípios do Estado do Pará - FAMEP

CONSELHO DIRETOR 2017/2020

PRESIDENTE LICENCIADO: Francisco Nélio Aguiar da Silva – Prefeito do Município de Santarém;

1° VICE – PRESIDENTE E PRESIDENTE EM EXERCÍCIO: Wagne Costa Machado – Representante Legal do Município de Piçarra;

2° VICE – PRESIDENTE: José Antônio de Azevedo Leão - Prefeito do Município de Breves.

SECRETÁRIO EXECUTIVO: Josenir Gonçalves Nascimento

01-AMAM – Carlos Augusto de Lima Gouvêa (Presidente) – Prefeito de Soure
02-AMATCARAJÁS – Jair Lopes Martins (Presidente) – Prefeito de Conceição do Araguaia
03-AMUNEP – Egilásio Alves Feitosa – (Presidente) Prefeito de Inhangapí
04-AMUCAN – Odair José Farias Albuquerque - Respondendo Interinamente (Prefeito de Terra Santa)
05-AMUT – Rosibergue Torres Campos (Presidente) – Prefeito de Porto de Moz
06-COIMP – Marcos Cesar Barbosa e Silva (Presidente) - Prefeito de São Francisco do Pará
07-CODESEI – Claudio Iago (Representante) – Secretário Executivo
08-COMPART – Flavio Marcos Mezzomo (Presidente) – Prefeito de Breu Branco
09-AMCBM (BELO MONTE) – Leila Raquel Possimoser –(Presidente) Prefeita de Placas
10- COPSAL – Jefferson Ferreira de Miranda (Presidente) – Prefeito de Curuçá
11- CISAT – Maria da Graça Medeiros Matos (Presidente) - Prefeita de Nova Ipixuna
12 – CONSÓRCIO TAPAJÓS – Vilson Gonçalves (Presidente) - Prefeito de Aveiro

O Diário Oficial dos Municípios do Estado do Pará é uma solução voltada à modernização e transparência da gestão municipal.

ESTADO DO PARÁ

Publique-se
Cumpra-se

Gabinete do Presidente da Câmara Municipal de Canaã dos Carajás-PA, ao dia 30/12/2021

DINILSON JOSÉ DOS SANTOS
Presidente da Câmara Municipal de Canaã dos Carajás - PA
Biênio 2021/2022

Publicado por:
Rosilene Monteiro Oliveira
Código Identificador:2768EFD4

CÂMARA MUNICIPAL DE CANAÃ DOS CARAJÁS
PORTARIA Nº 265/2021

O Presidente do Poder Legislativo de Canaã dos Carajás
No uso de suas atribuições legais.

RESOLVE:

Art. 1° - EXONERAR a partir 31 de dezembro de 2021, todos os cargos comissionados nomeados pela legislatura do ano de 2021.

Art. 2° - As novas portarias de nomeações serão feitas individualmente para a administração da casa; bem como os assessores parlamentares através de memorandos emitidos pelos devidos gabinetes.

Art. 2°- Esta portaria entra em vigor nesta data, revogando as disposições em contrário.

Registre-se
Publique-se
Cumpra-se

Gabinete do Presidente da Câmara Municipal de Canaã dos Carajás-PA, ao dia 30/12/2021

DINILSON JOSÉ DOS SANTOS
Presidente da Câmara Municipal de Canaã dos Carajás - PA
Biênio 2021/2022

Publicado por:
Rosilene Monteiro Oliveira
Código Identificador:C46E1166

Figura 9 – Página de um diário oficial.

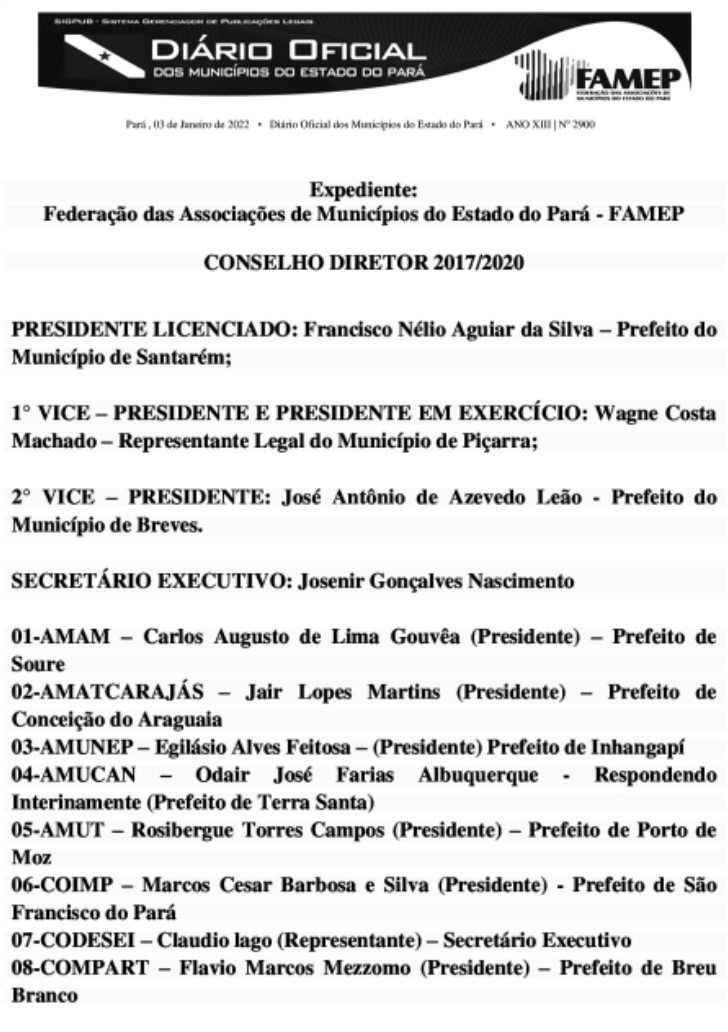
Fonte: https://www.diariomunicipal.com.br/famep/

5.3 Criação da Base de Dados

As figuras 12 e 13 apresentam um exemplo de um trecho de diário contendo uma separação entre publicações e o arquivo de *targets* correspondente. Os números à esquerda correspondem às mesmas linhas nos dois arquivos. As linhas 25 e 42 correspondem a transições entre publicações e recebem o rótulo "1". Este procedimento foi repetido manualmente para cada diário da base. Ao final do processo de anotação, um total de 191 diários com 191 arquivos de *targets* compunham a base de dados.

5.4 Treinamento do Sistema para Segmentação de Publicações

Após a anotação da base de dados, foi criado um código em linguagem Python que ingeria as linhas de texto de todos os documentos da base em conjunto com os respectivos rótulos, ou *targets*. Após divisão das linhas de texto em conjuntos de treinamento e teste, os dados foram vetorizados utilizando um de dois métodos, *Bag of Words* ou TF-IDF, e para



DIÁRIO OFICIAL
DOS MUNICÍPIOS DO ESTADO DO PARÁ
FAMEP

Pará , 03 de Janeiro de 2022 • Diário Oficial dos Municípios do Estado do Pará • ANO XIII | Nº 2900

Expediente:
Federação das Associações de Municípios do Estado do Pará - FAMEP

CONSELHO DIRETOR 2017/2020

PRESIDENTE LICENCIADO: Francisco Nélio Aguiar da Silva – Prefeito do Município de Santarém;

1° VICE – PRESIDENTE E PRESIDENTE EM EXERCÍCIO: Wagne Costa Machado – Representante Legal do Município de Piçarra;

2° VICE – PRESIDENTE: José Antônio de Azevedo Leão - Prefeito do Município de Breves.

SECRETÁRIO EXECUTIVO: Josenir Gonçalves Nascimento

01-AMAM – Carlos Augusto de Lima Gouvêa (Presidente) – Prefeito de Soure
02-AMATCARAJÁS – Jair Lopes Martins (Presidente) – Prefeito de Conceição do Araguaia
03-AMUNEP – Egilásio Alves Feitosa – (Presidente) Prefeito de Inhangapí
04-AMUCAN – Odair José Farias Albuquerque - Respondendo Interinamente (Prefeito de Terra Santa)
05-AMUT – Rosibergue Torres Campos (Presidente) – Prefeito de Porto de Moz
06-COIMP – Marcos Cesar Barbosa e Silva (Presidente) - Prefeito de São Francisco do Pará
07-CODESEI – Claudio lago (Representante) – Secretário Executivo
08-COMPART – Flavio Marcos Mezzomo (Presidente) – Prefeito de Breu Branco

Figura 10 – A mesma página da imagem anterior após refluxo de texto.

Fonte: Autor

cada vetorização, 11 diferentes classificadores da biblioteca *Scikit-Learn* foram treinados nos dados e seus resultados salvos para seleção do melhor modelo para ser utilizado na aplicação final.

Para capturar informações do contexto de cada linha de texto e tentar assim melhorar o desempenho da classificação, os dados foram aumentados com uma janela de linhas de texto: para cada linha, um determinado número de linhas antes e depois da linha em questão foi acrescentado à lista de linhas, e o procedimento de vetorização e treinamento descrito anteriormente foi repetido para cada tamanho de janela. Este método aumenta a dimensionalidade dos dados, mas fornece informações sobre o contexto imediato de cada linha de texto e assim auxilia da discriminação de linhas que dividem duas publicações.

Os resultados são apresentados a seguir. As Figuras 14 e 15 representam o desempenho relativo de cada classificador para os métodos de vetorização *Bag-of-Words* e TF-IDF, respectivamente, com um tamanho de janela igual a 5. As barras de cada gráfico representam

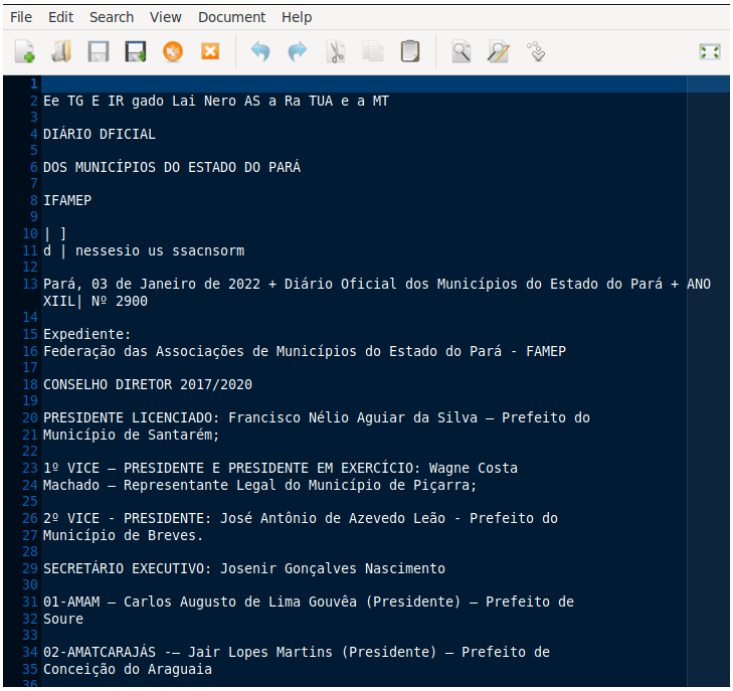


Figura 11 – Texto extraído com a biblioteca *Tesseract*.

Fonte: Autor

o tempo de treinamento, tempo de testes e score de cada modelo. Observa-se que os tempos de treinamento e teste foram menores para o esquema de vetorização TF-IDF do que para o esquema *Bag-of-Words*. Isso provavelmente se deve ao fato de que o esquema TF-IDF é mais informativo do tocante à importância relativa de cada *token* para o documento. De forma geral, os mesmos desempenhos relativos foram observados para outros tamanhos de janela.

A Figura 16 apresenta uma comparação visual dos melhores desempenhos obtidos dentre os classificadores testados, em funçao do tamanho de janela, WINDOW_SIZE. A métrica utilizada no gráfico foi o F1 Score mais elevado para a classe 1, para cada tamanho de janela. Os resultados detalhados dos testes são apresentados no apêndice, onde tabelas contendo precisão, recall, F1-Score e suporte para cada classificador em cada configuração de método de vetorização e tamanho de janela são descritos, assim como os resultados de *F1-Score* de uma etapa de validação cruzada realizada para certificação de que a divisão dos dados não estava demasiadamente enviesada. A validação cruzada foi realizada com uma divisão do conjunto de treinamento em 5 conjuntos para cada rodada de validação.

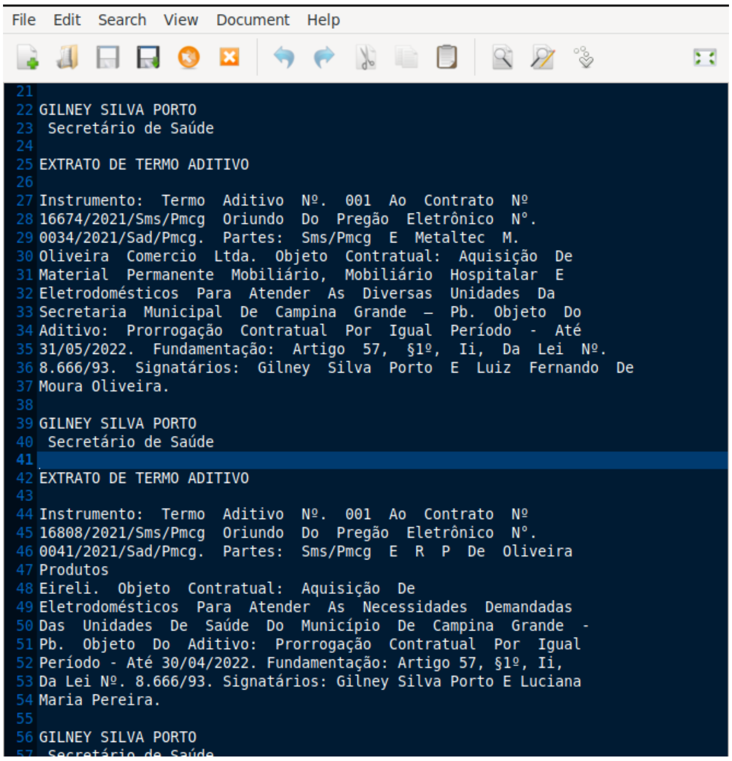


Figura 12 – Parte de um diário onde há uma transição de uma publicação para outra.

Fonte: Autor

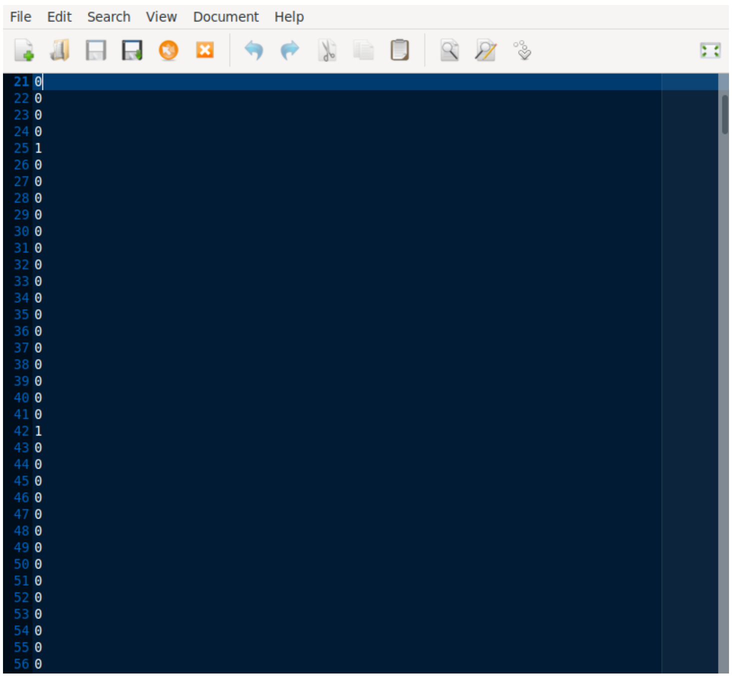


Figura 13 – Anotação do documento da imagem anterior.

Fonte: Autor

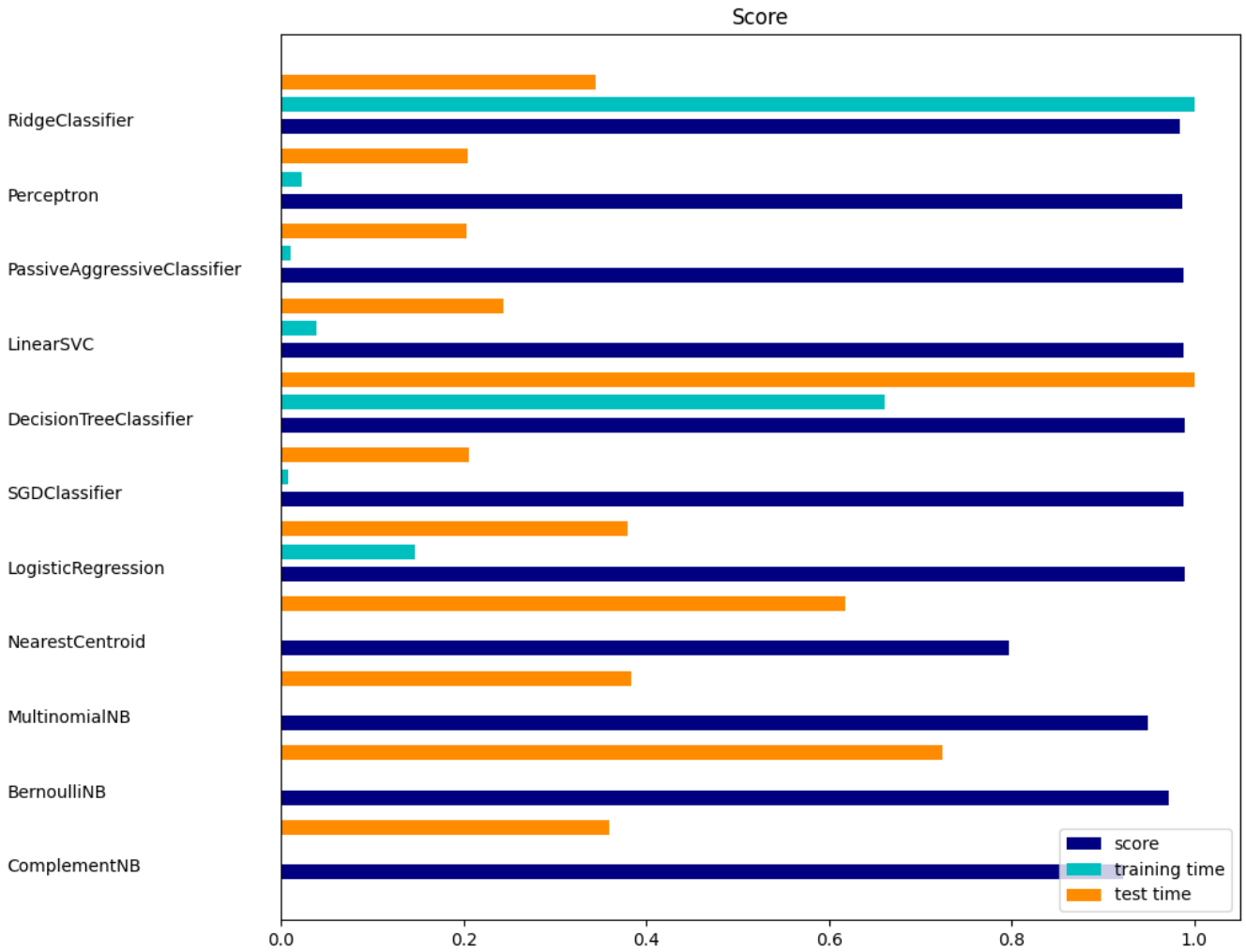


Figura 14 – Desempenho relativo dos classificadores testados com janela de tamanho igual a 5 e vetorização usando *Bag of Words*.

Fonte: Autor

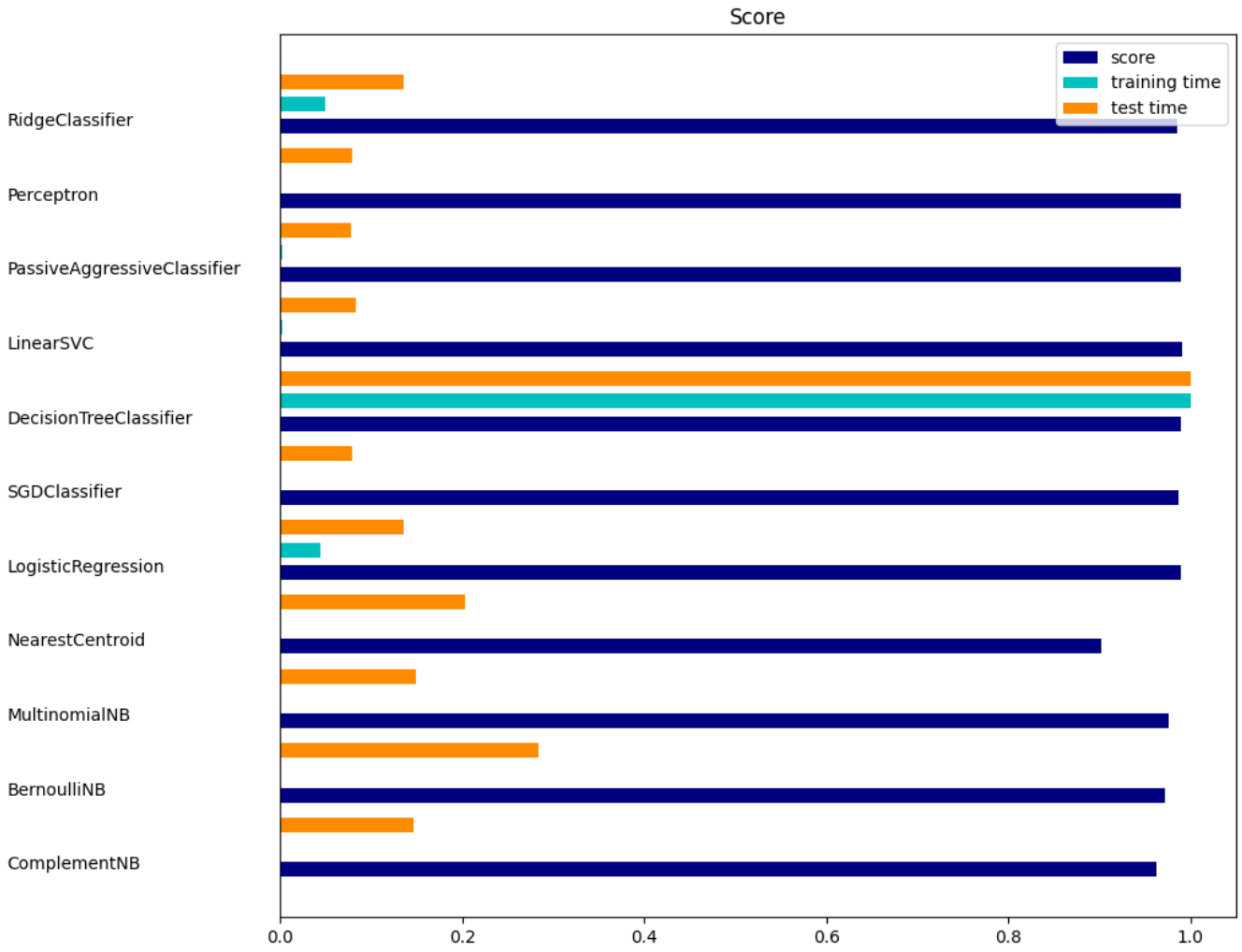


Figura 15 – Desempenho relativo dos classificadores testados com janela de tamanho igual a 5 e vetorização usando TF-IDF.

Fonte: Autor

Olhando primeiro para as Figuras 14 e 15, pode-se observar que os tempos de treinamento e teste dos modelos tenderam a ser maiores para o esquema de vetorização *Bag-of-Words* do que para o esquema TF-IDF. Por brevidade são apresentadas apenas os desempenhos relativos para uma janela de 5 linhas, mas os tempos de treinamento e testes também tenderam a subir quando foi aumentado o valor de WINDOW_SIZE, o que é esperado, uma vez que se está aumentando a dimensionalidade dos dados. Pode-se observar também que todos os modelos apresentam scores elevados, o que se deve ao fato de que a grande maioria das linhas de texto recebem o rótulo 0, o que eleva o score.

A figura 16 apresenta resumidamente os melhores *F1 Score* obtidos entre os classificadores em função do tamanho de janela de texto utilizado. Observa-se claramente um aumento de desempenho com aumento do tamanho de janela até um tamanho 3, e uma queda de desempenho a partir daí. Isso provavelmente se dese ao aumento da ambiguidade presente nos dados passados aos classificadores, quando aumenta-se o tamanho da janela de texto. Isso é esperado, uma vez que se está aumentando a dimensionalidade dos dados ao aumentar a janela, porém não se esperava que o pico de desempenho fosse atingido tão rápido.

Foram obtidos resultados para janelas variando de tamanho entre 0 e 17, porém o desempenho do sistema tendeu a diminuir quando o tamanho de janela foi aumentado acima de 7. Por brevidade, apenas os resultados para janela de tamanho zero e tamanhos ímpares entre 3 e 11 são apresentados.

As tabelas 2 a 13, apresentadas no apêndice, dão informações mais detalhadas no tocante à avaliação de desempenho do sistema. Os melhores *F1 Score* estão destacados nas tabelas, juntamente com o respectivo classificador. Pode-se observar que, para a classe 0 (correspondendo a uma linha que não é uma divisão entre uma publicação e outra) todos os classificadores apresentam elevada precisão, o que explica os scores elevados das figuras. Desta forma, as métricas de desempenho para a classe 1 são mais interessantes para a avaliação do desempenho.

Pode-se observar também das tabelas que todos os classificadores que implementam alguma versão do algoritmo Naive Bayes (NB), assim como o modelo NearestCentroid, apresentaram desempenho muito pobre para o problema em questão. Pode-se notar também que a maioria dos modelos, para um dado tamanho de janela, apresentam leves melhorias de desempenho quando o método de vetorização TF-IDF é utilizado, em comparação com o método Bag of Words. Os modelos que apresentaram o melhor desempenho de maneira mais consistente foram os modelos de árvore de decisão e LinearSVC. Os modelos Passive-Aggressive e Perceptron também apresentaram bons resultados quando o tamanho de janela foi aumentado.

O modelo que apresentou o melhor resultado geral foi o classificador LinearSVC com vetorização TF-IDF e WINDOW_SIZE igual a 3, com um f1-score para a classe 1 de 79%.

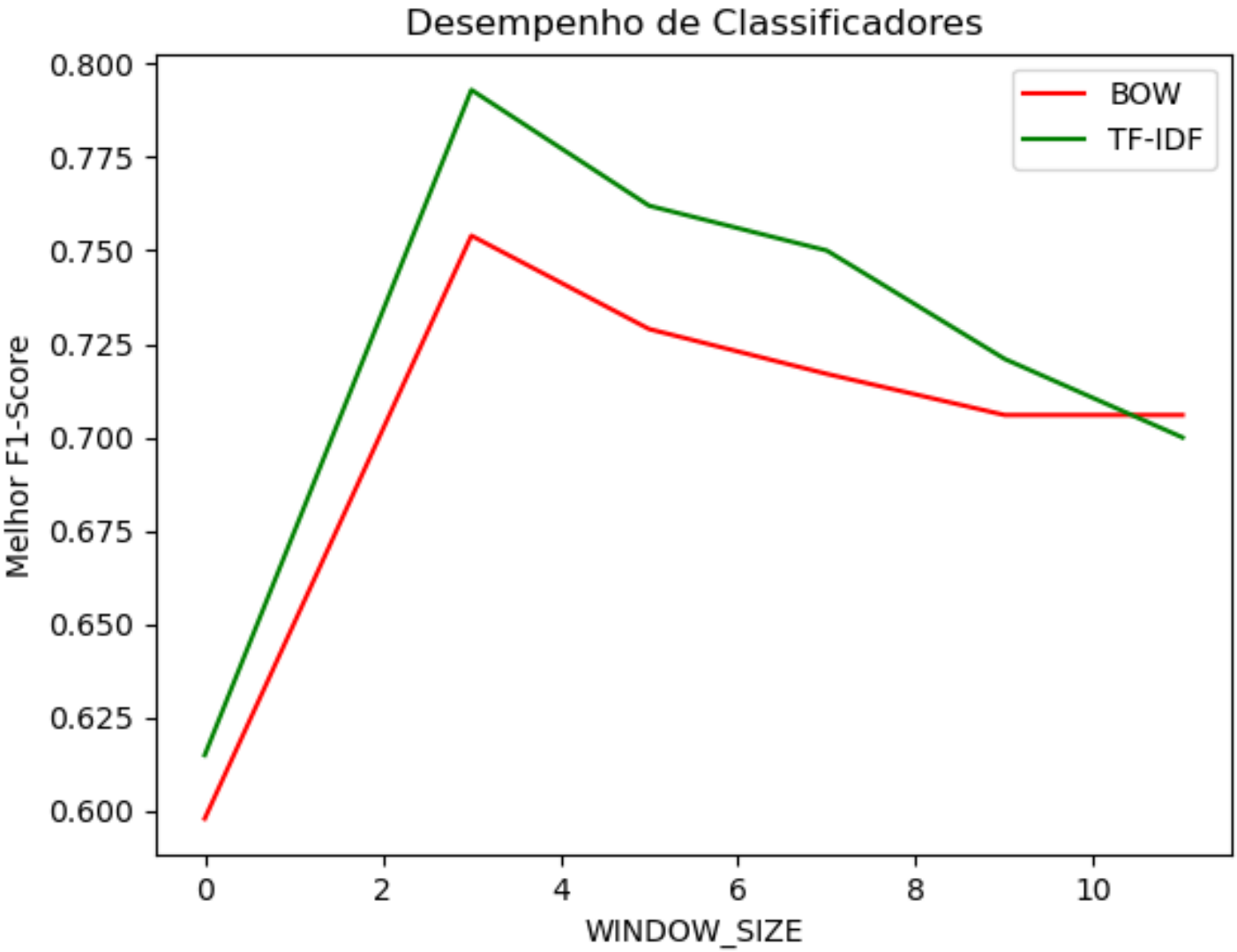


Figura 16 – Melhores desempenhos alcançados pelos classificadores testados em função do tamanho de janela de texto.

Fonte: Autor

6 Conclusões

Neste trabalho foram implementados os componentes de um sistema que visa realizar a segmentação de publicações de diários oficiais a partir dos arquivos PDF dos mesmos, extraídos de fontes *online*. Para isso, foi implementado um sistema que utiliza ferramentas para realizar o refluxo de textos de arquivos PDF, e a extração deste texto através de reconhecimento óptico de caracteres, através da biblioteca *Tesseract*. Também foi realizada a anotação de uma base de dados para treinamento de um sistema de segmentação de publicações a partir dos arquivos anotados. Este problema foi tratado como um problema de classificação binária de linhas de texto, no qual o objetivo foi diferenciar entre aquelas linhas que correspondem a uma separação entre uma publicação e outra, e todas as demais linhas do arquivo. O sistema foi treinado utilizando-se as ferramentas e classificadores da biblioteca *Scikit-Learn*, e constatou-se que o classificador LinearSVC, com vetorização dos textos utilizando TF-IDF e tamanho de janela igual a 3, apresentou o melhor desempenho. Ele foi capaz de classificar corretamente cerca de 79% das linhas que correspondem a uma separação entre publicações.

Um desempenho de 79%, embora não seja excelente, pode ser considerado bom para o problema em questão. É interessante notar que embora o porcentagem de classificações corretas tenha ficado em torno de 79%, o desempenho do classificador provavelmente é maior do que isso. Isso se deve ao fato que, devido à natureza do problema, a atribuição de classe 1 a uma linha localizada algumas linhas antes ou depois do início real de uma publicação, não tem um impacto tão grande sobre o verdadeiro desempenho do sistema.

6.1 Trabalhos Futuros

Como propostas de melhoria em trabalhos futuros, é possível citar a utilização de mais arquivos na base de dados para treinamento do sistema. Também pode-se tentar realizar a implementação de modelos mais sofisticados de vetorização, como *word embeddings* ou *sentece embeddings*, e de classificação, como por exemplo modelos baseados em topologias mais sofisticadas de redes neurais. Também é preciso explorar mais a fundo a otimização de hiper-parâmetros para os classificadores que obtiveram melhores resultados neste trabalho.

Outro ponto onde seria interessante buscar melhorias diz respeito ao processo de anotação dos textos para treinamento. A anotação neste trabalho foi feita de maneira inteiramente manual, conferindo-se os arquivos de texto extraídos dos PDFs originais e marcando-se manualmente em um segundo arquivo as linhas que deveriam receber o rótulo 1. Seria interessante então buscar formas de automatizar este processo e deixá-lo mais eficiente, uma vez que constitui a etapa mais demorada do trabalho.

Os próximos passos no desenvolvimento do projeto contarão com a inclusão de mais diários na base de dados do sistema, de forma a representar bem a variabilidade dos formatos de diários entre os diversos municípios e estados do país. Também pretende-se incluir em um trabalho futuro um conjunto de extratores de diários criados com a biblioteca *Scrapy* para extração de diários estaduais.

Uma abordagem interessante a testar para elaboração de uma nova estratégia de segmentação é aquela desenvolvida em (KOSHOREK et al., 2018), onde os autores utilizaram classificadores baseados em redes LSTM e *sentence embeddings* para vetorização de sentenças.

Referências

ABU KAUSAR, Mohammad; DHAKA, Vijaypal; SINGH, Sanjeev. Web Crawler: A Review. **International Journal of Computer Applications**, v. 63, p. 31–36, fev. 2013. DOI: 10.5120/10440-5125. Citado na p. 17.

ADOBE. **PDF Reference fifth edition**. [S.l.], 2004. Citado nas pp. 20, 21.

CARR, Les. **Web Data Formats**. [S.l.], 2008. Disponível em: <http://edshare.soton.ac.uk/1233/127/Formats.pdf>. Citado na p. 20.

CHAUDHURI, Arindam et al. Optical Character Recognition Systems for Different Languages with Soft Computing. v. 352, jan. 2017. DOI: 10.1007/978-3-319-50252-6. Citado na p. 24.

CHOMSKY, Noam. **Syntactic Structures**. [S.l.]: Mouton & Co., 1957. Citado na p. 26.

DAHL, Deborah. Natural Language Processing: Past, Present and Future. In: [s.l.: s.n.], jan. 2013. ISBN 978-1-4614-6017-6. DOI: 10.1007/978-1-4614-6018-3_4. Citado nas pp. 26, 27.

FRADKOV, Alexander L. Early History of Machine Learning. **IFAC-PapersOnLine**, v. 53, n. 2, p. 1385–1390, 2020. 21st IFAC World Congress. ISSN 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1888. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405896320325027>. Citado na p. 24.

GOOGLE. **Google Text Classification Workflow**. [S.l.: s.n.], 2022. Disponível em: <https://developers.google.com/machine-learning/guides/text-classification/>. Citado na p. 15.

JURAFSKY, Daniel; MARTIN, James. **Speech and Language Processing**. [S.l.: s.n.], jan. 2000. ISBN 0130950696. Citado nas pp. 26–29.

KHDER, Moaiad Ahmad. Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. **International Journal of Advances in Soft Computing and its Applications**, 2021. Citado na p. 16.

KIENLE, Holger; MÜLLER, Hausi. Legal aspects of web systems. In: p. 97–102. ISBN 978-1-4799-1608-5. DOI: 10.1109/WSE.2013.6642424. Citado na p. 19.

KOSHOREK, Omri et al. Text Segmentation as a Supervised Learning Task. In: PROCEEDINGS of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). New Orleans, Louisiana: Association for Computational Linguis-

tics, jun. 2018. P. 469–473. DOI: 10.18653/v1/N18-2075. Disponível em: <https://aclanthology.org/N18-2075>. Citado na p. 53.

KROTOV, Vlad; SILVA, Leiser. Legality and Ethics of Web Scraping. In. Citado na p. 19.

LIMA, Marcos Cavalcanti. **Deep Vacuity: Detecção e Classificação Automática de Padrões com Risco de Conluio em Dados Públicos de Licitações de Obras**. Set. 2021. Diss. (Mestrado) – Univesidade de Brasília, Brasília. Citado na p. 35.

LIU, Jiaying et al. Artificial Intelligence in the 21st Century. **IEEE Access**, PP, p. 1–1, mar. 2018. DOI: 10.1109/ACCESS.2018.2819688. Citado na p. 11.

OKBR. **Operação Serenata de Amor**. [S.l.: s.n.], 2016. Disponível em: <https://serenata.ai/about/>. Citado na p. 33.

OKBR. **Sobre o Querido Diário**. [S.l.: s.n.], 2022. Disponível em: <https://queridodiario.ok.org.br/sobre>. Citado na p. 32.

PANJWANI, Saurabh; UPPAL, Abhinav; CUTRELL, Edward. Script-Agnostic Reflow of Text in Document Images. In: p. 299–302. DOI: 10.1145/2037373.2037419. Citado na p. 23.

POTHEN, Ashlyn S. Artificial Intelligence and its Increasing Importance. In: [s.l.: s.n.], jan. 2022. P. 74–81. ISBN 978-93-92995-15-6. Citado na p. 11.

RAMAGERI, Bharati M. DATA MINING TECHNIQUES AND APPLICATIONS. **Indian Journal of Computer Science and Engineering**, 2010. Citado na p. 16.

S., Akhil. Overview of Tesseract OCR engine, dez. 2016. Citado na p. 24.

SCHANTZ, H.F.; ASSOCIATION, Recognition Technologies Users. **The History of OCR, Optical Character Recognition**. [S.l.]: Recognition Technologies Users Association, 1982. ISBN 9780943072012. Disponível em: <https://books.google.com.br/books?id=VehRAAAAMAAJ>. Citado na p. 24.

SMITH, Ray. History of the Tesseract OCR engine: what worked and what didn't. **Proceedings of SPIE - The International Society for Optical Engineering**, p. 02–, fev. 2013. DOI: 10.1117/12.2010051. Citado na p. 24.

TURING, Alan. Computing Machinery and Intelligence. In: [s.l.: s.n.], out. 1950. v. 59. DOI: 10.1093/mind/LIX.236.433. Citado na p. 26.

WEIZENBAUM, Joseph. ELIZA - A Computer Program For the Study of Natural Language Communication Between Man And Machine (Reprint). **Commun. ACM**, v. 26, p. 23–28, jan. 1983. DOI: 10.1145/357980.357991. Citado na p. 26.

ZAKI, Mohammed; MEIRA JR, Wagner. **Data Mining and Machine Learning: Fundamental Concepts and Algorithms**. [S.l.: s.n.], jan. 2020. ISBN 9781108473989. DOI: 10.1017/9781108564175. Citado nas pp. 24, 25, 30.

7 Apêndice

Neste apêndice são apresentadas as tabelas contendo os resultados detalhados to desempenho dos classificadores testados neste trabalho.

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.992	0.957	0.974	40641.000	0.960
ComplementNB	class 1	0.219	0.613	0.323	796.000	0.200
BernoulliNB	class 0	0.981	0.998	0.989	40641.000	0.989
BernoulliNB	class 1	0.211	0.030	0.053	796.000	0.048
MultinomialNB	class 0	0.983	0.998	0.990	40641.000	0.986
MultinomialNB	class 1	0.500	0.097	0.162	796.000	0.099
NearestCentroid	class 0	0.990	0.829	0.903	40641.000	0.901
NearestCentroid	class 1	0.063	0.590	0.115	796.000	0.113
LogisticRegression	class 0	0.987	0.995	0.991	40641.000	0.991
LogisticRegression	class 1	0.592	0.334	0.427	796.000	0.419
SGDClassifier	class 0	0.984	0.999	0.992	40641.000	0.991
SGDClassifier	class 1	0.853	0.153	0.260	796.000	0.253
DecisionTreeClassifier	class 0	0.993	0.991	0.992	40641.000	0.992
DecisionTreeClassifier	class 1	0.574	0.623	0.598	796.000	0.575
LinearSVC	class 0	0.989	0.994	0.992	40641.000	0.991
LinearSVC	class 1	0.598	0.441	0.508	796.000	0.480
PassiveAggressiveClassifier	class 0	0.992	0.987	0.989	40641.000	0.989
PassiveAggressiveClassifier	class 1	0.471	0.609	0.532	796.000	0.457
Perceptron	class 0	0.991	0.989	0.990	40641.000	0.989
Perceptron	class 1	0.496	0.534	0.515	796.000	0.470
RidgeClassifier	class 0	0.984	0.998	0.991	40641.000	0.991
RidgeClassifier	class 1	0.684	0.168	0.270	796.000	0.238

Tabela 2 – Resultados de treinamento, WINDOW_SIZE igual a zero e vetorização *Bag-of-Words*.

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.987	0.973	0.980	40641.000	0.966
ComplementNB	class 1	0.203	0.349	0.256	796.000	0.149
BernoulliNB	class 0	0.981	0.998	0.989	40641.000	0.989
BernoulliNB	class 1	0.211	0.030	0.053	796.000	0.048
MultinomialNB	class 0	0.981	1.000	0.990	40641.000	0.990
MultinomialNB	class 1	0.000	0.000	0.000	796.000	0.000
NearestCentroid	class 0	0.995	0.879	0.933	40641.000	0.935
NearestCentroid	class 1	0.112	0.780	0.196	796.000	0.201
LogisticRegression	class 0	0.987	0.996	0.991	40641.000	0.991
LogisticRegression	class 1	0.623	0.312	0.415	796.000	0.377
SGDClassifier	class 0	0.981	1.000	0.990	40641.000	0.990
SGDClassifier	class 1	0.000	0.000	0.000	796.000	0.000
DecisionTreeClassifier	class 0	0.992	0.993	0.993	40641.000	0.993
DecisionTreeClassifier	class 1	0.621	0.608	0.615	796.000	0.611
LinearSVC	class 0	0.989	0.995	0.992	40641.000	0.992
LinearSVC	class 1	0.628	0.443	0.520	796.000	0.477
PassiveAggressiveClassifier	class 0	0.988	0.994	0.991	40641.000	0.991
PassiveAggressiveClassifier	class 1	0.591	0.407	0.482	796.000	0.468
Perceptron	class 0	0.992	0.985	0.989	40641.000	0.989
Perceptron	class 1	0.443	0.607	0.512	796.000	0.479
RidgeClassifier	class 0	0.985	0.998	0.991	40641.000	0.991
RidgeClassifier	class 1	0.658	0.201	0.308	796.000	0.287

Tabela 3 – Resultados de treinamento, WINDOW_SIZE igual a zero e vetorização TF-IDF.

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.992	0.949	0.970	40641.000	0.970
ComplementNB	class 1	0.192	0.614	0.292	796.000	0.261
BernoulliNB	class 0	0.982	0.996	0.989	40641.000	0.988
BernoulliNB	class 1	0.228	0.067	0.103	796.000	0.037
MultinomialNB	class 0	0.990	0.973	0.981	40641.000	0.980
MultinomialNB	class 1	0.263	0.485	0.341	796.000	0.265
NearestCentroid	class 0	0.993	0.816	0.896	40641.000	0.895
NearestCentroid	class 1	0.071	0.711	0.128	796.000	0.132
LogisticRegression	class 0	0.994	0.998	0.996	40641.000	0.995
LogisticRegression	class 1	0.843	0.682	0.754	796.000	0.748
SGDClassifier	class 0	0.993	0.998	0.996	40641.000	0.995
SGDClassifier	class 1	0.861	0.655	0.744	796.000	0.724
DecisionTreeClassifier	class 0	0.995	0.996	0.995	40641.000	0.995
DecisionTreeClassifier	class 1	0.767	0.740	0.753	796.000	0.730
LinearSVC	class 0	0.995	0.995	0.995	40641.000	0.995
LinearSVC	class 1	0.764	0.742	0.753	796.000	0.740
PassiveAggressiveClassifier	class 0	0.994	0.997	0.995	40641.000	0.995
PassiveAggressiveClassifier	class 1	0.796	0.695	0.742	796.000	0.742
Perceptron	class 0	0.994	0.995	0.994	40641.000	0.994
Perceptron	class 1	0.730	0.692	0.711	796.000	0.706
RidgeClassifier	class 0	0.988	0.997	0.992	40641.000	0.992
RidgeClassifier	class 1	0.678	0.357	0.467	796.000	0.468

Tabela 4 – Resultados de treinamento, WINDOW_SIZE igual a 3 e vetorização *Bag-of-Words*.

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.982	0.977	0.980	40641.000	0.977
ComplementNB	class 1	0.085	0.111	0.096	796.000	0.050
BernoulliNB	class 0	0.982	0.996	0.989	40641.000	0.988
BernoulliNB	class 1	0.228	0.067	0.103	796.000	0.037
MultinomialNB	class 0	0.981	0.996	0.988	40641.000	0.987
MultinomialNB	class 1	0.069	0.015	0.025	796.000	0.003
NearestCentroid	class 0	0.996	0.915	0.954	40641.000	0.955
NearestCentroid	class 1	0.160	0.828	0.269	796.000	0.279
LogisticRegression	class 0	0.993	0.998	0.995	40641.000	0.995
LogisticRegression	class 1	0.844	0.626	0.719	796.000	0.729
SGDClassifier	class 0	0.990	0.999	0.995	40641.000	0.994
SGDClassifier	class 1	0.942	0.469	0.626	796.000	0.621
DecisionTreeClassifier	class 0	0.995	0.995	0.995	40641.000	0.995
DecisionTreeClassifier	class 1	0.758	0.736	0.747	796.000	0.729
LinearSVC	class 0	0.995	0.997	0.996	40641.000	0.996
LinearSVC	class 1	0.826	0.763	0.793	796.000	0.796
PassiveAggressiveClassifier	class 0	0.996	0.995	0.996	40641.000	0.996
PassiveAggressiveClassifier	class 1	0.772	0.785	0.778	796.000	0.780
Perceptron	class 0	0.996	0.994	0.995	40641.000	0.995
Perceptron	class 1	0.708	0.776	0.741	796.000	0.748
RidgeClassifier	class 0	0.989	0.999	0.994	40641.000	0.993
RidgeClassifier	class 1	0.865	0.426	0.571	796.000	0.548

Tabela 5 – Resultados de treinamento, WINDOW_SIZE igual a 3 e vetorização TF-IDF.

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.993	0.927	0.959	40641.000	0.963
ComplementNB	class 1	0.153	0.673	0.249	796.000	0.243
BernoulliNB	class 0	0.984	0.988	0.986	40641.000	0.986
BernoulliNB	class 1	0.222	0.173	0.195	796.000	0.099
MultinomialNB	class 0	0.991	0.956	0.973	40641.000	0.975
MultinomialNB	class 1	0.199	0.557	0.294	796.000	0.270
NearestCentroid	class 0	0.990	0.801	0.885	40641.000	0.885
NearestCentroid	class 1	0.054	0.584	0.099	796.000	0.105
LogisticRegression	class 0	0.993	0.997	0.995	40641.000	0.995
LogisticRegression	class 1	0.785	0.641	0.705	796.000	0.692
SGDClassifier	class 0	0.990	0.999	0.994	40641.000	0.994
SGDClassifier	class 1	0.874	0.469	0.610	796.000	0.663
DecisionTreeClassifier	class 0	0.994	0.995	0.995	40641.000	0.994
DecisionTreeClassifier	class 1	0.747	0.712	0.729	796.000	0.715
LinearSVC	class 0	0.994	0.994	0.994	40641.000	0.994
LinearSVC	class 1	0.706	0.685	0.695	796.000	0.689
PassiveAggressiveClassifier	class 0	0.992	0.996	0.994	40641.000	0.994
PassiveAggressiveClassifier	class 1	0.730	0.598	0.657	796.000	0.665
Perceptron	class 0	0.992	0.995	0.993	40641.000	0.993
Perceptron	class 1	0.689	0.607	0.645	796.000	0.625
RidgeClassifier	class 0	0.986	0.997	0.992	40641.000	0.991
RidgeClassifier	class 1	0.661	0.281	0.395	796.000	0.378

Tabela 6 – Resultados de treinamento, WINDOW_SIZE igual a 5 e vetorização *Bag-of-Words*..

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.982	0.981	0.981	40641.000	0.979
ComplementNB	class 1	0.057	0.059	0.058	796.000	0.020
BernoulliNB	class 0	0.984	0.988	0.986	40641.000	0.986
BernoulliNB	class 1	0.222	0.173	0.195	796.000	0.099
MultinomialNB	class 0	0.981	0.996	0.988	40641.000	0.987
MultinomialNB	class 1	0.000	0.000	0.000	796.000	0.000
NearestCentroid	class 0	0.996	0.903	0.947	40641.000	0.948
NearestCentroid	class 1	0.143	0.830	0.245	796.000	0.251
LogisticRegression	class 0	0.992	0.997	0.995	40641.000	0.995
LogisticRegression	class 1	0.818	0.611	0.699	796.000	0.686
SGDClassifier	class 0	0.988	0.999	0.994	40641.000	0.993
SGDClassifier	class 1	0.904	0.377	0.532	796.000	0.498
DecisionTreeClassifier	class 0	0.994	0.995	0.995	40641.000	0.994
DecisionTreeClassifier	class 1	0.739	0.711	0.725	796.000	0.709
LinearSVC	class 0	0.995	0.996	0.995	40641.000	0.996
LinearSVC	class 1	0.785	0.740	0.762	796.000	0.772
PassiveAggressiveClassifier	class 0	0.995	0.995	0.995	40641.000	0.995
PassiveAggressiveClassifier	class 1	0.746	0.729	0.737	796.000	0.750
Perceptron	class 0	0.994	0.995	0.995	40641.000	0.994
Perceptron	class 1	0.747	0.686	0.715	796.000	0.723
RidgeClassifier	class 0	0.987	0.998	0.993	40641.000	0.993
RidgeClassifier	class 1	0.805	0.353	0.491	796.000	0.468

Tabela 7 – Resultados de treinamento, WINDOW_SIZE igual a 5 e vetorização TF-IDF.

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.993	0.910	0.949	40641.000	0.956
ComplementNB	class 1	0.125	0.662	0.211	796.000	0.207
BernoulliNB	class 0	0.985	0.979	0.982	40641.000	0.982
BernoulliNB	class 1	0.176	0.234	0.201	796.000	0.120
MultinomialNB	class 0	0.991	0.939	0.964	40641.000	0.970
MultinomialNB	class 1	0.156	0.579	0.246	796.000	0.233
NearestCentroid	class 0	0.990	0.790	0.878	40641.000	0.879
NearestCentroid	class 1	0.051	0.582	0.094	796.000	0.095
LogisticRegression	class 0	0.992	0.996	0.994	40641.000	0.993
LogisticRegression	class 1	0.727	0.567	0.637	796.000	0.624
SGDClassifier	class 0	0.989	0.998	0.994	40641.000	0.993
SGDClassifier	class 1	0.817	0.454	0.583	796.000	0.615
DecisionTreeClassifier	class 0	0.994	0.995	0.995	40641.000	0.994
DecisionTreeClassifier	class 1	0.733	0.702	0.717	796.000	0.697
LinearSVC	class 0	0.993	0.994	0.993	40641.000	0.993
LinearSVC	class 1	0.660	0.628	0.644	796.000	0.642
PassiveAggressiveClassifier	class 0	0.992	0.993	0.993	40641.000	0.993
PassiveAggressiveClassifier	class 1	0.641	0.611	0.625	796.000	0.564
Perceptron	class 0	0.992	0.993	0.992	40641.000	0.991
Perceptron	class 1	0.623	0.570	0.595	796.000	0.573
RidgeClassifier	class 0	0.985	0.997	0.991	40641.000	0.991
RidgeClassifier	class 1	0.632	0.237	0.345	796.000	0.338

Tabela 8 – Resultados de treinamento, WINDOW_SIZE igual a 7 e vetorização *Bag-of-Words*..

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.981	0.982	0.981	40641.000	0.980
ComplementNB	class 1	0.008	0.008	0.008	796.000	0.006
BernoulliNB	class 0	0.985	0.979	0.982	40641.000	0.982
BernoulliNB	class 1	0.176	0.234	0.201	796.000	0.120
MultinomialNB	class 0	0.981	0.995	0.988	40641.000	0.987
MultinomialNB	class 1	0.005	0.001	0.002	796.000	0.002
NearestCentroid	class 0	0.997	0.883	0.936	40641.000	0.937
NearestCentroid	class 1	0.125	0.853	0.218	796.000	0.225
LogisticRegression	class 0	0.992	0.997	0.995	40641.000	0.994
LogisticRegression	class 1	0.807	0.584	0.678	796.000	0.650
SGDClassifier	class 0	0.986	1.000	0.993	40641.000	0.992
SGDClassifier	class 1	0.922	0.269	0.416	796.000	0.385
DecisionTreeClassifier	class 0	0.994	0.994	0.994	40641.000	0.994
DecisionTreeClassifier	class 1	0.717	0.717	0.717	796.000	0.704
LinearSVC	class 0	0.995	0.996	0.995	40641.000	0.995
LinearSVC	class 1	0.781	0.721	0.750	796.000	0.733
PassiveAggressiveClassifier	class 0	0.995	0.995	0.995	40641.000	0.995
PassiveAggressiveClassifier	class 1	0.723	0.725	0.724	796.000	0.716
Perceptron	class 0	0.993	0.995	0.994	40641.000	0.994
Perceptron	class 1	0.711	0.618	0.661	796.000	0.676
RidgeClassifier	class 0	0.986	0.998	0.992	40641.000	0.992
RidgeClassifier	class 1	0.767	0.298	0.429	796.000	0.409

Tabela 9 – Resultados de treinamento, WINDOW_SIZE igual a 7 e vetorização TF-IDF.

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.992	0.898	0.942	40641.000	0.952
ComplementNB	class 1	0.108	0.633	0.185	796.000	0.188
BernoulliNB	class 0	0.985	0.972	0.978	40641.000	0.980
BernoulliNB	class 1	0.151	0.256	0.190	796.000	0.137
MultinomialNB	class 0	0.991	0.928	0.958	40641.000	0.966
MultinomialNB	class 1	0.130	0.553	0.211	796.000	0.209
NearestCentroid	class 0	0.990	0.772	0.867	40641.000	0.869
NearestCentroid	class 1	0.049	0.603	0.091	796.000	0.092
LogisticRegression	class 0	0.991	0.995	0.993	40641.000	0.993
LogisticRegression	class 1	0.699	0.536	0.607	796.000	0.582
SGDClassifier	class 0	0.992	0.992	0.992	40641.000	0.993
SGDClassifier	class 1	0.608	0.616	0.612	796.000	0.571
DecisionTreeClassifier	class 0	0.994	0.995	0.994	40641.000	0.994
DecisionTreeClassifier	class 1	0.729	0.683	0.706	796.000	0.693
LinearSVC	class 0	0.992	0.992	0.992	40641.000	0.992
LinearSVC	class 1	0.612	0.607	0.609	796.000	0.606
PassiveAggressiveClassifier	class 0	0.991	0.993	0.992	40641.000	0.992
PassiveAggressiveClassifier	class 1	0.596	0.536	0.565	796.000	0.542
Perceptron	class 0	0.993	0.987	0.990	40641.000	0.991
Perceptron	class 1	0.503	0.655	0.569	796.000	0.492
RidgeClassifier	class 0	0.985	0.997	0.991	40641.000	0.991
RidgeClassifier	class 1	0.625	0.222	0.328	796.000	0.310

Tabela 10 – Resultados de treinamento, WINDOW_SIZE igual a 9 e vetorização *Bag-of-Words*..

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.981	0.984	0.982	40641.000	0.981
ComplementNB	class 1	0.005	0.004	0.004	796.000	0.004
BernoulliNB	class 0	0.985	0.972	0.978	40641.000	0.980
BernoulliNB	class 1	0.151	0.256	0.190	796.000	0.137
MultinomialNB	class 0	0.981	0.995	0.988	40641.000	0.987
MultinomialNB	class 1	0.005	0.001	0.002	796.000	0.002
NearestCentroid	class 0	0.997	0.865	0.926	40641.000	0.928
NearestCentroid	class 1	0.111	0.861	0.196	796.000	0.202
LogisticRegression	class 0	0.992	0.997	0.994	40641.000	0.994
LogisticRegression	class 1	0.794	0.570	0.664	796.000	0.624
SGDClassifier	class 0	0.984	1.000	0.992	40641.000	0.992
SGDClassifier	class 1	0.929	0.165	0.280	796.000	0.266
DecisionTreeClassifier	class 0	0.994	0.994	0.994	40641.000	0.994
DecisionTreeClassifier	class 1	0.710	0.700	0.705	796.000	0.691
LinearSVC	class 0	0.994	0.996	0.995	40641.000	0.995
LinearSVC	class 1	0.753	0.691	0.721	796.000	0.723
PassiveAggressiveClassifier	class 0	0.993	0.995	0.994	40641.000	0.994
PassiveAggressiveClassifier	class 1	0.711	0.663	0.686	796.000	0.692
Perceptron	class 0	0.993	0.995	0.994	40641.000	0.993
Perceptron	class 1	0.700	0.623	0.659	796.000	0.669
RidgeClassifier	class 0	0.986	0.998	0.992	40641.000	0.992
RidgeClassifier	class 1	0.753	0.268	0.395	796.000	0.383

Tabela 11 – Resultados de treinamento, WINDOW_SIZE igual a 9 e vetorização TF-IDF.

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.991	0.888	0.937	40641.000	0.949
ComplementNB	class 1	0.096	0.606	0.166	796.000	0.166
BernoulliNB	class 0	0.985	0.962	0.974	40641.000	0.977
BernoulliNB	class 1	0.126	0.275	0.172	796.000	0.129
MultinomialNB	class 0	0.989	0.920	0.954	40641.000	0.964
MultinomialNB	class 1	0.108	0.496	0.178	796.000	0.178
NearestCentroid	class 0	0.989	0.752	0.854	40641.000	0.856
NearestCentroid	class 1	0.043	0.574	0.081	796.000	0.081
LogisticRegression	class 0	0.990	0.995	0.992	40641.000	0.992
LogisticRegression	class 1	0.652	0.470	0.546	796.000	0.520
SGDClassifier	class 0	0.992	0.992	0.992	40641.000	0.990
SGDClassifier	class 1	0.591	0.598	0.595	796.000	0.520
DecisionTreeClassifier	class 0	0.994	0.995	0.994	40641.000	0.994
DecisionTreeClassifier	class 1	0.725	0.687	0.706	796.000	0.688
LinearSVC	class 0	0.991	0.993	0.992	40641.000	0.992
LinearSVC	class 1	0.595	0.545	0.569	796.000	0.565
PassiveAggressiveClassifier	class 0	0.989	0.996	0.992	40641.000	0.991
PassiveAggressiveClassifier	class 1	0.656	0.436	0.524	796.000	0.505
Perceptron	class 0	0.991	0.989	0.990	40641.000	0.990
Perceptron	class 1	0.507	0.555	0.530	796.000	0.410
RidgeClassifier	class 0	0.985	0.997	0.991	40641.000	0.991
RidgeClassifier	class 1	0.619	0.235	0.341	796.000	0.306

Tabela 12 – Resultados de treinamento, WINDOW_SIZE igual a 11 e vetorização *Bag-of-Words*..

Classificador	Classe	Precisão	Recall	F1-Score	Suporte	F1-Score (cross validation)
ComplementNB	class 0	0.981	0.984	0.982	40641.000	0.981
ComplementNB	class 1	0.003	0.003	0.003	796.000	0.004
BernoulliNB	class 0	0.985	0.962	0.974	40641.000	0.977
BernoulliNB	class 1	0.126	0.275	0.172	796.000	0.129
MultinomialNB	class 0	0.981	0.995	0.988	40641.000	0.987
MultinomialNB	class 1	0.005	0.001	0.002	796.000	0.005
NearestCentroid	class 0	0.997	0.850	0.917	40641.000	0.919
NearestCentroid	class 1	0.100	0.852	0.179	796.000	0.183
LogisticRegression	class 0	0.991	0.997	0.994	40641.000	0.994
LogisticRegression	class 1	0.768	0.536	0.632	796.000	0.592
SGDClassifier	class 0	0.983	1.000	0.991	40641.000	0.991
SGDClassifier	class 1	0.907	0.122	0.215	796.000	0.186
DecisionTreeClassifier	class 0	0.994	0.994	0.994	40641.000	0.994
DecisionTreeClassifier	class 1	0.695	0.675	0.685	796.000	0.681
LinearSVC	class 0	0.994	0.995	0.994	40641.000	0.994
LinearSVC	class 1	0.726	0.676	0.700	796.000	0.695
PassiveAggressiveClassifier	class 0	0.994	0.994	0.994	40641.000	0.994
PassiveAggressiveClassifier	class 1	0.695	0.676	0.685	796.000	0.665
Perceptron	class 0	0.991	0.995	0.993	40641.000	0.993
Perceptron	class 1	0.676	0.538	0.599	796.000	0.641
RidgeClassifier	class 0	0.986	0.998	0.992	40641.000	0.992
RidgeClassifier	class 1	0.726	0.256	0.379	796.000	0.361

Tabela 13 – Resultados de treinamento, WINDOW_SIZE igual a 11 e vetorização TF-IDF.