DISSERTAÇÃO DE MESTRADO

PROPOSTA DE UM MOTOR DE BUSCA METASEMÂNTICA COMO FERRAMENTA DE INVESTIGAÇÃO DE DADOS OBTIDOS ATRAVÉS DA INTERNET

LUCAS COELHO DE ALMEIDA

Brasília, Dezembro de 2022

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROPOSTA DE UM MOTOR DE BUSCA METASEMÂNTICA COMO FERRAMENTA DE INVESTIGAÇÃO DE DADOS OBTIDOS ATRAVÉS DA INTERNET

PROPOSAL FOR A METASEMANTIC SEARCH ENGINE AS A TOOL FOR INVESTIGATING DATA OBTAINED VIA THE INTERNET

LUCAS COELHO DE ALMEIDA

ORIENTADOR: FÁBIO LÚCIO LOPES DE MENDONÇA, DR.
COORIENTADOR: RAFAEL TIMÓTEO DE SOUSA JÚNIOR, DR.

DISSERTAÇÃO DE MESTRADO PROFISSIONAL EM
ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPEE.MP.034
BRASÍLIA/DF: DEZEMBRO – 2022

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO

PROPOSTA DE UM MOTOR DE BUSCA METASEMÂNTICA COMO FERRAMENTA DE INVESTIGAÇÃO DE DADOS OBTIDOS ATRAVÉS DA INTERNET

LUCAS COELHO DE ALMEIDA

Dissertação de Mestrado submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Fábio Lúcio Lopes de Mendonça, Ph.D, FT/UnB _____
Orientador/Presidente

Prof. Georges Daniel Amvame Nze, Ph.D, FT/UnB _____
Examinador Interno

Anderson C A Nascimento, Ph.D, School of Engineering and Technology/UW _____
Examinador Externo

Prof. Daniel Alves da Silva, Ph.D, FT/UnB _____
Examinador/Suplente

FICHA CATALOGRÁFICA

DE ALMEIDA, LUCAS COELHO
PROPOSTA DE UM MOTOR DE BUSCA METASEMÂNTICA COMO FERRAMENTA DE INVESTIGAÇÃO DE DADOS OBTIDOS ATRAVÉS DA INTERNET [Distrito Federal] 2022.
xvi, 114 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2022).
Dissertação de Mestrado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Search Engine	2. Semantic Search
3. Meta-Semantic Search	4. Natural Language Processing
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

DE ALMEIDA, L.C. (2022). *PROPOSTA DE UM MOTOR DE BUSCA METASEMÂNTICA COMO FERRAMENTA DE INVESTIGAÇÃO DE DADOS OBTIDOS ATRAVÉS DA INTERNET*. Dissertação de Mestrado, Departamento de Engenharia Elétrica, Publicação PPEE.MP.034, Universidade de Brasília, Brasília, DF, 114 p.

CESSÃO DE DIREITOS

AUTOR: LUCAS COELHO DE ALMEIDA
TÍTULO: PROPOSTA DE UM MOTOR DE BUSCA METASEMÂNTICA COMO FERRAMENTA DE INVESTIGAÇÃO DE DADOS OBTIDOS ATRAVÉS DA INTERNET.
GRAU: Mestre em Engenharia Elétrica ANO: 2022

LUCAS COELHO DE ALMEIDA
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Agradecimentos

Agradeço a minha esposa Anna, por todo apoio e suporte incondicional durante esses anos de intensos trabalhos, estudos e publicações, a qual esteve sempre presente e se tornou minha companheira em definitivo neste mesmo ano.

Agradeço a minha mãe, Cláudia Coelho, pelo cuidado e carinho em cada etapa, em cada dia e em cada mínimo detalhe e esforço para que eu pudesse me dedicar aos meus sonhos sem receios.

Agradeço a minha irmã, Priscila Coelho, e ao seu marido, Luiz Muller, pelo apoio e presença, e pela amizade eterna que me encoraja a prosseguir. Também pelas contribuições a respeito do caráter jurídico presente neste trabalho.

Agradeço a meu pai, Antônio Vicente, que desde meus primeiros anos me guia, aconselha e mostra o caminho. Seu esforço constante construiu o fundamento da minha incansável busca pelo aprimoramento e pela realização de sonhos não apenas meus, mas das pessoas que amo.

Agradeço ao meu coorientador Prof. Dr. Rafael T. de Sousa Jr., por todos os ensinamentos e oportunidades oferecidos durante a maior parte deste curso de mestrado. É fato que seus conselhos estarão sempre em minha memória.

Agradeço ao doutorando e amigo Francisco Filho pelas oportunidades, conselhos e por todo o suporte, suas contribuições foram chave para vários momentos da minha vida recente e para a finalização do curso.

Agradeço a família da minha esposa, e em especial à dona Ilma, Marli e Caroline, pois cada uma manteve um papel importante e central para que meu relacionamento e recente casamento estivesse vivo e saudável mesmo nos momentos mais difíceis.

Agradeço aos meus orientadores Professores Fábio Lúcio Lopes de Mendonça e Rafael Timóteo de Sousa Júnio pelo apoio na condução das pesquisas conduzidas no Laboratório UIoT e nas publicações e pelo acolhimento para a completa e bem sucedida finalização deste curso.

Agradeço ao pessoal da secretaria do curso, que sempre esteve disponível e me ajudou a resolver diversos problemas durante este período de pandemia pelo qual passamos.

Agredeço ao meu amigo Cláudio Pacheco, que embora tenha sido levado de nós durante a pandemia, sempre estará presente em nossas memórias. Seu amor ao próximo e sua crença em mim serão sempre combustível para que eu prossiga.

O Laboratório UIoT conta com apoio da Fundação de Apoio à Pesquisa do Distrito Federal FAPDF (Projetos UIoT 0193.001366/2016 e SSDDC 0193.001365/2016), bem como do Laboratório LATITUDE/UnB (Projeto SDN 23106.099441/2016-43). Agradeço às duas instituições pelo apoio e pelas bolsas. Agradeço o apoio técnico e computacional do Laboratório de Tecnologias para Tomada de Decisão - LATITUDE, da Universidade de Brasília, que

conta com o apoio do CNPq - Conselho Nacional de Pesquisa (Processo 312180/2019-5 PQ-2, e 465741/2014-2 INCT de Cibersegurança), pela CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Processo 88887.144009/2017-00 PROBRAL), pela FAP-DF - Fundação de Amparo à Pesquisa do Distrito Federal (Processo 0193.001366/2016 UIoT, e Projeto Brasileiro 0193.001365/2016 SSDDC), pelo Ministério da Economia (Operação 005/2016 DIPLA, e Outorga 083/2016 ENAP), pela Secretaria de Segurança Institucional da Presidência da República (Operação ABIN 002/2017), pelo Conselho Administrativo de Defesa Econômica (Outorga CADE 08700.000047/2019-14), da Advocacia Geral da União (Processo AGU 697.935/2019), do Departamento Nacional de Auditoria do SUS (Processo DENASUS 23106.118410/2020-85), da Procuradoria Geral da República para o Tesouro Nacional (Outorga PGFN 23106.148934/2019-67), e pela Universidade de Brasília (Outorga UnB COPEI 7129).

Durante o desenvolvimento deste mestrado, fui também bolsista pela FINATEC e pelo Departamento de Inovaçao da Universidade de Brasília. Agradeço também às duas instituições pelo apoio recebido.

Agradeço a Deus por esta oportunidade e por direcionar meus passos para que meus sonhos se realizem. De fato, eu não sou nem mais consciente que um cometa a vagar pelo espaço e nem mais enérgico que uma grande pedra depositada no fundo do mar, mas pela Sua misericódia, meus passos se acertam mais que a batida de um relógio, minhas mãos alcançam até a última das estrelas e minha energia se torna como uma árvore que dá frutos incessantemente.

RESUMO

A digitalização das relações e da informação têm aumentado a capacidade do ser humano de produzir dados de forma exponencial. Contudo, na mesma taxa em que novos dados são criados, é cada vez mais necessário também compreender e garimpar grandes bases de dados, inclusive sem qualquer estrutura ou formatação e com propósitos diversos. Nesse contexto, o uso de técnicas de indexação de dados usando motores de busca (do inglês *Search Engines*) e de interpretação de conjuntos de dados com o objetivo de classificá-los e categorizá-los se mostra indispensável para cenários de *Big Data* e *Data Lake*, em que a informação pode vir de diversas fontes com características técnicas e semânticas diferentes, exigindo classificações multi-classe e técnicas de processamento de linguagem natural, comumente designadas por técnicas de NLP (do inglês *Natural Language Processing*).

Adicionalmente, é preciso entender se as ferramentas de classificação têm viés e se os resultados são úteis e condizentes com o esperado, especialmente em contextos de investigação de crimes digitais. Esse é o problema da transparência da tomada de decisão, ou seja, da clara e/ou legível representação dos parâmetros que levaram a máquina a uma determinada decisão/classificação. Um sistema de investigação ideal, portanto, deveria ser capaz de indexar grandes bases de dados, entender a semântica e ser passível de adaptação/aprendizagem para atuar em diferentes cenários, e ao final do processo, ainda fornecer resultados enriquecidos com os parâmetros que levaram a máquina a tomar determinadas decisões para posterior auditoria da transparência no processo.

Portanto, esta dissertação tem como objetivo propor uma arquitetura fim a fim de um motor de busca que indexe e use interpretações metasemânticas baseadas em técnicas de processamento de linguagem natural em dados oriundos de páginas *Web*, de forma a prover, também, exemplos de parâmetros similares às classificações derivadas das amostras. O prefixo ”meta” no termo ”metasemântica” se refere a um conjunto de técnicas de classificação, predição e enriquecimento de dados aplicados para emular o processo de indexação semântica, porém preservando a auditabilidade do processo. Para efeito de validação da proposta, foram criadas amostras de páginas *Web* e utilizou-se bases de dados oficiais para treinamentos de instâncias de aprendizado de máquina para simulação de contextos reais de aplicação do projeto.

Como resultado, a validação mostra como o motor de busca proposto permite o armazenamento e processamento de dados sem formatação originários de páginas *Web* e aumenta a velocidade e objetividade com que investigações passam a ser realizadas e auditadas em contextos de processamento de linguagem natural, especialmente relevantes para contextos de crimes digitais.

ABSTRACT

The digitization of relationships and information has increased human beings' ability to produce data exponentially. However, at the same rate at which new data is created, it is increasingly necessary to understand and mine large databases, even without any structure or formatting and with different purposes. In this context, the use of data indexing techniques using search engines and the interpretation of datasets with the aim of classifying and categorizing them proves to be indispensable for scenarios of Big Data and Data Lake, where information can come from different sources with different technical and semantic characteristics, requiring multi-class classifications and natural language processing techniques, commonly known as NLP techniques, called Natural Language Processing techniques.

Additionally, it is necessary to understand whether the classification tools are biased and whether the results are useful and consistent with expectations, especially in cybercrime investigation contexts. This is the problem of decision-making transparency, that is, the clear and/or legible representation of the parameters that led the machine to a certain decision/classification. An ideal research system, therefore, should be able to index large databases, understand the semantics and be subject to adaptation/learning to act in different scenarios, and at the end of the process, still provide results enriched with the parameters that led to machine to make certain decisions for subsequent auditing of transparency in the process.

Therefore, this dissertation aims to propose an end-to-end architecture of a search engine that indexes and uses metasemantic interpretations based on natural language processing techniques on data from Web pages, in order to also provide examples of parameters similar to the classifications derived from the samples. The ”meta” prefix in the term ”metasemantics” refers to a set of classification, prediction and data enrichment techniques applied to emulate the semantic indexing process, while preserving the auditability of the process. For the purpose of validating the proposal, samples of Web pages were created and official databases were used to train instances of machine learning to simulate real contexts of application of the project.

As a result, the validation shows how the proposed search engine allows the storage and processing of plain data originating from Web pages and increases the speed and objectivity with which investigations are carried out and audited in language processing contexts natural, especially relevant to cybercrime contexts.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIAÇÕES

ESI	*Electronic Stored Information*
ETL	*Extract, Transform, Load*
Fess	*Full Text Search Server*
PDF	*Portable Document Format*
ISO	*International Organization for Standardization*
OCR	*Optical Character Recognition*
HTML	*Hyper Text Markup Language*
RFC	*Request for Comments*
API	*Application Programming Interface*
TF-IDF	*Term Frequency–Inverse Document Frequency*
NLP	*Natural Language Processing*
SVM	*Support Vector Machine*
YAML	*Yet Another Markup Language*
REST	*Representational State Transfer*
JSON	*JavaScript Object Notation*
CSV	*Comma Separated Values*
BASH	*Bourne Again Shell*

1 INTRODUÇÃO

Cada vez mais, as atividades que antes envolviam poucos indivíduos localizados todos numa mesma região física, agora envolvem diversos dispositivos, comumente mais de um por ator, e todos espalhados por remotas localidades. Em (MORGAN, 2014) tem-se uma comparação bastante extensa e interessante sobre as relações de trabalho e a evolução dessas. Adicionalmente, um evento ocorrido em determinado lugar não deixava muitos rastros digitais do acontecimento (se é que deixava algum), de forma que a verificação desses eventos se dava por poucas fontes e/ou testemunhas com relatos difusos baseados em lembranças e documentos e/ou objetos tidos como prova de determinado evento. Hoje, entretanto, conforme (CASEY, 2018), a maior parte dos eventos ocorre deixando algum rastro digital, e nos casos específicos das investigações de crimes, nunca foi tão atual a discussão do uso de tecnologias de *Big Data* e de *data mining* para que investigadores possam criar teses e compreender fontes diversas e sobrepostas de rastros deixados por um crime, como por exemplo filmagens, mensagens deixadas em fóruns virtuais, postagens em redes sociais, conversas em aplicativos de mensagens, gravações de ligações, depoimentos transcritos, fotos, entre outros. O tema desta dissertação, portanto, está intimamente ligado aos esforços para reunião e compreensão geral desses rastros digitais, especialmente para a verificação de crimes.

O campo da investigação de crimes, sejam esses cometidos de forma digital ou física, está formalmente contido dentro da Ciência Forense. Qualquer investigação pode se beneficiar da aplicação de técnicas da Ciência Forense, sempre com o objetivo de integrar diversas disciplinas e conhecimentos previamente obtidos para confirmar, com alguma precisão, determinado evento e os atores que participaram desse. Em (CAMBRIDGE, 2022) tem-se a correspondente definição formal do termo na língua inglesa. Note-se que não necessariamente tem de ocorrer um crime ou ato ilícito, na verdade a investigação é uma atividade bastante abrangente que busca criar uma tese que explique satisfatoriamente e da forma mais coerente possível determinado evento usando como base amostras de objetos e/ou, no caso do foco deste trabalho, dados gerados digitalmente. Entretanto, quando o termo ”forense” está presente, refere-se, portanto, à indicação do uso de técnicas multidisciplinares para investigações criminais, conforme significado do termo na língua portuguesa em (PRIBERAM, 2022a). E nesse caso, existem detalhes adicionais a serem explorados nos outros capítulos relacionados ao fato de que as teses criadas como fruto da investigação serão objetos a serem avaliados perante um tribunal.

Contudo, no contexto de investigação forense de um crime digital, percebe-se logo que existe alguma semelhança entre o papel de um investigador e as tarefas realizadas por um profissional que atua com análise de dados, afinal os dois casos dependem da reunião de numerosas informações de naturezas diversas. Ainda, com base na compreensão de todas as amostras de dados, os dois profissionais precisam formular teses, visualizações, organizar e categorizar os objetos de estudo e, por fim, apresentar uma conclusão analiticamente fundamentada. Essa similaridade é

notória especialmente em cenários de forense digital, em que as informações brutas apreendidas são fornecidas e esses profissionais precisam limpá-las, estruturá-las, armazená-las, e por fim encontrar padrões e correlações auditáveis. Todos esses pontos fazem alusão aos conceitos de *Big Data*, *Data Lake*, *Data Warehouse* e *Data mining*, especialmente a análise classificatória neste último, explorados em detalhes no próximo capítulo, conceitos corriqueiros para um profissional do campo da análise de dados. O contexto deste trabalho, portanto, abrange os aspectos e desafios técnicos do trabalho de investigação de massas de dados digitais aplicados à identificação de crimes.

Sabendo, portanto, que o resultado de uma investigação forense será uma tese a ser apresentada perante um tribunal, essa tese precisa trazer de forma clara e legível a materialidade dos eventos, a responsabilização dos atores, as circunstâncias e a possível autoria. O trabalho em (CHADREQUE, 2012) traz um ensaio completo sobre o papel da Estatística na investigação forense e como cada um desses elementos se beneficia do uso adequado de técnicas analíticas. Ainda que a decisão final recaia sobre a Corte, é por meio dos relatórios investigativos que a aplicação da Lei terá justa causa, preservará as provas e evidências dos eventos e evitará a imputação de culpa infundada, conforme (BARROS et al., 2021) mostra em um ensaio bastante completo sobre o papel da Ciência Forense no processo legal.

Entretanto, ainda que pareça simples e direta a aplicação dessas novas tecnologias em processos de investigação, existem diversos desafios, principalmente relacionados à praticidade, flexibilidade e efetividade das ferramentas disponíveis. Com base, portanto, nos entregáveis de uma tese de investigação, é direta a conclusão de que qualquer relatório criado precisa, necessariamente, estar acompanhado de justificativas analiticamente claras e auditáveis para fundamentar as inferências apresentadas. É exatamente neste ponto onde se começa a verificar os grandes desafios da investigação forense quando vivenciada num cenário de *Big Data*. Por exemplo, é possível que se torne extremamente oneroso e pouco desejável pedir a um grupo de investigadores que de fato revise, manualmente, todas as conversas gravadas, ligações e mensagens trocadas entre membros de uma quadrilha criminosa, ainda mais se os dados foram adquiridos ao longo de vários meses ou anos. A quantidade de informações será grande demais para que seres humanos consigam encontrar todas as relações, decifrem todos os códigos e gírias, e consigam criar teses bem fundamentadas em um espaço de tempo razoável. Pode vir a ser tarefa onerosa e por vezes inviável a eficiente manipulação de grandes massas de dados com a consequente formulação de teses bem fundamentadas e auditáveis, e é esse o problema para o qual este trabalho tentará propor uma contribuição.

Ainda, caso os investigadores decidam utilizar-se de complexas ferramentas de análise semântica e aprendizado profundo (do inglês *Deep Learning*), ainda que estes consigam resultados, não há garantias de que esses resultados tenham validade perante um tribunal e nem que esses resultados sejam satisfatórios, pois a fundamentação analítica e o padrão não linear de classificação e tomada de decisão dessas ferramentas põe em cheque a sua utilidade num julgamento. Mais ainda, não será possível nem ao menos medir o quão semelhantes são os resultados de uma investigação para dois casos semelhantes. O trabalho em (ALMEIDA et al., 2022) explica bem esse

problema, mostrando que apesar de parecerem desejáveis, a aplicação de técnicas demasiadamente complexas em atos de investigação podem acabar por comprometer e enviesar os relatórios finais.

Logo, aplicar os conceitos de mineração de dados (do inglês *data mining*) num processo forense de forma eficiente e satisfatória do ponto de vista jurídico se mostra, portanto, um grande desafio a ser resolvido. É necessária a adoção de ferramentas que não apenas auxiliem os investigadores a filtrar, classificar e interpretar grandes massas de dados de fontes diversas em tempo hábil, como também forneçam relatórios e informações que permitam a comparação entre teses de investigações de casos semelhantes. Esses pontos promovendo a transparências e auditabilidade ao processo, o que leva naturalmente ao conceito chamado de jurisprudência no Direito, conforme (PRIBERAM, 2022b), em que a definição da palavra faz menção sobre decisões passadas e reiteradas sobre determinados temas.

Para resolver tais problemas, existem conjuntos de técnicas já aplicadas e validadas em outros cenários como os de ferramentas de busca (do inglês *Search Engines*) públicas, os processos de trabalho em empresas utilizando técnicas de enriquecimento de dados (do inglês *Data Enrichment*), os algoritmos lineares de aprendizado de máquina (do inglês *Machine Learning*), os quais podem ter comportamento preditivo contínuo, e as técnicas de raspagem de dados (do inglês *Data Scraping*), em especial aquelas voltadas para conteúdos disponíveis em páginas *Web*, que comporão o objeto de estudo deste trabalho devido o seu comportamento mais anômalo e abrangente que outros tipos de fontes de dados.

Em vista do contexto descrito e dos desafios apresentados, esta dissertação irá apresentar a proposta e os testes de uma ferramenta de buscas fim a fim que possibilite a um investigador obter o conteúdo de páginas *Web* de diversos tipos e formatações, estruturar os dados, classificá-los adaptativamente em várias classes de acordo com uma base qualquer de dados de treinamento previamente fornecida e explorar esses dados através de relatórios. Esses relatórios conterão informações que possibilitem o entendimento do rastro do processo de decisão tomado pelas máquinas e a averiguação manual e humanamente legível da eficácia dos resultados. Todos os componentes serão abordados com base em motivações técnicas que são descritas a seguir e que levam a definição dos objetivos geral e específicos do trabalho de mestrado descritos mais adiante.

1.1 MOTIVAÇÃO

Conforme inicialmente apresentado, uma quantidade massiva de dados vem sendo gerada e usada para a investigação de crimes, sejam eles ocorridos no mundo físico, sejam no mundo virtual, através de aplicativos móveis e da Internet, por exemplo. Entretanto, nessa mescla de diversas fontes de dados, colocam-se problemas que são causados pela natureza rica e diversa de quantidade de informações, de tecnologias, protocolos, sistemas, integrações e significados. Tem-se cada vez mais dados a serem considerados, mais dificuldade em reunir todas as análises e

motivações em relatórios simples e legíveis, e mais ainda complexidade em exprimir com poucas e coesas palavras as metodologias usadas e a fundamentação de cada uma.

Para resolver os problemas relacionados à escala, o caminho mais natural tem sido recorrer ao apoio da tecnologia e da automação. O trabalho em (GUARINO, 2013) provê uma visão geral dos desafios e das vantagens da adoção de novas tecnologias e integrações no processo de investigação. Com o uso de técnicas de mineração de dados e de algoritmos de aprendizado de máquina, investigadores têm tido resultados bastante expressivos em filtrar as amostras mais significantes em meio a "oceanos" de dados. No documento de trabalho intitulado "Aprendizado de Máquina e antitruste" (LIMA, 2022), do CADE - Conselho Administrativo de Defesa Econômica (CADE, 2022) , vinculado ao Ministério da Justiça e Segurança Pública do Brasil (MJSP, 2022), pode-se verificar modelos e procedimentos de aprendizagem de máquina a ser utilizados na análise antitruste, ou seja, na verificação da atividade ilegal de alinhamento entre empresas para dominar um mercado e diminuir a concorrência, conforme (PRIBERAM, 2022c). Ainda, a empresa Kaspersky (KASPERSKY, 2022b) publicou um artigo (KASPERSKY, 2022a) em que são dados exemplos de tecnologias de *Big Data* e suas aplicações em casos reais, e um deles se refere a como a polícia da cidade de Seattle, no estado de Washington, nos Estados Unidos, criou um sistema para se basear em análises de grandes volumes de dados para localizar aglomerações de crimes pela cidade, conforme (GEEKWIRE, 2022). A postagem original pode ser vista em (POLICE, 2014).

Dentro desse contexto, uma categoria bastante desafiadora de crimes são aqueles em que o ato ilícito é verificado através da constatação do significado criminoso de determinada ação ocorrida no mundo digital, ou seja, quando uma postagem, um *website*, um comentário, um fórum, uma foto, entre outros, é disponibilizado *online* e fere algum princípio legal. Contudo, hoje, as classificações dadas por meio de algoritmos de aprendizado profundo, por exemplo, não são considerados durante o processo judicial, visto que não há linearidade e nem explicação clara e transparente das motivações de cada classificação. Em (SUPRA, 2022), uma abrangente discussão é fornecida, citando casos famosos do direito americano em que técnicas de processamento de dados foram bem vistas por juízes, como em (AL, 2022), em que um juiz entendeu ser aceitável o uso de técnicas computacionais baseadas em predição para a análise de grandes volumes de dados digitais, chamados de ESI, do inglês *Electronic Stored Information*, e chamando a atenção para critérios no uso de inteligência artificial em processos legais.

Portanto, nota-se que o desafio relacionado aos crimes digitais reside na complexidade de se reunir, organizar e categorizar provas, conforme (KARIE; VENTER, 2015), que mostra uma revisão dos principais desafios encontrados na área da Forense Digital. Contudo, mais além, os crimes cibernéticos ainda carecem de melhores padrões de julgamento e de identificação, uma vez que hoje é comum a discussão sobre qual o limiar para que uma atividade digital tenha se tornado crime, e o trabalho em (SOARES, 2022) revisa diversos temas nesse contexto sobre os limites de expressão em crimes digitais. Logo, vê-se uma clara e crescente divergência dentro do mundo da investigação e da aplicação das técnicas forenses: por um lado, as tecnologias de análise de dados podem ajudar os profissionais a identificar e delinear melhor e mais rápido os

crimes cibernéticos, principalmente aqueles cometidos através da Internet, todavia, quanto mais complexos e autônomos esses sistemas, mais difícil também se torna o julgamento e a validação dos resultados apresentados.

Como exemplo dos desafios, tome-se um caso em que um usuário de um fórum virtual seja indiciado pela publicação de um comentário racista. Existirão desafios, conforme (CONJUR, 2022b) exemplifica em caso semelhante e metodologias descritas por (GONÇALVES et al., 2012), podemos citar ao menos:

- Guardar as provas digitais do ato;
- Responsabilizar o usuário, ou seja, provar que aquele nome de usuário corresponde a uma determinada pessoa física e que de fato foi aquela pessoa, e não outra, a qual utilizou aquela conta virtual e promoveu a postagem do cometário;
- Provar, ou ao menos indicar com algum grau de precisão, que o comentário objeto da ação de fato teve conteúdo racista.

É curioso comparar esse mesmo caso hipotético com o caso em que, por exemplo, um aluno faça o mesmo comentário racista, porém fisicamente dentro de uma sala de aula de uma universidade. Nesse caso, em geral, segundo exemplo em (CONJUR, 2022a), os desafios serão:

- Encontrar testemunhas do ato;
- Colher as histórias e depoimentos das testemunhas;
- Verificar se existem filmagens e/ou gravações do ato e, caso existam, investigá-las usando os mesmos critérios de um crime digital.

Nota-se, portanto, que quando a informação não está precisamente representada (caso do crime cometido fisicamente), a investigação primeiramente busca reconstruir a evento através das pistas e da memória das testemunhas. Porém, comumente e quando disponíveis, acaba-se recorrendo às mídias digitais como prova do crime. Já no crime cibernético, como a informação já está precisamente representada (a postagem está ou esteve visível de forma inquestionável), porém distante da realidade tangível do mundo físico, é requerida muito mais precisão na aquisição, tratamento e comparação dos dados. Todavia, uma vez vencidos esses desafios, os investigadores apresentam o relatório ao tribunal com o comentário (ou a transcrição desse) e cabe à corte decidir pelo significado do ato.

Percebe-se, por fim, uma espécie de lacuna existente no relatório final das investigações: a indicação analítica do significado dos atos em geral não está presente de forma concreta, e muitas vezes ainda é pouco claro como e quão precisas essas conclusões devem ser. O trabalho em (SILVA, 2010) faz uma revisão do histórico da perícia no Brasil e da falta de legislação específica sobre partes da área. Ou seja, apesar de os investigadores dominarem a técnica de análise e comparação de dados, e de ter-se vasto conhecimento sobre a aplicação de aprendizado de máquina

em processamento de linguagem natural, das estruturas gramaticais as quais podem ser previsíveis e mensuráveis e da vasta teoria sobre vetorização de palavras para a análise da similaridade de sentenças e textos, a investigação forense acaba por deixar a cargo da corte a decisão sem prover fundamentação técnica sobre a significância dos achados, abrindo precedentes para discussões e questionamentos subjetivos.

Como já mencionado, uma das preocupações sobre a entrega de laudos resultantes de processos investigativos usando sistemas de classificação e aprendizado de máquina, ou ainda baseados em inteligência artificial, reside no risco da parcialidade, conforme (NUNES; MARQUES, 2018). Contudo, o que motiva esta dissertação é a proposta de um método de trabalho que fornece de modo intrínseco a transparência e auditabilidade dos resultados da investigação. Ou seja, nessa proposta reúne-se o uso de tecnologias de motores de busca, de classificadores lineares baseados em aprendizado de máquina, de técnicas de raspagem de dados de *websites* e de interfaces amigáveis e tecnologias multiplataforma para a experimentação de um método, ou *framework*, que possibilite:

- Automatizar etapas do processo de investigação;
- Classificar as amostras de dados colhidas;
- Prover relatórios instantâneos e auditáveis, com indicação de significado e possíveis jurisprudências relacionadas;
- Realizar o processo de investigação através de interfaces de usuário amigáveis e simples de ferramentas de buscas.

Nessa proposta, o uso de classificadores lineares irá afastar as possibilidades de comportamentos anômalos no tempo, ou seja, existirá a tendência de que para o mesmo conjunto de amostras, a classificação sugerida seja sempre a mesma. Além disso, o uso de técnicas de raspagem de dados de páginas *web* (do inglês *webscraping*) irá fornecer a base para a obtenção de texto puro de diversas https://www.overleaf.com/project/638c286ca5a60c35b28ef88bfontes possíveis (diversos *websites* com formatos e conteúdos diferentes). Ainda, o uso de enriquecimento de dados (do inglês *data enrichment*) irá prover a base auditável dos resultados mostrados, além de inflar a significância de cada evento e aumentar as chances desses serem revisados. Todas essas etapas resultarão num processo classificatório metasemântico, ou seja, os resultados terão dentro de si elementos atribuídos a análises semânticas, contudo, a amostra completa conterá também dados brutos e enriquecidos, os quais serão disponibilizados pela ferramenta de busca para investigação tanto por classes quanto por conteúdo, abrindo margem inclusive para que o investigador avalie se concorda com a máquina.

É interessante notar também que o uso de interfaces de ferramentas de busca tornará a experiência de investigação mais eficiente e amigável, uma vez que os usuários poderão se concentrar em buscar pelas categorias, ou classes de amostras, que lhe interessam e estudar por conta própria os resultados retornados pela ferramenta. E por fim, esses resultados serão fornecidos acompanhados de amostras da base de treinamento com exatamente as mesmas características (segundo

a máquina) do dado bruto sendo investigado. Dessa forma, se a base de dados fornecida for confiável e auditável, por exemplo uma relação de diversos crimes cibernéticos já julgados e suas classificações, o investigador obterá de forma instantânea as classificações e as possíveis jurisprudências. Dessa forma, a decisão continuará sendo responsabilidade devida do tribunal, contudo a perícia se tornará capaz de fornecer relatórios auditáveis, transparentes e com indicações de significância dos entregáveis obtidos analiticamente. Tais pressupostos levam à definição dos objetivos da dissertação a seguir apresentados.

1.2 OBJETIVOS DO TRABALHO

1.2.1 Objetivo Geral

Desenvolver e implementar uma arquitetura orientada a serviços de uma ferramenta de busca aberta que automatize etapas do processo de investigação forense de crimes cibernéticos e promova a geração automática de relatórios auditáveis e transparentes com base em classificações multiclasse obtidas através de aprendizado de máquina e enriquecimento de dados.

1.2.2 Objetivos Específicos

Detalhando o objetivo geral, os objetivos específicos englobam a concepção, o desenvolvimento e a validação dos itens da propostas, incluindo:

- Concepção de *framework* de investigação de crimes cibernéticos cometidos através da publicação e/ou compartilhamento de informação, especialmente, porém não limitado, a conteúdo escrito;
- Concepção de ferramenta de busca aberta, capaz de adquirir dados de páginas *web* e realizar o processamento, classificação e geração de relatórios de forma adaptativa;
- Concepção de serviço adaptativo de classificação de amostras que se adapte a uma base de dados fornecida inicialmente, treine diversas instâncias de aprendizado de máquina e permita a integração com a ferramenta de busca;
- Implementar um protótipo da proposta para efeito de validação, usando um cenário de teste fim a fim, que permita à ferramenta adquirir amostras de teste e obter resultados para análise e discussão.

1.3 TRABALHOS PUBLICADOS

Durante o desenvolvimento desta dissertação, foram publicados artigos científicos propondo soluções interoperáveis, onde serviços independentes se combinam para a entrega de uma tarefa

mais complexa e dividida em etapas. A pesquisa, que foi direcionada inicialmente à aquisição e processamento de dados de diversas fontes e formatos, entre elas a Internet, e posteriormente à aplicação da classificação de crimes usando enriquecimento de dados e aprendizado de máquina, resultou assim nas seguintes publicações:

1. C. de Almeida., L.; FILHO., F.; L. de MendonÇa., F.; Sousa Jr.., R. Meta-semantic search engine method proposition for transparent decision auditing. In: INSTICC. *Proceedings of the 19th International Conference on Smart Business Technologies - ICSBT,*. [S.l.]: SciTePress, 2022. p. 80–89. ISBN 978-989-758-587-6. ISSN 2184-772X.

2. ALMEIDA, L. C. de; PRADO, D. S. do; FILHO, F. L. de C.; MENDONÇA, F. L. de; JR, R. T. de S. et al. Design and implementation of a collaborative platform model for epidemic and collective health services. *IADIS International Journal on WWW/Internet*, v. 18, n. 2, 2020.

3. ALMEIDA, L. C. de; PRADO, D. S. do; LUCAS, M.; MARTINS, D. A.; OLIVEIRA, J. A. de; JÚNIOR, R. T. de S. Projeto e implementação de um sistema de visualização de dados de epidemias obtidos colaborativamente. CIAWI CIACA 2020 Proceedings, 2020.

4. ALMEIDA, L. C. d.; FILHO, F. L. d. C.; MARQUES, N. A.; PRADO, D. S. d.; MENDONÇA, F. L. L. d.; JR., R. T. d. S. Design and evaluation of a data collector and analyzer to monitor the covid-19 and other epidemic outbreaks. In: ROCHA, Á.; FERRÁS, C.; LÓPEZ-LÓPEZ, P. C.; GUARDA, T. (Ed.). *Information Technology and Systems*. Cham: Springer International Publishing, 2021. p. 23–35. ISBN 978-3-030-68285-9.

5. ALMEIDA, L. C. de; PRADO, D. S. do; MARQUES, N. A.; FILHO, F. L. de C.; MARTINS, L. M. C. e; SOUSA, R. T. de. Data science procedures to aggregate unstructured disease data in georeferenced spreading analysis. In: ROCHA, Á.; ADELI, H.; DZEMYDA, G.; MOREIRA, F.; CORREIA, A. M. R. (Ed.). *Trends and Applications in Information Systems and Technologies*. Cham: Springer International Publishing, 2021. p. 641–652. ISBN 978-3-030-72660-7.

6. PRADO, D. S. d.; FILHO, F. L. d. C.; ALMEIDA, L. C. d.; MARTINS, L. M. C. e.; MENDONÇA, F. L. L. d.; SOUSA, R. T. d. Design of a fog controller to provide an iot middleware with hierarchical interaction capability. In: ROCHA, Á.; FERRÁS, C.; LÓPEZ-LÓPEZ, P. C.; GUARDA, T. (Ed.). *Information Technology and Systems*. Cham: Springer International Publishing, 2021. p. 280–292. ISBN 978-3-030-68285-9.

Também cita-se os seguintes reconhecimentos:

1. Prêmio de Melhor artigo intitulado ”Meta-semantic Search Engine Method Proposition for Transparent Decision Auditing” - ICSBT 2022, realizado em Lisboa, Portugal

2. Prêmio de Melhor Artigo intitulado ”PROJETO E IMPLEMENTAÇÃO DE UM SISTEMA DE VISUALIZAÇÃO DE DADOS DE EPIDEMIAS OBTIDOS COLABORATIVAMENTE” - CIACA 2022, Lisboa, Portugal

1.4 METODOLOGIA DE PESQUISA

Nesta dissertação, a proposta de pesquisa foi dividida em três etapas para facilitar a compreensão do trabalho. Essa metodologia visa a aprofundar o estudo relacionado ao tema e aos problemas relacionados, identificando os assuntos abordados pela comunidade acadêmica e os desafios na construção de uma arquitetura que permita a automação do processo de investigação forense para crimes cibernéticos com a entrega de resultados que promovem a auditabilidade e transparência, conforme detalhado nas fases a seguir apresentadas.

1.4.1 Fase 1

Realizar pesquisa bibliográfica para identificar e analisar artigos que abordem os problemas de transparência na decisão tomada por algoritmos de aprendizado de máquina e aprendizado profundo, técnicas agnósticas de extração de dados da Internet, enriquecimento de dados em problemas de classificação, e problemas gerais na investigação de cibercrimes cometidos através da publicação e/ou compartilhamento de conteúdo.

1.4.2 Fase 2

Obter informações sobre tecnologias relacionáveis ao tema e sobre como elas podem auxiliar na resolução dos problemas encontrados na Fase 1.

1.4.3 Fase 3

Projetar, e implementar para fins de validação, uma arquitetura baseada em serviços que integra uma solução fim a fim de uma ferramenta de busca que apoie as atividades investigativas cumprindo os objetivos de automatizar etapas, atuar da forma mais agnóstica e adaptável possível e fornecer resultados auditáveis e transparentes.

1.5 CONTRIBUIÇÕES DO TRABALHO

O presente trabalho busca trazer as seguintes contribuições:

- Apresentação de um modelo fim a fim de arquitetura e *framework* de investigação de crimes cibernéticos baseados em compartilhamento e/ou publicação de conteúdo;
- Desenvolvimento de uma solução fim a fim aberta na qual o processo de investigação de cibercrimes possa se basear, especialmente para o caso de conteúdos escritos;
- Implementação funcional da proposta de arquitetura e ferramenta com posterior disponibilização dos códigos de software.

1.6 ORGANIZAÇÃO DO TRABALHO

Este trabalho foi ordenado em cinco capítulos, sendo este primeiro o de Introdução.

Para facilitar o entendimento da pesquisa, os demais capítulos estão organizados como descrito a seguir.

O Capítulo 2 oferece uma revisão atual das principais tecnologias utilizadas na mineração de dados e como essas técnicas podem ser aplicadas durante as investigações forenses para crimes digitais. O capitulo 2 contempla também uma revisão bibliográfica sobre os novos paradigmas computacionais na forense digital, assim como sobre técnicas de aquisição e filtragem de dados da Internet utilizadas pelas polícias para obter uma atuação mais abrangente na repressão aos cibercrimes.

O Capítulo 3 apresenta a arquitetura proposta do *framework* e da ferramenta resultante, composta de um serviço de classificação baseado em aprendizado de máquina (do inglês *machine learning*) e enriquecimento de dados (do inglês *data enrichment*), um serviço de raspagem de dados (*webscraping*, ou também do inglês *crawling*), e um serviço de ferramenta de busca (do inglês *search engine*), detalhando como ocorre a interação entre esses componentes.

O capitulo 4 apresenta os resultados práticos obtidos pela ferramenta, aplicando as técnicas de classificação e enriquecimento de dados.

Finalmente, O Capítulo 5 conclui este trabalho, sintetizando os resultados encontrados e sinalizando caminhos futuros, que podem ser seguidos para dar prosseguimento a este trabalho.

2 ESTADO DA ARTE E REVISÃO BIBLIOGRÁFICA

Este capítulo contém a revisão dos principais conceitos abordados nesta dissertação como motores de busca, raspagem de dados da Internet, garimpo de dados, processamento de linguagem natural, enriquecimento de dados e aprendizado de máquina, além de contextualizar brevemente as tecnologias e padrões utilizados aplicando-os à área. Ao final, considerações adicionais sobre os novos e correntes desafios na investigação de crimes digitais.

2.1 GARIMPO DE DADOS

A expressão garimpo de dados (do inglês *Data Mining*) se refere ao processo de busca e processamento de informação com o objetivo de se encontrar anomalias, padrões e/ou correlações entre diferentes amostras de uma ou mais bases de dados (SAS, 2022c).Um processo de garimpo de dados deve contemplar, no mínimo, os seguintes passos, conforme (LEE; SIAU, 2001):

- Adquirir ou capturar dados de uma grande base;
- Filtrar ou selecionar um universo limitado de informação para ser objeto do trabalho;
- Implementar soluções relacionadas à formatação, redução de dimensionalidade e limpeza dos dados;
- Treinar modelos adaptativos de processamento para obter os dados significativos desejados.

Essa atividade é ainda comumente classificada em pelo menos duas categorias bastante abrangentes com algumas subcategorias:

- Mineração Preditiva (do inglês *Predictive Data Mining*. Esse termo se refere ao processamento de dados previamente obtidos para que então possa-se prever um outro evento, seja no futuro, seja em outro conjunto de dados, conforme (WEISS; INDURKHYA, 1998). Essa categoria ainda pode ser dividida em mais quatro tipos de operações gerais:
  - Análise Classificatória (do inglês *Classification Analysis*): tem por objetivo decidir as classificações e apresentar dados organizados em diferentes e pré selecionadas categorias, comumente chamadas de classes.
  - Análise de Regressão (do inglês *Regression Analysis*): tem por objetivo encontrar padrões e correlações entre diferentes variáveis.
  - Análise de Séries Temporais (do inglês *Time-series Analysis*): também tem por objetivo encontrar padrões e correlações entre diferentes variáveis, com a diferença de

que todas elas estão relacionadas a amostras contidas no eixo temporal, de forma que outros modelos e transformações são passíveis de reuso.

  - Análise de Predição (do inglês *Prediction Analysis*): tem como objetivo a predição de valores de variáveis dependentes usando como base outras variáveis independentes.

- Mineração Descritiva (do inglês *Descriptive Data Mining*. Conforme (OLSON; LAUHOFF, 2019), se refere ao processamento de dados previamente obtidos para melhor classificá-los, organizá-los, descrevê-los ou ainda apresentá-los. Essa categoria também pode ser dividida em mais quatro tipos gerais:

  - Análise de Clusterização (do inglês *Clustering Analysis*): tem por objetivo agregar objetos e amostras de dados em grupos com características semelhantes. Se diferencia da análise classificatória pois as classes não são pré-selecionadas.
  - Análise de Sumarização (do inglês *Summarization Analysis*): tem por objetivo representar algum tipo de informação e/ou características de uma base de dados de forma mais fácil e intuitiva, como por exemplo através de gráficos de dispersão.
  - Aprendizado de Regras de Associação (do inglês *Association Rule Learning*): tem por objetivo encontrar padrões escondidos e não tão triviais entre diferentes variáveis de uma base de dados ou *dataset* do inglês.
  - Análise de Sequência de Descoberta (do inglês *Sequence Discovery Analysis*): tem por objetivo encontrar padrões significativos em bases de dados usando como referência alguma métrica pré estabelecida.

Todas essas técnicas são, de forma geral, preferíveis e aplicadas em conceitos de grandes volumes de dados, cenários nos quais já não é possível aplicar o uso de ferramentas manuais e/ou técnicas simples de exploração de dados, conceito de *Big Data*, segundo (SAS, 2022a). Nesses cenários, a geração de informação ocorre de forma descontrolada e muitas vezes sem padrão definido, seja na frequência, seja na formatação dos dados. Dessa forma, tem-se dados sem estrutura definida, sem padrão de formatação, originários de diversas fontes e com taxas de atualização diferentes. Quando esses dados são reunidos sem tratamento prévio e sem (ainda) um propósito único, tem-se então a formação de um Lago de Dados, do inglês *Data Lake* (SAS, 2022d). Esse repositório centralizado e repleto de dados brutos fornece, então, a base para realizar diversos processamentos, transformações e agregações, tudo com o objetivo de organizar esses dados e permitir análises mais significativas através de bases de dados estruturados, culminando no conceito de um Armazém de Dados, do inglês *Data Warehouse* (SAS, 2022b).

Vê-se portanto que a mineração de dados não é o nome de uma técnica global que resolve qualquer problemática relacionada a grandes volumes de informação, na verdade, é uma das várias peças na elaboração de um processo de investigação de dados.

Os trabalhos em (KARYDA; MITROU, 2007), (SUAIB; AKBAR; HUSAIN, 2020) e (SINDHU; MESHRAM, 2012) comentam sobre os desafios legais e técnicas e apresentam métodos disponí-

veis. Entretanto, ainda existe a lacuna de uma ferramenta que auxilie os investigadores do começo ao fim do processo, a qual este trabalho visa preencher.

Sendo assunto de interesse para esta dissertação e ferramentas de importância geral nas soluções de garimpo de dados, as principais tecnologias usadas para o desenvolvimento do experimento e relacionadas com esta seção serão descritas nas próximas subseções. De modo específico, a próxima subseção detalha alguns pontos importantes da Linguagem Python e sua relação tão próxima com a Ciência de Dados.

2.1.1 Linguagem *Python* e a Ciência de Dados

As análises no campo da Ciência de dados em geral dependem de programas de computador criados com um fim específico para cada tipo de processamento requerido, e essas implementações são em geral desenhadas e executadas usando linguagens de programação. Apesar de existirem diversas linguagens de programação, poucas conseguiram se espalhar por tantas indústrias e áreas de estudo como a Linguagem Python, conforme (SRINATH, 2017) observa. Esta é uma linguagem de alto nível, ou seja, o desenvolvedor experiente na linguagem irá se preocupar muito mais com a manipulação de suas variáveis e lógicas de programação do que com o uso de recursos, representação de informação e controle de fluxo de execução de seus programas.

Além disso, Python é também uma linguagem interpretada, em que os códigos em bytes são compilados, não as instruções. Ou seja, o desenvolvedor usará comandos de alto nível, bastante próximos de sua compreensão, e um interpretador irá traduzir esses comandos para instruções de máquina. Essas instruções de máquina, por sua vez, não são executadas diretamente pelo computador, na verdade, elas são repassadas a uma máquina virtual que é capaz de executá-las usando o computador hospedeiro da aplicação. Essa característica faz com que a linguagem seja independente de plataforma para a maioria das aplicações criadas, ou em outras palavras, faz com que o código desenvolvido possa ser transportado entre diversos sistemas operacionais e ser executado da mesma forma, mudando-se apenas as máquinas virtuais, conforme (ORG, 2022i). É também por causa dessa característica que a linguagem suporta escrita dinâmica, que é quando o programador é capaz de digitar e executar sequências de código em tempo real. Diferente das linguagens compiladas, em que há uma verificação geral antes da execução do programa, na linguagem Python, os possíveis problemas são identificados em tempo de execução. Existem críticas a esse modelo por tornar complexo o processo comumente chamado de *debug*, em que o programador precisa encontrar falhas no seu código, contudo, essa característica se torna fundamental na construção de grandes e complexos projetos, conforme conclui-se observando (DOWNEY, 2012).

Apesar de ser popular em diversas áreas, essa linguagem é especialmente importante na manipulação de dados. Por ter várias abstrações e por delegar várias responsabilidades importantes para o interpretador, diminuindo a carga sintática do desenvolvedor, se torna possível o desenvolvimento de programas complexos usando conceitos alto nível e poucas linhas de código (NAGPAL; GABRANI, 2019). Adicionalmente, a grande comunidade, como aponta o trabalho citado,

faz com que haja grande engajamento e evolução constante das bibliotecas dessa linguagem, que constituem o principal motivo da popularidade com a Ciência de Dados.

A próxima sub-seção detalha o uso de uma biblioteca específica da Linguagem Python que tem importante papel no processo de mineração de dados.

2.1.2 Biblioteca *Pandas*

O desenvolvimento de programas de computador usando linguagens de programação comumente depende do reuso de código, principalmente para o aproveitamento de rotinas já validadas pela comunidade e aumento da velocidade na implementação, conforme (HAEFLIGER; KROGH; SPAETH, 2008) aponta. Entre as bibliotecas voltadas para Ciência de Dados, e em especial para a linguagem Python, uma se tornou globalmente utilizada: a biblioteca Pandas. Inicialmente desenvolvida para análise de dados financeiros, a biblioteca tem seu nome derivado do inglês em ”panel data”, um termo comum para dados multidimensionais em Estatística e Econometria, conforme (HSIAO, 2007). Hoje, é um dos principais recursos da linguagem Python, sendo considerada umas das habilidades principais para trabalhar no campo da Ciência de Dados, conforme artigo em (TECHTARGET, 2022).

As principais característica são a otimização natural para grandes quantidades de dados e a forma de representação dos dados, que é feita através de uma estrutura tabular denominada, do inglês, *dataframe*. Um *dataframe pandas*, como é comumente referido, possui nomes para cada dimensão, linha e coluna relacionados a uma indexação hierárquica que facilita o uso de vários métodos prontos para manipulação de dados baseados nesses índices. Essa soma de pequenas soluções para manipulação de dados tabulares torna tarefas como ordenamento, alinhamento de *dataframes*, união, comparação, somas, interseções, cortes (também chamados no inglês de *slices*, em que se filtra os dados com base num conjunto de regras), e até mesmo rotações, conforme a documentação oficial em (ORG, 2022h).

Adicionalmente, a biblioteca também conta com funções prontas para tarefas complexas como o carregamento de dados disponíveis em arquivos e/ou fontes da Internet, inclusive com a possibilidade de análises multidimensionais e configuração de tratamento automatizado para problemas encontrados em linhas problemáticas de um arquivo usado como fontes de dados, por exemplo. É possível, portanto, perceber que o uso da biblioteca agiliza e facilita uma tarefa bastante complicada, principalmente em cenários de *Big Data*, que seria o carregamento e a preparação da informação provenientes de um Armazém de Dados (do inglês *Data Warehouse*) para o ambiente de análise em memória num sistema informatizado.

A próxima seção abordará conceitos gerais e específicos dos motores de busca e ao final, discorrerá brevemente sobre um projeto de código aberto de um motor de busca que foi escolhido para compor o experimento a ser descrito nos próximos capítulos desta dissertação.

2.2 MOTORES DE BUSCA

A necessidade de um motor de busca, ou comumente do inglês *Search Engine*, com interface gráfica em que o usuário digita algumas palavras e a ferramenta traz um apanhado de resultados parece, hoje, simples e popular, conforme nota (PURCELL; RAINIE; BRENNER, 2012). Entretanto, existe um conjunto de desafios e tecnicalidades que fazem com que esse conceito não seja tão trivial assim. A primeira necessidade é a de entender as etapas de funcionamento de um motor de busca. Segundo (HALAVAIS, 2017), é constituído de no mínimo três partes, ou processos:

- Aquisição e estruturação de dados das fontes predefinidas;
- Indexação dos dados adquiridos para a decisão quanto à relevância;
- Recebimento, por meio de alguma interface, de termos que guiarão a realização da busca em si na base de dados previamente tratada.

Isso significa que um motor de busca não deveria realizar a tarefa de aquisição e indexação de dados somente após receber a entrada do usuário, na verdade, sua lógica interna é bastante parecida com a de uma biblioteca de livros físicos: um funcionário organiza, previamente, os livros por tema e por outras categorias como data de publicação, letra inicial do título, faixa etária do público alvo, entre outros, e agrupa os livros iguais no mesmo local. Dessa forma, quando um cliente chega e requisita um livro específico, ou um tema, por exemplo, o funcionário é capaz de, em pouco tempo, mesmo num universo grande de exemplares, encontrar resultados relevantes para aquele cliente. Da mesma forma, um motor de busca precisa, antes de receber interações de usuários, preparar os dados e indexá-los de forma eficiente para conseguir, em pouco tempo, encontrar respostas eficientes para seus usuários, similar ao encontrado em (HALAVAIS, 2017).

Para a aquisição e estruturação de dados, é comum o uso de ferramentas de raspagem de dados e rotinas de ETL, do inglês *Extract, Transform, Load*, ou Extração, Transformação, Carregamento. Esses componentes e conceitos serão melhor comentados em seções posteriores deste capítulo, especialmente na próxima seção. Já sobre o processo de indexação, este resultará nos dados organizados de forma a facilitar e evidenciar características que ajudem futuras pesquisas a encontrar resultados relevantes. Diferenciar as duas etapas pode parecer complexo do ponto de vista de máquinas informatizadas, mas trazendo novamente a comparação com uma biblioteca de livros, se torna trivial: a etapa de aquisição e estruturação seria equivalente a um dono de uma biblioteca que pega todos os livros que possui e os armazena nas prateleiras usando qualquer característica geral, como por exemplo ordenamento alfabético dos títulos. Contudo, essa organização, ainda que ajude a enumerar os livros e a buscar nomes específicos, não ajudará a encontrar opções de temas semelhantes, ou ainda publicações de um mesmo autor, e é nesse momento que os funcionários da biblioteca entrarão em ação e irão remanejar os livros da melhor forma possível, fazendo alusão à etapa de indexação.

Nas próximas sub-seções, são apresentados e detalhados três tipos possíveis de abordagens na aplicação de um motor de busca numa ou mais bases de dados. De modo específico, a próxima

subseção detalha motores de busca baseados em palavras-chave, o primeiro e mais simples motor de busca a ser explorado.

2.2.1 Motores baseados em palavras-chave

Uma ferramenta de busca, como dito anteriormente, fará pesquisas em sua base usando um ou mais parâmetros recebidos pelo usuário, seja esse um ser humano ou outro programa. Esses parâmetros poderão ou não ser tratados previamente para que a busca se torne mais eficiente ou precisa. Esse ponto importante a ser observado é crucial para o completo entendimento das ferramentas de busca baseadas em palavras-chave. De fato, o conceito é extremamente simples, conforme (HALAVAIS, 2017) discorreu, o significado traduzido de uma ferramenta de busca baseada em palavras chaves seria o de um sistema de recuperação de de informação que permite o use de palavras-chave para a realização de buscas em textos digitalmente distribuídos. Isso significa que o resultado retornado por esse tipo de ferramenta de busca deve mostrar todos os locais em que foi encontrada uma exata cópia da entrada recebida. Ainda que essa entrada possa ser pré-processada para mudar sua representação, por exemplo, se tornando um vetor, como será discorrido em seções posteriores deste trabalho, não deve haver qualquer inferência adicional sobre o significado das entradas. Usando texto de linguagem natural é direta a compreensão esse conceito, visto que a ferramenta deverá entregar apenas ocorrências de uma ou mais palavras digitadas pelo usuário.

Apesar de o fluxo de funcionamento descrito previamente ser composto por poucos passos e baseado em buscas de cópias simples, a definição específica da granularidade de uma palavra-chave ainda pode permanecer em aberto, necessitando explicações complementares. Existem outros cenários em que esse conceito se torna mais completo e rico e um exemplo que, ainda que esteja relacionado a outro domínio de busca, é valioso para o trabalho, é o de um buscador gráfico. Para que uma ferramenta de busca de imagens realize as mesmas etapas de indexação descritas previamente e seja capaz de retornar ao usuário todos os arquivos em que uma determinada imagem for verificada, caso não haja qualquer tipo de pré-processamento, será necessário que a mesma figura, pixel a pixel, seja verificada em todos os documentos retornados ao usuário. Contudo, no caso da existência de um documento com essa mesma imagem, porém com cores trocadas, essa simples ferramenta de busca descartaria esse objeto e o usuário não veria o documento na lista de resultados. Adicionalmente, um documento com uma cópia exata, porém com qualidade reduzida, teria grandes chances de ser ignorado. Esse mesmo fenômeno pode ser verificado também no universo da linguagem natural, em que o usuário procura por expressões com várias palavras, ou ainda frases e textos inteiros, e a ferramenta de busca acaba descartando resultados que contém quase a totalidade do que o usuário digitou e que poderiam ser relevantes, porém que não representam cópias exatas da entrada fornecida. Esses exemplos servem ao propósito de facilitar a compreensão do conceito de uma palavra-chave, que em resumo, seria a menor unidade possível de texto que representa um conceito ou ideia a qual o usuário da ferramenta objetiva encontrar conteúdos relacionados e que contenham uma ou mais cópias exatas do termo,

conforme discussões em (HALAVAIS, 2017).

Outras implicações dos exemplos acima também podem ser observadas quando, de modo complementar, um usuário recebe resultados que contém cópias exatas das palavras-chave usadas, porém que não têm qualquer relevância para os objetivos daquela busca. Ou seja, além da probabilidade de ignorar resultados relevantes, ferramentas de busca baseadas em palavras-chave também podem entregar resultados precisos, porém sem qualquer valor para o usuário. Todos esses casos, por fim, são reflexo da decisão técnica de não envolver, no funcionamento da ferramenta, a inferência e decisão sobre a similaridade entre os resultados, apenas cópias exatas são admitidas.

Todavia, ainda que apresentem limitações, principalmente relacionadas ao significado dos resultados apresentados, uma ferramenta de busca baseada em palavras-chave também tem vantagens, principalmente relacionadas à velocidade e eficiência no uso de recursos, já que a busca ocorre da forma mais simples possível num sistema de armazenamento de dados: busca-se cópias exatas dos termos inseridos. Isso também implica em menos comparações a serem realizadas pelo programa para verificar as ocorrências das palavras-chave, e como as buscas objetivam encontrar cópias exatas, o uso de estratégias para mudar a representação da informação resultam, finalmente, num processo de pesquisa otimizado e rápido, conforme pode ser visto no *framework* proposto em (GIATSOGLOU et al., 2017).

A próxima sub-seção apresentará, em contraste aos motores de busca baseados em palavras-chave, a introdução de análise semântica ao processo, e os ganhos e perdas decorrentes dessa adição.

2.2.2 Motores baseados em análise semântica

De forma complementar aos exemplos abordados na subseção anterior, as ferramentas de busca semântica utilizarão técnicas de processamento de linguagem natural para criar mecanismos de comparação entre expressões, palavras-chave e/ou frases não mais baseados apenas na organização e nas palavras, mas sim, no sentido, ou significado, dos arranjo gramaticais, conforme (KASSIM; RAHMANY, 2009).

Existem diversos desafios na implementação de algoritmos de busca semântica, principalmente relacionados à velocidade e uso de recursos, conforme nota (DING et al., 2011). Isso porque, assim como a ferramenta de busca por palavras-chave depende de uma base de dados previamente tratada e indexada, um buscador semântico também depende de uma massa de dados previamente interpretada para ser capaz de, em tempo hábil, prover resultados satisfatórios ao usuário. Pelo fato de ainda não existirem conceitos claros sobre o armazenamento de significados, exatamente pela subjetividade deste conceito, conforme (XU et al., 2003) tenta explorar os requisitos formais de tal empreitada, não havendo numerosos avanços na área de armazenamento semântico de dados desde então, outras estratégias devem ser adotadas para o sucesso desses sistemas, como um buscador semântico, em que não basta apenas o uso da definição presente no

dicionário, este depende do entendimento do contexto ao redor do uso de determinada palavra ou expressão para identificar outras ocorrências semelhantes e solucionar o problema do armazenamento de dados com contexto semântico.

É nesse cenário que a relação entre aprendizado de máquina, inteligência artificial e ferramentas de busca se torna bastante necessária. Usando algoritmos de aprendizagem, por exemplo, relações entre termos e semelhanças encontradas através de métodos numéricos podem ser exploradas para a classificação de resultados quanto à sua similaridade com uma ou mais entradas fornecidas. Também, com o uso de aprendizado profundo e outras técnicas de inteligência artificial, padrões escondidos e de difícil representação podem ser verificados em tempo hábil, trazendo várias benefícios ao processo. Visto que já não há tanta preocupação com a precisão das palavras usadas, e sim com o sentido, o usuário da ferramenta terá menos dificuldade em decidir quais termos usar para a pesquisa. Em outras palavras, usando a mesma alusão anterior feita com buscadores gráficos, seria equivalente a uma ferramenta que, dada a foto de um carro qualquer, é capaz de retornar todas as outras fotos em que um carro fora observado. Esse fenômeno no domínio da linguagem natural seria exemplificado por um usuário que submete à máquina um texto sobre um determinado tema e o sistema analisa esse objeto, identifica o tema central ou os significados de várias frases mais significativas dentro do texto e compara com as mesmas classificações já realizadas em toda a sua base de dados. Uma prova da dificuldade técnica e computacional da concepção e sustentação de um programa como esse é o fato de que, hoje, portais famosos da Internet usam como forma de aumentar a sua segurança desafios baseados em classificação de imagens e de interpretação de texto para tentar barrar programas de consumirem serviços com os quais apenas humanos deveriam interagir, conforme (SINGH; PAL, 2014), que explica o funcionamento dos chamados *captchas*.

Todavia, nem todos os cenários são beneficiados pelo uso da análise semântica em processos de busca, justamente pela subjetividade das classificações escolhidas pela máquina. O trabalho em (ALMEIDA et al., 2022) exemplifica cenários de investigação de conversas criminosas em que o uso de classificadores semânticos poderia incorrer em prejuízo para a investigação devido à natural tendência dos criminosos em esconder e relativizar suas comunicações usando códigos e gírias. Além disso, a tarefa de treinamento de algoritmos é custosa computacionalmente e demanda bastante tempo, dependendo do tamanho das bases de dados, o que a torna onerosa em relação às ferramentas baseadas em palavras-chave. Também, nem sempre o sentido pretendido pelo usuário será o mesmo identificado e aplicado na pesquisa pela máquina, havendo probabilidade variável e maior de entrega de resultados sem qualquer valor para a busca.

A próxima subseção irá abordar um conceito intermediário, em que a análise semântica está presente, porém não como sistema decisor final, de modo que os resultados apresentados pelo motor de busca continuam centrados nas palavras e expressões chave usadas pelos usuários, porém com a semântica sendo usada como elemento calibrador.

2.2.3 Motores baseados em análise metasemântica

De acordo com (BURGESS; SHERMAN, 2014), Metasemântica é o estudo dos fundamentos constituintes da semântica da linguagem natural. Todavia, esse termo, quando associado a sistemas de busca, está relacionado não ao conceito formal metalinguístico, mas sim, ao fato de serem ferramentas que utilizam técnicas parciais de análise semântica para melhorar a qualidade dos resultados de suas pesquisas, conforme (MUKHOPADHYAY et al., 2013) e (ALMEIDA et al., 2022) mostram.

Esse é um conceito ainda pouco explorado, existindo apenas alguns trabalhos que exploram a fundo a completa definição e caracterização de uma ferramenta de busca metasemântica.

Dado que este trabalho segue a definição de que um método metasemântico seria baseado no uso limitado de algoritmos de aprendizagem de máquina não para a classificação final dos termos, mas sim para a melhora da qualidade dos resultados de uma ferramenta de busca baseada em palavras-chave, se fazem necessárias algumas explicações adicionais sobre essa categoria intermediária de buscadores. Novamente, recorrendo-se à alusão dos buscadores gráficos, um programa poderia utilizar algoritmos de aprendizagem de máquina para encontrar o ponto ideal de suavização da variação de cor dos pixels de uma imagem e adicionar transformações como por exemplo trocar cores por escala de luminosidade, e comparar o resultado dessas transformações com sua base de dados tratada previamente usando a mesma lógica. Nesse exemplo, essa máquina teria grandes probabilidades de encontrar toda e qualquer reprodução de uma determinada figura ainda que a qualidade ou as cores estejam diferentes ou relativizadas. No domínio da linguagem natural, uma situação equivalente seria a de um programa que utiliza classificadores para encontrar os termos mais significativos de um texto e classificá-los quanto ao seu significado, e baseado nos termos chave desse significado, encontrar outros objetos previamente analisados que resultem na mesma classificação exata. Ou seja, percebe-se que toda e qualquer estratégia de inferência é adicionada ao processo não para gerar o resultado final, mas sim para tentar reduzir o ruído existente nos dados da base e nas entradas do usuário e aumentar a probabilidade de se encontrar resultados pós-processamento exatamente iguais. Note-se que só é plausível o exposto caso os algoritmos usados tenham comportamento linear, como será explicado em seções posteriores deste capítulo.

Existem algumas vantagens na adoção desse paradigma intermediário de ferramenta de busca. Primeiramente, devido ao fato de que estratégias específicas de aplicação de aprendizado de máquina serão adotadas para cumprir apenas certas tarefas do processo de análise e transformação, se torna possível reduzir a complexidade geral da aplicação. Além disso, os resultados finais apresentados ao usuário tendem a ser quase sempre bastante precisos, visto que são frutos de transformações dos objetos da base e da própria entrada fornecida pelo usuário, contudo sem perder objetos relevantes que de outra forma seriam descartados por não terem uma completa semelhança com a entrada fornecida, conforme mostra o trabalho em (ALMEIDA et al., 2022). Na próxima sub-seção, será apresentado um projeto de um motor de busca de código aberto denominado *Fess* (CODELIBS, 2022a), o qual foi escolhido como parte integrante do sistema a ser

desenvolvido e testado nos experimentos descritos nesta dissertação.

2.2.4 Ferramenta de Busca *Fess*

Fess é o nome de uma ferramenta de busca de código aberto baseada em palavras-chave desenvolvida no Japão usando a linguagem Java. A sigla se origina do inglês em *Full Text Search Server*, trocando a letra ”t” pela primeira vogal na segunda palavra e juntado as demais primeiras letras na ordem, ou seja, servidor de busca de texto completo (CODELIBS, 2022b). Suas principais aplicações estão relacionadas à instalação numa infraestrutura própria para indexação de pastas e arquivos gerais e também à integração com sites, permitindo a indexação de páginas e a buscas de subpáginas usando palavras-chave.

Esta é uma aplicação de código aberto desenvolvida no Japão em linguagem Java com interface gráfica acessível através de uma página *web* disponível em rede. É uma solução completa de busca e indexação de arquivos e páginas *web* que pode ser instalada como uma aplicação Java. É uma ferramenta de busca nativamente baseada em palavras-chave.

Na seção seguinte, são descritos conceitos importantes sobre os processos de raspagem de dados, as etapas, os principais tipos e problemáticas, e ao fim, uma descrição de uma ferramenta bastante importante na raspagem de dados de páginas da Internet.

2.3 RASPAGEM DE DADOS

Raspagem de dados é um termo relacionado à tarefa de aquisição de informação gerada através de programas de computador (FERRARA et al., 2014). Essa tarefa, devido à dificuldade técnica, quantidade de tarefas manuais ou tempo necessário para completa realização, é atribuída a programas especializados.

Deve-se notar que raspagem de dados não é um termo genérico, e sim bastante específico e relacionado com fontes de informação de difícil acessibilidade estruturada, conforme visto em (SIRISURIYA et al., 2015). Ou seja, no evento de tentativa de cópia e estruturação automatizada de dados através de interfaces que não foram desenhadas para programas, e sim para humanos, verificar-se-á o uso de técnicas de raspagem de dados. É comum a necessidade de uso dessas metodologias quando as fontes de dados são arquivos em formatos que impedem edição, ou ainda documentos formados apenas por fotos, e mais comumente ainda, quando trata-se de aquisição automatizada de dados de páginas *web*. Em todos esses casos, e especialmente no último, a informação é normalmente visível e clara para um ser humano, porém não está disponibilizada para a consulta através de um programa de computador usando métodos e sistemas triviais. Dessa forma, se torna necessário ao interessado o uso de técnicas de análise de dados que emulam o todo, ou parte, do processo de consulta manual que seria feita caso um ser humano fosse o encarregado da tarefa.

Os trabalhos em (RAHMAN; TOMAR, 2020) e (MUEHLETHALER; ALBERT, 2021) comentam sobre técnicas de raspagem de dados de paginas da Internet, do inglês *webscraping*. Todavia, os esforços destes trabalhos se concentram em etapas isoladas do processo de investigação, havendo espaço para a proposição de métodos que não apenas extraiam dados, mas que os classifiquem e aglutinem com outros semelhantes, por exemplo.

Nas sub-seções seguintes, as etapas de um processo de raspagem de dados e os tipos principais de fontes são descritos. Ao fim, uma tecnologia específica que tem papel importante no experimento desta dissertação como meio de extração de dados da Internet é apresentada. De forma específica, na próxima sub-seção são apresentadas as três principais etapas de um processo genérico de ingestão de dados.

2.3.1 Extração, Transformação e Carga de Dados

A maturidade crescente e necessária das organizações em relação ao tratamento e integração de suas fontes de dados exigiu o nascimento de determinados modelos de trabalho, ou *frameworks*, que servissem de base para a evolução constante da diversidade de tipos e qualidades dos sistemas que geravam e disponibilizavam informação, conforme (MUKHERJEE; KAR, 2017). Ou seja, para que essas organizações pudessem se adequar às inovações que chegassem e ainda manter um mesmo fluxo de trabalho com dados, que não exigisse novos treinamentos e mudanças constantes em toda a sua infraestrutura de computação, o *framework* ETL, acrônimo da sequência de termos em inglês *Extract, Transform, Load*, se popularizou bastante durante os anos de 1990 (MUKHERJEE; KAR, 2017). Nessa simples sequência de trabalho, os dados de diferentes fontes de informação deveriam ser extraídos como primeiro passo, transformados para um ou mais modelos previamente definidos, e por fim carregados nos sistemas de armazenamento, culminando num conceito discutido anteriormente, o armazém de dados, ou do inglês *Data Warehouse* (INMON, 1995).

Esse eficiente método de trabalho garantiu que, mesmo com fontes de dados completamente desacopladas e independentes, as organizações ainda seriam capazes de criar fluxos de trabalhos contínuos e independentes de inovações e mudanças drásticas nas fontes. Um exemplo poderia ser um banco, que no passado guardava dados digitados por um atendente sobre uma transação realizada numa agência de forma física, depois passou a armazenar dados vindos de caixas eletrônicos e terminais de autoatendimento, e por fim ainda teve de se reinventar para guardar dados de transações feitas através de páginas *web* de autoatendimento acessadas por meio de computadores pessoais nas casas de seus clientes. Independentemente de todas as variações possíveis na forma e regra de trabalho para cada etapa histórica e atual (visto que as três possibilidades ainda existem simultaneamente), seguindo o *framework* de extração, transformação e carregamento, esse banco foi provavelmente capaz de manter o ritmo de negócio e os processos de dados, pois a mudança de fonte e característica do dado bruto implica apenas na mudança das etapas de extração e na de transformação, mas após o carregamento, a lógica negocial é mantida não importa a forma. Portanto, no exemplo, em qualquer das três situações, o banco registrará de forma única as qualidades

fundamentais de uma transação, ainda que dependa de processos diversos para isso.

Na próxima sub-seção, são detalhados conceitos sobre a ingestão de dados advindos de documentos.

2.3.2 Conceitos Iniciais e Raspagem de Dados em Documentos

A informação armazenada em um banco de dados passível de consulta não será alvo de raspagem de dados. Complementarmente, os dados disponibilizados de forma desordenada e muitas vezes consultáveis apenas visualmente em diferentes formatos de arquivos, esses sim, serão objetos de análise durante um processo de raspagem de dados (DIOUF et al., 2019).

Entre esses tipos, um dos formatos que mais comumente gera objetos de processos de raspagem é o PDF, ou do inglês *Portable Document Format*, criado pela empresa Adobe (ADOBE, 2022). Esse formato foi desenhado com o intuito de prover uma versão digital e fiel que pudesse substituir um documento físico (ADOBE, 2015). Esse seria um passo intermediário na digitalização das relações de trabalho, pois se ainda não havia maturidade tecnológica suficiente no mundo para substituir os documentos por formulários digitais e interfaces responsivas, o formato PDF provia uma forma de digitalizar fielmente o que estaria no papel, inclusive limitando e/ou impedindo a edição de um documento após sua criação. O primeiro padrão ISO (do inglês *International Organization for Standardization*, ou Organização Internacional para Padronização) para esse formato está disponível em (32000-1, 2008), tendo sido lançado no ano de 2008. Adicionalmente, o PDF ainda permitia a digitalização de documentos físicos preexistentes. Todavia, o formato PDF não foi criado como um documento que podia ser lido de forma trivial e estruturada, instrução por instrução, por um software. Na verdade, o uso desse formato tinha como objetivo a criação de um modelo de descrição de instruções que guiassem a renderização livre de cada página do documento, ou seja, o formato e disposição dos dados só é passível de interpretação visual após a renderização e, mesmo nesse ponto, como a formatação visual é livre, interpretar um documento PDF acaba se tornando semelhante a reconhecer figuras em uma imagem do ponto de vista de uma máquina. Esses detalhes se tornaram ainda mais relevantes nas demais atualizações do padrão, a exemplo da versão 2.0 lançada em 2017, com vários suportes a elementos gráficos. Esse formato segue um padrão internacional e a última versão da referência é o documento ISO 32000-2:2020 (32000-2, 2020).

No começo da digitalização de documentos, havia pouco poder de processamento computacional disponível, especialmente para o estudo de grandes massas de documentos, sendo esse um mercado novo e em expansão, conforme pode ser visto no trabalho em (FRIEDMAN, 1998) publicado no ano de 1998. Entretanto, dois fenômenos convergiram para a atualidade: a popularidade dos documentos em formato PDF se firmou como padrão preferível para diversos processos e, ao mesmo tempo, o interesse e a necessidade de expansão das possibilidades provenientes dos armazéns de dados demandou a busca de novas técnicas de extração de dados de fontes não tão triviais como imagens e documentos em formato PDF. Uma forma encontrada foi a renderização do arquivo e a interpretação, em tempo de execução, das partes renderizadas. Essa técnica

é usada por ferramentas como Apache Tika (ORG, 2022b) e PDFMiner (SHINYAMA, 2022). Outra técnica bastante famosa é a chamada OCR, do inglês *Optical Character Recognition*, ou Reconhecimento Óptico de Caracteres, que em resumo, se trata do uso de aprendizado de máquina (e algumas vezes de Inteligência Artificial) para o reconhecimento de padrões em gráficos que remetam a um determinado caractere ou palavra/expressão, e assim, parte por parte, a máquina vai desvendando o que está escrito numa figura. Mais informações sobre a técnica podem ser revisadas no trabalho em (MEMON et al., 2020).

Existem ainda diversos outros tipos de dados e formatos a serem discutidos, porém que fogem ligeiramente ao escopo deste trabalho, que está focado especialmente em fontes de dados da Internet, ou seja, páginas *web*. Esta seção discutiu as noções básicas de extração não trivial de dados em documentos para dar suporte às técnicas que serão discutidas para páginas da Internet na seção seguinte. Na próxima sub-seção, são detalhados detalhes importantes da ingestão de dados da Internet e uma distinção especial é feita entre as tecnologias de integração de sistemas e os processos chamados, no inglês, de *webscraping*.

2.3.3 Evolução de Conceitos e a Raspagem de Dados da Internet

Raspagem de dados de páginas da Internet, ou do inglês, *webscraping*, é um termo bastante usado na atualidade e se relaciona com o uso de técnicas especiais de extração de dados de páginas *web* usando programas de computador e automações de software, existindo uma revisão completa sobre os conceitos e aplicações em (SINGRODIA; MITRA; PAUL, 2019). Assim como descrito anteriormente sobre os documentos em PDF, a mesma lógica se aplica neste cenário: quando as tecnologias de renderização de páginas na Internet foram desenhadas, por questões de design e eficiência no uso de recursos de processamento, e com o foco na finalidade, que era velocidade de carregamento e liberdade de desenho e detalhamento visual, questões como a estruturação dos dados para a consulta automatizada foram deixados em segundo plano, e essas características podem ser revistas no trabalho descrito em (WEIDEMAN; SCHWENKE, 2006). Esse conceito sofreu grande expansão, inclusive existindo empresas hoje que desenvolvem técnicas especiais e regras de uso das páginas para impedir e/ou atrapalhar ao máximo seus usuários a usarem programas para interagir com seus serviços providos através de interfaces gráficas em páginas *web*, existindo complexas discussões a respeito da legalidade completa da atividade, como mostra (KROTOV; JOHNSON; SILVA, 2020).

As páginas *web* foram inicialmente desenhadas para serem renderizadas com base num conjunto de dados recebidos de um servidor remoto, que seria onde estariam armazenadas essas instruções. A linguagem escolhida para tal finalidade foi o HTML, do inglês *Hyper Text Markup Language*, ou Linguagem de Marcação de Hipertexto, existindo uma boa descrição do funcionamento geral no padrão RFC (do inglês *Request for Comments*, ou solicitação de comentários, nome comumente dado a padrões para a Internet) de número 9110 (9110, 2022). Sua principal característica é por definir uma estrutura mínima fixa com diversos comandos ordenados de tal forma que um programa de computador desenvolvido com a finalidade de interpretá-los consiga

traduzir as instruções em elementos visuais e com ações programáticas ocorrendo de forma relacionada à interação do usuário, como a chamada de uma função remota no servidor atrelada ao clique de um botão, ou a mudança de cor de um elemento quando o mouse passa por cima deste. Os programas responsáveis por essa interpretação são os navegadores, ou comumente denominados *browsers* no inglês.

O HTML evoluiu bastante, alcançando a versão 5 de forma acordada e bem documentada no ano de 2019 conforme documentação pública disponível e regularmente atualizada até a data deste trabalho em (COMMUNITY, 2022). Contudo, desde o início percebeu-se a necessidade de recursos adicionais para a renderização condicional e assíncrona de elementos, características complexas de serem trabalhadas com as instruções nativas do HTML. Ou seja, com o paradigma clássico do HTML, boa parte dos componentes tinha de estar disponível em memória para serem vistos e passar por interações do usuário, e a quantidade de funções e possibilidades era limitada. Assim, surgiu em 1995 uma outra linguagem que apoiaria e automatizaria a criação dessas instruções: o Javascript, com histórico e detalhes disponíveis em (ORG, 2022g). Essa linguagem, que no início não foi criada especificamente para um único navegador *web*, conforme (WIRFS-BROCK; EICH, 2020), veio como uma espécie de camada adicional durante a renderização de páginas, tornando-se popular no início dos anos 2000. Isso porque se antes o servidor precisava enviar todo o código HTML que o navegador iria usar para cada página, com o Javascript, esse mesmo servidor conseguia enviar códigos muito menores, deixando que o trabalho de tradução do código Javascript para HTML fosse feito pelo navegador. Além disso, os profissionais passaram a conseguir desenvolver com agilidade novas funcionalidades, visto que com o uso da linguagem, conjuntos complexos de comportamentos assíncronos podiam ser descritos em poucas linhas com código Javascript e esse código, ao ser enviado para o computador de um usuário, era interpretado localmente e guiava a montagem e renderização das instruções HTML, que por sua vez serviam de base para a formação visual da página e de suas funções.

Dado, portanto, que seja usando HTML, seja usando Javascript, é necessário compreender códigos e padrões para ser capaz de extrair informação textual e visual de páginas da Internet, adicione-se ainda o fato de que existem páginas extremamente complexas com códigos assíncronos em Javascript realizando diversas rotinas internas à memória, e modificando o comportamento e conteúdo dos elementos visuais de uma página em tempo real, e tem-se a receita para um grande desafio computacional ao desenvolver um programa que seja capaz de extrair fielmente as informações úteis e específicas dessas fontes de dados na *web* (THOMAS; MATHUR, 2019). Fica claro, portanto, que ao se referir à raspagem de dados da Internet, não se está fazendo alusão à consulta de de uma API, do inglês *Application Programming Interface*, ou interface de programação de aplicação, que seria uma interface especificamente preparada para que outros programas se conectem e recebam informações em formatos estruturados e passíveis de processamentos usando lógicas triviais, mas sim, a alusão está sendo feita ao processo complexo de extração de informação útil de páginas da Internet que são renderizadas com base em códigos muitas vezes de execução assíncrona e recebidos por meio de APIs.

A extração de dados de páginas descritas puramente por código HTML é bastante facilitada

na atualidade devido aos esforços constantes para listagem e documentação de todas as instruções descritas pelos padrões HTML conforme citado anteriormente. Para tanto, basta carregar o código HTML e realizar-se buscas simples pelos trechos corretos onde a informação visual estará descrita, sem a necessidade de renderização dos componentes. Já para o caso em que existe código Javascript, devido ao fato de que esse é usado para guiar a montagem do código HTML em tempo de execução, é necessário um programa que efetivamente interprete esse código e renderize as instruções HTML que descrevem a página (DEURSEN; MESBAH; NEDERLOF, 2015). Ou seja, é uma situação bastante parecida com a descrita anteriormente em relação aos documentos em formato PDF: é necessário que um programa emule todo o processo que seria executado por um ser humano, não havendo atalhos que possam ser tomados, como no caso do HTML puro, em que basta compreender a sintaxe e buscar pelos conteúdos nas seções corretas do código. De fato, as páginas *web* se comportam como documentos que são servidos por APIs para o usuário final, e esses documentos especiais precisam ser interpretados, e a diferença primordial decorrente da evolução das funcionalidades dessas páginas é o fato de que a pura interpretação já não é suficiente, havendo a necessidade de renderização e execução de código para gerar, previamente, e tornar disponível a informação a ser extraída.

Além da renderização, outro método existente também está relacionado com o reconhecimento de padrões em imagens, o OCR, como descrito anteriormente, em que um programa analisa porções gráficas de páginas da Internet e tenta extrair informação útil. Um último detalhe, no entanto, é que não existe outro método de extração de dados mais completo que a renderização devido à necessidade intrínseca de interação com as páginas. Em outras palavras, é comum que para que a informação útil se torne disponível, seja necessário interagir com botões e campos de texto de páginas *web*, e por causa disso, apenas um programa que tenha renderizado todos os componentes e consiga emular as ações de usuários e acionar as sequências de códigos assíncronos correta será capaz de extrair o mesmo dado que um ser humano faria manualmente. Vê-se que a extração de dados em páginas da Internet pode se tornar um trabalho bastante complexo e repleto de etapas, e é por isso que ferramentas especiais foram criadas para esse fim, como será apresentado na próxima sub-seção.

Na próxima sub-seção, é apresentada uma tecnologia extremamente importante e que permite que sistemas automatizados interajam com páginas *web* e possam realizar leituras, ações e navegações tal qual um ser humano faria.

2.3.3.1 *Selenium Grid*

Entre as ferramentas disponíveis, uma tem se destacado pelo suporte a diversas linguagens e flexibilidade no uso: Selenium (ORG, 2022k). Essa ferramenta de código aberto consegue interagir com navegadores e emular todas as interações de usuário com os elementos de uma página, além de prover acesso a todo o código HTML renderizado, não importando se há ou não Javascript envolvido.

Adicionalmente, a ferramenta Selenium permite a renderização de páginas *web* em qualquer

plataforma através de emulação, ou seja, é possível testar, por exemplo, a visão de um usuário através de dispositivos móveis, como celulares, sem a necessidade real do aparelho físico. O chamado Selenium Grid (ORG, 2022m) é a versão distribuída da ferramenta, permitindo que seja montado uma rede de nós independentes em que cada um pode receber requisições de renderização e extração de dados de páginas, inclusive de plataformas variadas, e tudo através de apenas um nó mestre.

No *Selenium Grid*, cada nó especialista tem um Sistema Operacional e atua emulando as operações numa página *web* em um navegador específico. Ou seja, o programa recebe os comandos do nó mestre para acessar uma página, rolar uma tela, e até mesmo clicar em um botão, e os executa internamente. O nó mestre, por sua vez, recebe os comandos e fluxos de extração e administra a carga entre os nós especialistas. Cada nó executa um fluxo inteiro de extração e caso não existam nós disponíveis, o nó mestre Selenium Hub forma uma fila até que findem os processamentos vigentes. Isso ocorre porque o processo de interação com páginas *web* é custoso computacionalmente e dependente de dados de sessão, que não podem ser compartilhados entre diferentes navegadores. Também porque cada nó é especialista em emular um determinado tipo de sistema. Todas essas informações e outros detalhes podem ser obtidos na documentação oficial do projeto (ORG, 2022l).

Na próxima seção, o processamento de linguagem natural, comumente denominado pela sigla NLP, será abordado, dando destaque para algumas técnicas de transformação necessárias e ao fim, detalhando uma biblioteca da Linguagem Python usada no experimento apresentado nos próximos capítulos desta dissertação.

2.4 PROCESSAMENTO DE LINGUAGEM NATURAL

Diferentemente dos casos de aplicação de métodos computacionais para resolução de problemas de natureza puramente matemática, o processamento de termos, expressões, palavras isoladas e textos exige transformações e conceitos adicionais para tornar-se viável, segundo (THANAKI, 2017). Se nos caso de uma regressão linear simples, os valores numéricos são conhecidos a priori e serão processados tal qual sua formatação original, no processamento de linguagem natural, as palavras precisam passar por etapas de pré-processamento para assumir novas representações que contenham de forma mais explícita os significados numéricos e, portanto, possa ser analisadas usando métodos numéricos clássicos.

Os trabalhos em (UKWEN; KARABATAK, 2021) e (AMATO et al., 2019) discorrem sobre as ferramentas e propõem soluções e métodos úteis à forense digital baseados em processamento de linguagem natural. Contudo, há espaço para melhoria na exploração da auditabilidade e transparência dos relatórios, além de pouco foco na fundamentação analítica das teses apoiadas pelas plataformas, ou seja, no afastamento da subjetividade.

Nas sub-seções a seguir, são apresentados conceitos sobre vetorização, transformações e ou-

tras técnicas que permitem aos sistemas digitalizados trabalhar com dados de linguagem natural. Ao fim, uma biblioteca importante da Linguagem Python e que constitui parte do experimento a ser apresentado nesta dissertação nos demais capítulos será apresentada.

2.4.1 Vetorização

Palavras, ou conjuntos de palavras, podem conter vários significados, e é uma tarefa complexa para uma máquina entender e listar todas as possibilidades imagináveis, visto que existe um grande universo de combinações possíveis. Entretanto, existe uma estrutura matemática que também guarda vários significados em uma única instância e já tem toda uma história de sucesso no mundo do processamento computacional: os vetores. A vetorização é passo importante no processamento de linguagem natural pois traduz elementos complexos como palavras para um domínio matemático separado em dimensões com parâmetros claros e características mensuráveis, tornando viável e extremamente eficiente o processamento de grandes massas de textos, ainda segundo (THANAKI, 2017). Além disso, existe uma vantagem adicional: vetores são preferíveis para aplicações de aprendizado de máquina e inteligência artificial, pois boa parte desses algoritmos trabalham com modelos orientados a números e transformações matemáticas.

Na próxima sub-seção, algumas das principais técnicas de vetorização de dados são apresentadas.

2.4.2 Principais Técnicas de Vetorização

Existem diversas técnicas de vetorização, geralmente relacionadas com a frequência de termos num universo finito de palavras (como um documento ou texto), ou ainda com algum tipo de transformação em relação a um dicionário. Uma visão geral dessas e outras técnicas pode ser vista em (PROKHOROV; SAFRONOV, 2019). A seguir, algumas das principais técnicas são explicitadas brevemente:

1. Método Saco-de-Palavras (do inglês *Bag of Words*): Em resumo, esse método baseia-se na criação de uma lista de palavras relevantes e únicas, separação do texto em porções (por exemplo em frases), e a criação de vetores que em suma indicam a ocorrência ou não de cada palavra da lista.

2. Método TF-IDF, ou Termo Frequência-Frequência Inversa de Documento (do inglês *Term Frequency–Inverse Document Frequency*): Essa técnica leva em conta, também, a frequência, ou quantidade de ocorrências, de um termo para decidir se ele é importante, entretanto, existe um peso inverso para o caso de palavras que sejam muito frequentes. Em resumo, o algoritmo tenta encontrar um ponto ótimo, dentro do universo de documentos participantes da análise, em que as palavras mais relevantes são listadas, mas sem necessariamente considerar apenas as mais frequentes para a tarefa. Baseado nessa lista, os vetores são formados conforme a ocorrência dessas palavras nas sentenças.

3. Método Word2Vec: Esse método apresentado pela empresa Google (GOOGLE, 2022), diferentemente dos anteriores, não se baseia em frequência, mas sim, no uso de redes neurais simples para agrupar termos os quais uma máquina faça inferências sobre seus significados em relação às ocorrências nos documentos.

4. Método GloVe: Segue uma abordagem semelhante à do método Word2Vec, contudo, o processo de aprendizagem é guiado pelo contexto completo, ou seja, estatísticas globais, e não apenas locais, ajudam a calibrar os vetores e os agrupamentos. Desenvolvido na Universidade de Stanford, nos Estados Unidos (STANFORD, 2022).

5. Método FastText: Enquanto Word2Vec e GloVe focam em melhorar os agrupamentos das palavras, o método FastText, apresentado pela empresa Facebook (FACEBOOK, 2022), diminui a granularidade, formando agrupamentos de caracteres que, quando relacionados, formam os vetores das palavras.

Na próxima sub-seção, a biblioteca contendo os métodos de vetorização usados no experimento desta dissertação é apresentada em detalhes.

2.4.2.1 Biblioteca Scikit-Learn

A biblioteca Sklearn (ORG, 2022o), resultado do projeto Scikit-Learn, é uma das principais ferramentas da atualidade no estudo, exercício e implementação de técnicas de aprendizado de máquina usando código aberto (KRAMER, 2016). A biblioteca escrita em linguagem Python contém diversos módulos para automatizar etapas do treinamento e avaliação de modelos de aprendizado de máquina, inclusive também conta com funções de pré-processamento e vetorização, e entre essas, o pacote TfidfVectorizer (ORG, 2022p), o qual foi usado no projeto descrito neste documento como implementação da técnica de vetorização TF-IDF.

Na seção seguinte, conceito importantes sobre aprendizado de máquina serão expostos de forma a viabilizar o completo entendimento desta dissertação e das motivações sobre as tecnologias e o experimento conduzido e apresentado nos capítulos seguintes.

2.5 APRENDIZADO DE MÁQUINA

Aprendizado de máquina é o nome de um campo da computação que estuda o uso e otimização de algoritmos que imitam o processo de aprendizado de um ser humano para resolver problemas, na maioria das vezes, de natureza matemática (ou descritos de forma matemática) (NAQA; MURPHY, 2015). Isso significa que a definição completa desse campo ainda está em construção, visto não existir uma teoria comprovada sobre como o processo cognitivo funciona e evolui (CONWAY, 2020). No entanto, independente das comprovações sobre a forma como os seres humanos aprendem, como visto no trabalho citado, existem diversas semelhanças nesse processo com a tendência de convergência por iterações de vários algoritmos numéricos.

De forma geral, o objetivo é resolver problemas de forma iterativa, ou seja, repetindo-se uma sequência de passos que, a cada iteração completa, chegam mais perto da solução. Isso também significa que não se busca, em geral, a solução perfeita, mas sim, uma solução ótima, que tenha a maior eficiência possível dado o modelo de treinamento usado e o problema em si (VU et al., 2018).

Em relação às estratégias de treinamento, existem três principais categorias, segundo (JORDAN; MITCHELL, 2015):

1. Aprendizado Supervisionado: Nessa técnica, o treinamento é feito pautado numa base de dados previamente fornecida e com exemplos, também, previamente revisados. Dessa forma, a máquina vai usar esses dados como bases para, iterativamente, aprender. Um exemplo é uma máquina que aprende a classificar imagens de carros através de um banco de imagens de carros fornecido previamente.

2. Aprendizado Não Supervisionado: Nessa técnica, a máquina tentará, sozinha, encontrar um ponto ótimo, ou seja, não existirão dados previamente preparados, e o algoritmo será responsável por convergir a uma resposta adequada. Esse método de aprendizado é marcado pela busca por padrões que possam ser aproveitados. Um exemplo é a regressão linear, em que um algoritmo precisa encontrar um ponto médio entre várias amostras de dados, e não existe nenhuma referência previamente preparada para servir de base para a tarefa.

3. Aprendizado por Reforço: Nesse tipo de aprendizado, a máquina irá fazer testes do que considera uma resposta correta ao problema, e cada vez que perceber que errou, irá repetir suas iterações até reduzir ao máximo os erros. Um exemplo seria uma máquina que aprende a jogar e vencer um jogo qualquer. A cada tentativa falha de prosseguir, a máquina leva em conta essa e as outras falhas antes de tentar novamente.

No trabalho em (QADIR; VAROL, 2020), tem-se uma revisão sobre o papel do aprendizado de máquina na Ciência Forense digital. Já os trabalhos em (BHATT; RUGHANI, 2017) e (TALLÓN-BALLESTEROS; RIQUELME, 2014) focam em implementações e questões técnicas da aplicação de metodologias da Ciência de dados no campo da investigação de crimes digitais. Todavia, em todos ainda há pouco foco na questão da auditabilidade e na discussão sobre a construção de um conceito de jurisprudência aplicado à investigação forense. O presente trabalho tenta preencher esse espaço ao propor métodos e técnicas que embasem não apenas decisões isoladas, mas sirvam de referência facilmente legível para embasar futuras decisões.

Nas sub-seções a seguir, serão expostos brevemente alguns conceitos sobre linearidade e comportamento dos algoritmos de aprendizado de máquina, além da breve apresentação de alguns dos principais modelos utilizados em tarefas de classificação.

2.5.1 Linearidade dos Algoritmos de Aprendizado de Máquina

Existem duas possibilidades para algoritmos em relação à linearidade, segundo (L'HEUREUX et al., 2017):

1. Algoritmos Lineares: Nesses, assume-se que existe uma função que é capaz de traduzir os dados de entrada para os de saída de forma direta e linear, e portanto, basta encontrar essa relação. Ou seja, o resultado é resultado de uma combinação linear das entradas.

2. Algoritmos Não Lineares: No caso de algoritmos não lineares, assume-se que a relação entre os dados de entrada e os de saída, por exemplo, uma classificação, podem assumir qualquer nível de complexidade e, portanto, provavelmente não podem ser descritos por uma relação direta.

Na sub-seção a seguir, os principais modelos disponíveis de algoritmos de aprendizado de máquina, especialmente aqueles relacionados ao processamento de linguagem natural, alvo das discussões presentes nesta dissertação.

2.5.2 Principais Modelos

Além das considerações sobre a linearidade e sobre as estratégias de treinamento, existem vários algoritmos de aprendizado de máquina. Esses algoritmos são espécies de modelos matemáticos generalistas que descrevem a forma de cálculo e os passos a serem executados para a resolução de um problema, ou seja, para a predição, como é comumente designado num algoritmo de aprendizado de máquina. Por existirem diversos modelos de classificação de linguagem natural, para diversas aplicações, e dado que o trabalho exposto nesta dissertação estará relacionado ao caso multiclasse da análise de sentimento, serão relacionados a seguir quatro dos principais modelos atualmente usados para esse fim (todos supervisionados), segundo (SINGH; THAKUR; SHARMA, 2016):

1. Regressão Logística: Também chamada de modelo logit, é um algoritmo que assume que a predição resultante será dependente de uma ou mais variáveis de entrada, e tenta encontrar a probabilidade da saída. Assume uma relação linear entre as entradas e saídas.

2. Floresta Aleatória: Esse algoritmo usa o conceito de árvores de decisão, no entanto, ao criar várias árvores com regras mais aleatórias e menos estritas e calcular a média de todas, assume-se que a predição final tenderá a estar balanceada e eficiente.

3. Naive-Bayes: É um modelo probabilístico que assume que as variáveis a serem consideradas nos dados de entrada são independentes e contribuem individualmente para cada classe a ser considerada na predição. É baseado no Teorema de Bayes.

4. Máquinas de Vetor Suporte: Também comumente denominadas de SVM, do inglês *Support Vector Machines*, esse algoritmo busca calcular hiperplanos que dividam os dados em duas

classes distintas. Em duas dimensões, um hiperplano seria uma linha reta, em três dimensões, seria uma área, um plano, por exemplo. A biblioteca SKLearn, descrita anteriormente na seção sobre vetorização, também disponibiliza um modelo de SVM, e foi exatamente esse que baseou a implementação utilizada neste trabalho, conforme será explicado nas seções posteriores e principalmente no capítulo 3.

Na seção a seguir, alguns detalhes importantes sobre enriquecimento de dados serão apresentados e que contribuem para o total entendimento dos objetivos desta dissertação.

2.6 ENRIQUECIMENTO DE DADOS

É comum que os dados obtidos interna ou externamente numa organização careçam de detalhes cruciais para as aplicações desejadas. Esses dados brutos podem, portanto, passar por processamento visando o cruzamento de informações para enriquecê-los, conforme (XUE; WU; LU, 2021) aplica o conceito na área de enriquecimento semântico de modelos de informação para grandes infraestruturas. Um exemplo de enriquecimento de dados está presente quando um banco usa dados de seus clientes para montar perfis de risco usando informações de terceiros, quando permitido em lei, ou também quando faz a relação usando dados de outros setores internos, como de cartões de crédito e consórcios. Ou seja, enriquecer os dados, em geral, é uma técnica de aglutinação de informação para que o valor agregado dessas massas de dados se torne mais útil ao negócio. Esse enriquecimento pode ser fruto de cruzamentos diretos de informações, como quando um número único de cadastro é rastreado entre diferentes bases, ou pode ser fruto de inferências, como quando um algoritmo tenta encontrar registros que pertençam à mesma pessoa usando como base nomes incompletos e endereços de *email*, os quais não têm, num primeiro momento, garantia de associação com a pessoa.

Esse processo é semelhante ao que ocorre em processos de investigação, pois os cientistas recebem uma massa de informações desorganizadas e precisam encontrar dados que se relacionem e aumentem a confiabilidade de uma tese. Ou seja, uma tese deve ser aglutinada a outras pequenas teses relacionadas a provas e evidências que contribuam para a clareza e veracidade, ou falta dela, nas acusações.

De fato, a literatura sobre enriquecimento de dados aplicado à forense digital é quase inexistente, todavia, são numerosos os trabalhos que utilizam desses conceitos para agregar valor em outros setores como o de redes sociais e análises que visem compreender fenômenos complexos. Os trabalhos em (AGARWAL; TOSHNIWAL, 2019), (FOX et al., 2021) e (GUO et al., 2013) exploram técnicas de enriquecimento de dados de redes sociais para a adição de valor em análises complexas relacionadas à correlação entre eventos internos às plataformas sociais e relações externas entre os atores dessas redes. O trabalho em (BISCHKE et al., 2016) propõe o enriquecimento de dados obtidos através de imagens de satélites para monitoramento remoto de eventos. Já o trabalho disposto em (PREECE et al., 2018) propõe e apresenta uma plataforma de-

senvolvida colaborativamente para enriquecimento semântico de conteúdo obtido através de redes sociais. Todos esses trabalhos denotam a importância e relevância dos esforços para enriquecer amostras de dados isoladas e como essas técnicas adicionam valor a cada uma das indústrias. O presente trabalho tenta aplicar essa mentalidade à Ciência forense digital como forma de trazer auditabilidade às teses de investigação.

Na seção a seguir, algumas diferenciações importantes sobre arquitetura de aplicações são apresentadas e explicitadas, e ao final, a arquitetura orientada a serviços, aplicada no trabalho desta dissertação, é detalhada.

2.7 ARQUITETURA DE APLICAÇÕES ORIENTADA A SERVIÇOS

A construção e projeto de aplicações modernas exige o conhecimentos de diversas disciplinas, formando um campo de estudo por si só que foge ao escopo desta dissertação, porém uma visão geral pode ser vista em (VALIPOUR et al., 2009). Dessa forma, cada tipo de negócio a ser resolvido por uma aplicação pode ser modelado de forma genérica e se beneficiar de padrões de projeto de software já estudados e validados. Entre esses padrões, existe uma classe importante relacionada principalmente às aplicações de software distribuído, ou seja, técnicas de projeto e implementação de componentes de software que funcionam com base em elementos desacoplados física ou logicamente e que comunicam-se entre si para atingir o objetivo de negócio desejado conforme (BAL; STEINER; TANENBAUM, 1989). Nessa arquitetura, a aplicação é dividida em serviços que devem ser providos por componentes isolados entre si. Um exemplo simples ocorre naturalmente ao se analisar a estrutura de negócio de um cartório brasileiro (conforme (BLASKESI, 2019)):

1. Cidadãos que precisam de um determinado serviço, como o reconhecimento de uma assinatura, podem se dirigir a um posto de atendimento de um cartório.

2. Ao se dirigir a um atendente, este não é o responsável pelo reconhecimento em si, e sim apenas pelo atendimento ao cidadão. O atendente irá, então, verificar se o cidadão tem cadastro naquele cartório, e se o tiver, irá encaminhar o documento com a assinatura para um setor responsável pela comparação e confirmação, ou não, da semelhança entre a assinatura no documento e os registros prévios que tenham. Caso não tenha cadastro, o atendente possivelmente irá colher os dados do cidadão, sua assinatura, e encaminhá-los a outro setor responsável por novos cadastros, para somente depois encaminhar o documento ao setor de reconhecimento.

3. Quando o setor responsável pelo reconhecimento terminar de analisar a assinatura, irá devolver o documento ao atendente com o resultado da análise.

4. O atendente irá, então, expedir um pedido de cobrança pelo serviço e pedirá ao cidadão que se encaminhe ao setor responsável por receber pagamentos.

5. Após o setor que recebe e processa pagamentos confirmar a transação, o cidadão poderá voltar ao atendente e resgatar seu documento com algum selo atestando, em caso de sucesso, que a assinatura foi reconhecida.

O exemplo ilustrado acima deixa claro que a arquitetura dividida em serviços visa isolar as dependências e responsabilidades entre os módulos de um negócio, ou seja, entre os serviços fundamentais. O serviço de interesse do cidadão, o reconhecimento de assinaturas, na verdade, é uma composição de outros serviços internos à entidade e que funcionam de forma independente, porém orquestrada, ou estruturada.

De forma complementar, a arquitetura orientada a microsserviços também existe e diferencia-se especialmente na complexidade de cada serviço. Nessa, cada elemento deve ser especialista em um única e bem definida tarefa (LI et al., 2021). Apesar de ser bastante usado e difundido como método preferível no projeto de novos sistemas, ainda é um desafio aplicar esses conceitos em projetos que demandam bastante processamento e atingem alta complexidade computacional (KALIPE; BEHERA, 2019).

Os trabalhos em (SIBIYA; VENTER; FOGWILL, 2015), (AMATO et al., 2020) e (REDDY; VENTER, 2013) propõe, em diferentes níveis, a adoção de arquiteturas de aplicação e *frameworks* de trabalho que organizem as investigações forenses. Seguindo a mesma lógica, o presente trabalho propõe uma ferramenta de busca como ferramenta fim a fim para suportar investigações forenses de crimes digitais, ou seja, um software capaz de extrair, organizar, classificar e armazenar as evidências coletadas. Todavia, de forma diferente dos demais trabalhos citados, a plataforma proposta é capaz de ser adaptada para diferentes categorias de crimes e indústrias, sendo flexível o bastante para atuar com várias fontes de dados na Internet e também para ser treinada e realizar classificações embasadas em uma base de dados qualquer, trazendo adaptabilidade e liberdade para os investigadores.

Na seção seguinte, breves detalhes são apresentados sobre as aplicações em contêiner e a importância para o trabalho presente nesta dissertação.

2.7.1 Contêineres

Contêineres são ambientes isolados e minimalistas, na maioria das vezes, em que uma determinada aplicação é executada conjuntamente com todas as suas dependências e recursos necessários (PAHL; LEE, 2015). A principal característica se refere à mobilidade e escalabilidade, ou seja, aplicações em contêiner devem ser facilmente replicáveis e reproduzíveis em diferentes plataformas e sistemas operacionais, pois em tese, tudo o que é necessário para que a aplicação seja executada está presente dentro do ambiente isolado do contêiner, e o trabalho de tradução entre o Sistema Operacional e esse contêiner fica a cargo de uma outra camada de software externa e independente do contêiner em si.

Essa tecnologia é especialmente importante quando considerada no contexto de aplicações com arquitetura orientada a serviço, pois facilita a implementação e manutenção, além de trazer

mais liberdade na escolha das tecnologias e na replicação de serviços. Cada contêiner, de forma natural, reproduz os principais preceitos de desacoplamento de um serviço (AHAMAD, 2021).

Na sub-seção a seguir, uma tecnologia de código aberta de contêineres é apresentada, a qual foi utilizada no trabalho descrito nesta dissertação, como será exposto nos demais capítulos.

2.7.2 Docker

Docker (DOCKER, 2022a) é o nome de uma das mais populares tecnologias de contêiner. Sua base de código é aberta e recentemente introduziu a aplicação multiplataforma Docker Desktop, a qual tem suporte para os principais sistemas operacionais e permite a interação, execução e administração de aplicações em contêiner usando interfaces gráficas simples. Essa pilha de software se utiliza de funções nativas do Sistema Operacional Linux (ORG, 2022f) para administrar o uso de recursos entre aplicações rodando em contêineres isolados.

Além da interface gráfica e da base de código que define contêineres de forma padronizada, Docker também permite a descrição de novos projetos usando outras ”receitas” de contêineres já validadas anteriormente, ao que chama de imagens (DOCKER, 2022d). Ou seja, a partir da imagem de um contêiner que roda uma aplicação, é possível, através de arquivos chamados de *Dockerfile*, descrever um novo contêiner que usa esse projeto anterior como base e acrescenta outras aplicações e comandos. Esses projetos e imagens também estão disponíveis de forma aberta em repositórios públicos, sendo o mais proeminente deles atualmente o DockerHub (DOCKER, 2022c). Há ainda uma aplicação adicional, construída como uma espécie de *plugin* para o projeto original, denominada "Docker Compose", que agiliza a criação de arquiteturas distribuídas de aplicações e a descrição em alto nível de projetos. Essa aplicação usa o padrão YAML, do inglês *Yet Another Markup Language*, ou ’Ainda Outra Linguagem de Marcação’ (ORG, 2022s), para descrever arquiteturas de contêineres, redes de comunicação interna, protocolos, mapeamento de arquivos e pastas entre o sistema hospedeiro e os contêineres e até mesmo a execução de comandos de inicialização (como a definição de variáveis de ambiente) (DOCKER, 2022b). Essa tecnologia foi usada e é a base da descrição da arquitetura da aplicação a ser descrita no capítulo 3 desta dissertação.

Na seção a seguir, são explorados conceitos sobre a legalidade das provas e do processo de investigação que culminam nos requisitos básicos a serem atingidos com a aplicação dos conceitos até aqui apresentados.

2.8 APLICAÇÃO NA FORENSE DIGITAL

A leitura atenta do artigo 156 do Código do Processo Penal Brasileiro (BRASIL, 1941a) mostra que existe importância fundamental no uso de técnicas investigativas para apuração de dúvidas relacionadas a fatos relevantes. Também, em (BARROS et al., 2021), é possível ver

uma listagem de requisitos objetivos que garantem a qualidade dos resultados entregues por um processo de investigação forense, quais sejam:

1. Técnica autenticada;
2. Titulação dos instrumentos utilizados na análise;
3. Pessoas aptas a interpretar os dados;
4. Diretrizes para evitar contaminação;
5. Laboratório confiável;
6. Equipe forense e laboratório capacitados para efetuar testes e contínua avaliação da sua capacidade de análise;
7. Aptidão do pessoal de suporte técnico e bom desempenho do laboratório. Os peritos devem ser competentes para alcançar a excelência no serviço forense, além de trabalhar com sistema de qualidade e abordagem correta.

Esse conjunto de boas práticas e ordenamentos existem para garantir que os fatos sejam analisados com base em conhecimentos científicos e conclusões eficientemente precisas dentro da razoabilidade, conforme o mesmo trabalho. Assumindo-se que os conceitos apresentados são válidos para qualquer tipo de crime, deve-se adequar os requisitos para crimes digitais. Existem convenções e leis que suportam essas correspondências, como o artigo 19 da Convenção de Budapeste sobre os procedimentos e definições acerca da busca e retenção de dados em cibercrimes (EUROPA, 2001), a Lei nº 11.419/2006 (BRASIL, 2001), em seu artigo 11, que detalha que documentos produzidos digitalmente e anexados aos processos digitais de forma confiável e autenticada são válidos para todos os efeitos legais, e o Marco Civil da Internet (BRASIL, 2014), em diversos trechos, como no artigo 11, que detalha sobre a manutenção de custódia de registros digitais. Pode-se, portanto, derivar a seguinte lista de requisitos para a investigação de crimes digitais:

1. Técnicas autenticadas e reconhecidas pela comunidade de investigação e tratamento de dados;
2. Titulação dos softwares, sistemas e equipamentos utilizados na análise;
3. Pessoas aptas a interpretar os dados;
4. Diretrizes para evitar alterações, vícios e inconformidades com demais leis aplicáveis na extração, tratamento e armazenamento dos dados obtidos;
5. Laboratório com infraestrutura confiável;
6. Equipe forense e laboratório capacitados tecnicamente para efetuar testes, configurações e contínua avaliação da capacidade de análise dos sistemas empregados;

7. Aptidão do pessoal de suporte técnico para utilização e interpretação dos resultados providos pelas ferramentas tecnológicas e bom desempenho da infraestrutura do laboratório.

Deve-se ainda somar a esses pontos o fato de que o artigo 158 (BRASIL, 1941b) do já citado Código de Processo Penal brasileiro prevê que qualquer infração que deixa vestígios deve ser objeto de exame, ou investigação. Todos esses ordenamentos deixam claro, portanto, a necessidade do provimento de decisões judiciais com base em evidências verificadas e investigadas à luz das técnicas forenses, e esse fato é especialmente relevante e contemporâneo quando se analisa o processo legal para julgamento de conteúdos publicados na *Internet*.

Ocorre que, na atualidade, é prática comum a corte gozar de autonomia para julgar e analisar as informações apresentadas sem a ponderação analítica acerca da significância do conteúdo, conforme o próprio texto do artigo 156 do já citado Código de Processo Penal brasileiro, em que seria facultado ao juiz o pedido de investigação apenas caso este julgue existir fato relevante e necessário. Ou seja, há um claro ponto de subjetividade no procedimento: à autoridade judiciária é concedida a opção de interpretação subjetiva de conteúdo apresentado como prova de crime (por exemplo crime de racismo e/ou ameaça) sem o requerimento da devida justificativa analítica das conclusões. Esse ponto pode levar a inconsistências nas decisões em diferentes tribunais e casos, além de gerar inseguranças a respeito da clara aplicação dos conceitos de presunção de inocência. Em outras palavras, pode ocorrer insegurança quanto a existência de imparcialidade na análise dado que não há dispositivo legal que reconheça e garanta que os membros da corte são suficientemente especialistas em semântica e tratamento de dados de linguagem natural extraídos de diversas fontes da *Internet* para a correta consideração dos conteúdos sem nenhuma necessidade de parecer técnico previamente constituído.

Para a resolução dessa possível ineficiência no processo judicial, a reunião de todos os conceitos de extração, tratamento e processamento de dados abordados neste trabalho pode ser aplicada. Este trabalho pretenderá, portanto, prover um *framework* de trabalho na investigação de conteúdos disponíveis na Internet implementado numa ferramenta que automatiza as etapas de extração, armazenamento e classificação para facilitar e viabilizar o trabalho dos investigadores. Os objetivos a serem alcançados se baseiam na listagem apresentada anteriormente de requisitos para a eficiente aplicação da forense em crimes digitais.

Na seção a seguir, são apresentados os principais desafios na investigação forense, especialmente aqueles relacionados aos crimes digitais, em que sistemas informatizados são usados como meio de execução de atos ilícitos.

2.9 NOVOS DESAFIOS NA INVESTIGAÇÃO FORENSE PARA CRIMES DIGITAIS

Com base em (CAVIGLIONE; WENDZEL; MAZURCZYK, 2017), que explora o futuro da forense digital, desafios relacionados à heterogeneidade das fontes de dados e diversidade tecnológica seriam discussões frequentes para a área. Também com base em (VINCZE, 2016), que

explora desafios correntes, é possível identificar interseções entre os trabalhos, que convergem em relação aos problemas de escala e efetividade dos trabalhos investigativos no domínio digital.

A quantidade massiva e não padronizada de dados gerados é uma consequência direta da heterogeneidade dos dispositivos e tecnologias usados na comunicação entre criminosos e no registro e/ou cometimento de crimes digitais (SHALAGINOV; JOHNSEN; FRANKE, 2017). A tarefa de extrair dados de tão diversas fontes, além de ser complexa e difícil de ser escalada, ainda carece de procedimento padrão. Não existindo um único modelo de aplicativo de redes sociais, ou um único formato de persistência de mensagens, ou ainda uma única tecnologia de armazenamento, e somando-se a variedade de Sistemas Operacionais disponíveis tanto para aparelhos móveis como para computadores e sistemas informatizados, tem-se um universo altamente complexo de se integrar, e mais ainda de se extrair informação de qualidade e cruzar dados entre diferentes plataformas.

Outro desafio está relacionado à dificuldade de eliminar conceitos subjetivos e carentes de conclusões analíticas, como comenta (HUGHES; KARABIYIK, 2020). É tarefa comum da forense digital não apenas a atribuição de culpa, mas também a verificação da indicação de existência ou não de crime. E por vezes o resultado das investigações se torna subjetivo com relação às técnicas e considerações usadas pela equipe investigativa. É tarefa complexa, por exemplo, comparar os resultados das investigações de dois crimes digitais com contextos semelhantes, porém ocorridos em cenários diferentes, justamente pela falta de parâmetros claros na avaliação dos crimes e evidências. Apesar de existirem esforços importantes na definição de regras para evidências digitais aceitáveis, como descrito na Introdução deste trabalho, a diversidade de fontes e tecnologias dificulta a homogeneidade da técnica do trabalho forense clássico de ser cultivada na forense digital (ARSHAD; JANTAN; ABIODUN, 2018).

Esta seção conclui o segundo capítulo desta dissertação. A partir do capítulo seguinte, a arquitetura proposta e as escolhas técnicas que viabilizaram este trabalho e os resultados obtidos serão detalhados.

3 PROPOSTA DE MOTOR DE BUSCA METASEMÂNTICA PARA ANÁLISE DE DADOS DA INTERNET

3.1 ARQUITETURA PROPOSTA

A arquitetura proposta nesta dissertação descreve uma solução de investigação, classificação, indexação e enriquecimento de dados oriundos de páginas da Internet fim a fim. Essa aplicação é composta por:

1. Um serviço de busca e indexação que recebe configurações do usuário, persiste e disponibiliza os resultados da indexação e inicia os processos de extração e predição nos outros serviços;
2. Um serviço de extração especializado em emular ações humanas em páginas *Web* que faz a raspagem dos dados;
3. Um serviço que faz a classificação multi-etiqueta dos textos extraídos usando um número arbitrário de classes de acordo com os dados de treinamento fornecidos e também provê os dados para enriquecimento. A classificação é baseada em um conjunto arbitrário de instâncias que executam o algoritmo supervisionado denominado de máquina de vetor suporte com função de núcleo linear.

A comunicação entre os serviços se dá através de API REST, do inglês *Representational State Transfer*, ou transferência de estado representacional. A aplicação final é uma composição de serviços utilizando tecnologia de contêineres *Docker* e com topologia e configuração descritos através de um arquivo seguindo o padrão YAML no qual estão relacionados os serviços. Esse arquivo é utilizado por um programa auxiliar denominado *Docker Compose* que automatiza a criação e inicialização dos contêineres, da rede virtual que os conecta e executa as demais configurações de mapeamento de arquivos entre máquina hospedeira e contêineres. A configuração, construção e inicialização do serviço de busca e indexação e do serviço de classificação estão descritas em arquivos *Dockerfile*, que contém instruções a serem executadas pela aplicação *Docker* durante a construção dos contêineres.

A aplicação final é multiplataforma, dependendo apenas da instalação prévia da pilha de programas Docker e Docker Compose (e opcional da aplicação Docker Desktop quando aplicável). Isso significa que ela pode ser executada em hospedagem na nuvem, em máquinas físicas locais, em máquinas virtuais, entre outros. Além disso, a aplicação pode também ser compartilhada entre vários usuários, não necessitando de uma instância para cada investigador, sendo acessível e configurável via rede através de uma página *web* após a devida inicialização e provisionamento.

As figuras a seguir descrevem a topologia interna completa da aplicação e como se daria

seu uso compartilhado entre vários usuários. A topologia é composta por três serviços, dois em uma mesma rede, e um terceiro em uma rede própria devido à característica específica desse de funcionar como uma grade de processamento. O acesso da ferramenta pode ocorrer de forma compartilhada e depende um navegador para acesso à interface gráfica da ferramenta.

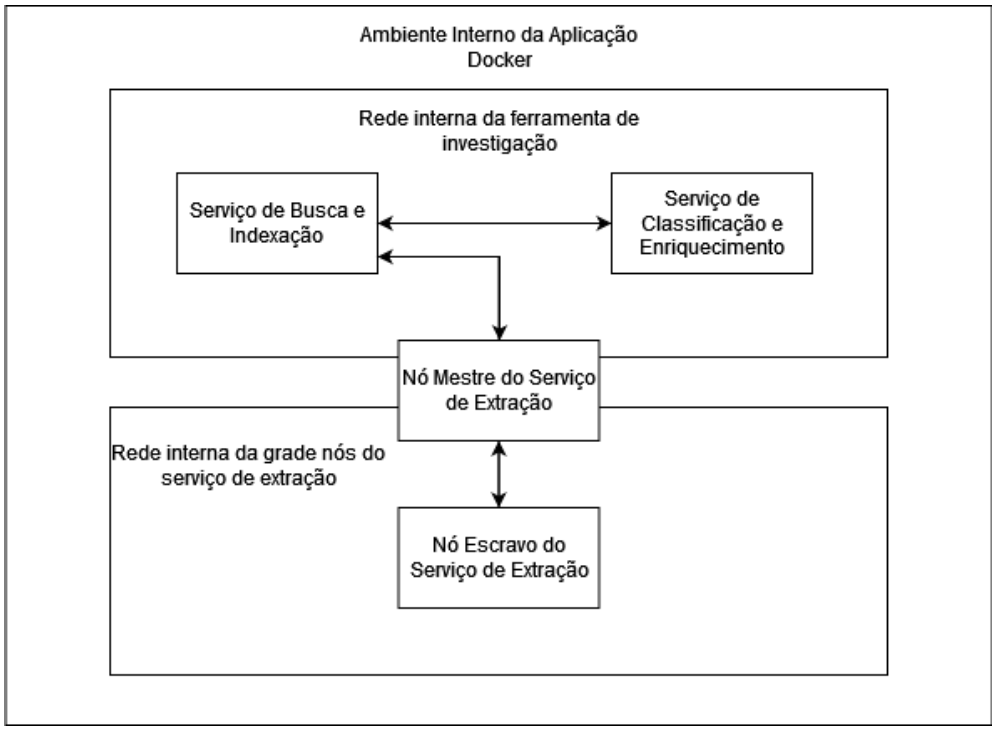

Figura 3.1: Componentes de software da ferramenta de investigação proposta.

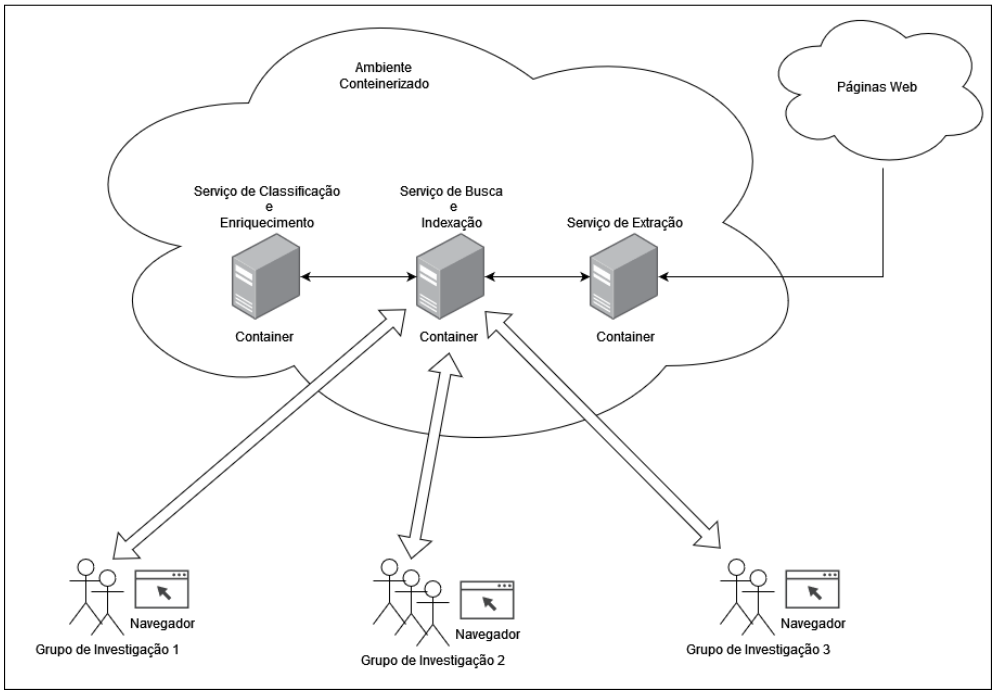

Figura 3.2: Uso compartilhado da aplicação em ambiente conteinerizado.

Um resultado esperado da topologia é a distribuição de carga de processamento. Cada contêiner que sustenta cada serviço tem seus próprios recursos e é responsável por uma parte das responsabilidades do processo de extração, indexação e classificação. Assim, o resultado final esperado é a redução de carga de processamento individual em cada serviço. Isso é especialmente importante devido ao fato de a ferramenta de investigação descrita neste trabalho tem a pretensão de atuar com grandes volumes de dados e de forma compartilhada entre diferentes usuários.

Nas seções seguintes, cada serviço será apresentado, conjuntamente com suas características e decisões de projeto relacionadas. A seção abaixo discorre sobre o primeiro e mais importante serviço, aquele com o qual o usuário interage diretamente e de onde saem os comandos para os demais serviços: o serviço de busca.

3.2 SERVIÇO DE BUSCA E INDEXAÇÃO

Este serviço é concebido através de um conjunto de alterações no funcionamento normal da aplicação de código aberto denominada Fess em sua versão 7.8.1 . Ou seja, esse serviço é uma cópia do projeto original da ferramenta Fess, porém alterado para a integração com outros serviços. Esse serviço é o responsável por disponibilizar uma interface de usuário amigável para realização e inspeção de buscas e também para fornecimento de configurações de customização um programa interno a esse serviço que coordena todo o processo extração, transformação e carga de dados denominado do inglês *Crawler*. Ao ser acionado, dá início a uma série de etapas de detecção de fontes de dados, extração de conteúdo e indexação utilizando um banco de dados local baseado em objetos denominado Elasticsearch (ELASTICSEARCH, 2022).

A sub-seção a seguir detalha a estrutura interna do serviço de busca e indexação.

3.2.1 Arquitetura Interna do Serviço de Busca

O serviço de busca e indexação é formado de diversos outros serviços internos relacionados a outras fontes de dados e funções que a ferramenta disponibiliza de forma nativa. Para este trabalho, no entanto, serão descritos apenas os componentes de software que serão usados na composição da aplicação final.

Esse serviço é composto, portanto, de:

1. Uma interface gráfica com tela de busca e painel de administração para configuração de processos de ETL. Está disponível como página *web* acessível via rede com o uso de um navegador. Esse é baseado num serviço de busca por palavras-chave.

2. Um processo de ETL denominado internamente de *Crawler* o qual, com base nas configurações providas através do painel de administração, irá detectar as fontes de dados, acionar a extração de conteúdo, requisitar a classificação e enriquecimento dos resultados e os indexar numa base de dados local;

3. Base de dados local, sendo constituída por um banco de dados orientado a documentos denominado ElasticSearch. Após o processamento e formação do documento de indexação do conteúdo extraído, cada arquivo será armazenado e consultado por meio deste banco. Ou seja, o resultado das buscas é obtido através de uma busca simples neste banco por palavras-chave contidas no campo de conteúdo extraído de cada documento gerado e persistido. O caráter metasemântico da aplicação é devido às inferências adicionadas ao conteúdo indexado através deste serviço.

A imagem a seguir resume os principais componentes de software importantes para o serviço descrito. O serviço de busca e indexação, que contém tem um banco de dados com os documentos gerados ao fim de cada execução dos processos *Crawlers*, e um agendador para iniciar esses

processos, além de prover a interface gráfica ao usuário; o serviços de classificação e enriquecimento, que irá receber e tratar os dados, e o serviços de extração, que irá buscar e organizar o conteúdo de páginas da *Internet*.

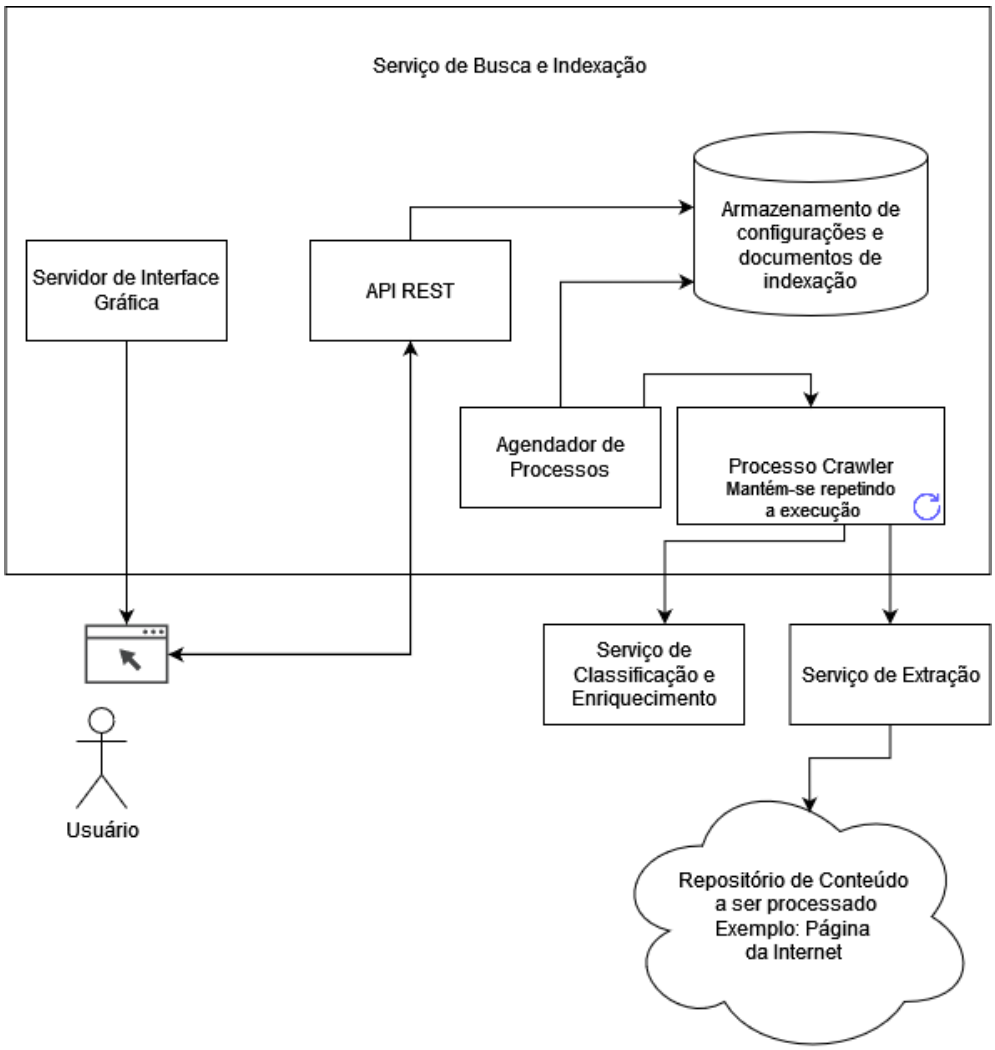

Figura 3.3: Estrutura interna básica do serviço de busca e indexação.

3.2.2 Modificação do Processo de Extração e Indexação de Dados de Páginas da Internet

Apesar de oferecer diversas funcionalidades como extração e indexação de dados de arquivos e de páginas da Internet, esta ferramenta não dispõe de nenhum método nativo que permita a raspagem de dados de páginas que utilizam Javascript e outras tecnologias de renderização dinâmica. Dado que o foco deste trabalho é, portanto, no desenvolvimento de uma solução agnóstica e com a maior abrangência possível em relação às fontes de dados, o componente nativo que realiza a indexação de dados de página *web* foi alterado para que pudesse substituir os dados extraídos nativamente de forma limitada do código fonte estático das páginas e dar lugar a uma sequência de acionamentos de serviços externos que irão lidar com a carga de processamento e complexidade

técnica de extração de dados de páginas *web*, sejam essas de renderização dinâmica ou não, e de classificação dos dados textuais extraídos. Essas alterações se concentram no arquivo *IndexUpdater.java* do projeto original, no seguinte caminho dentro do diretório principal do projeto: *src/main/java/org/codelibs/fess/indexer/IndexUpdater.java* .

A imagem a seguir resume o fluxo de funcionamento do serviço de busca e indexação de páginas *web* após as mudanças. Existem três possíveis fluxos, sendo dois relacionados às tarefas que o usuário executa como administrador/configurador e como usuário da ferramenta de buscas, e o terceiro relacionado com os processos que o programa *Crawler* executa a cada iteração.

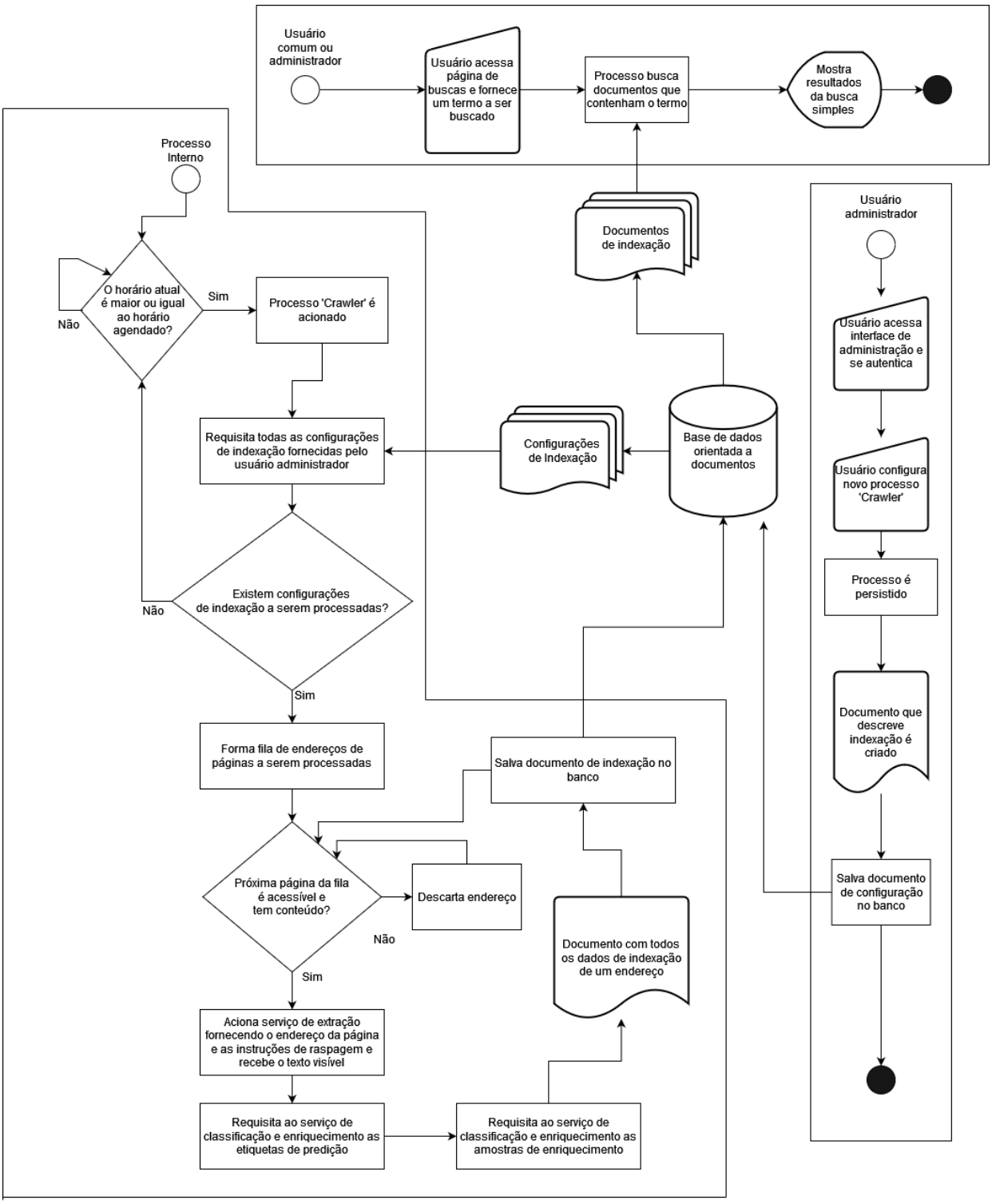

Figura 3.4: Fluxograma do funcionamento do serviço após modificações.

3.2.3 Formato dos Documentos Criados para Indexação

O documento gerado e persistido como resultado do processo completo de ETL tem estrutura de dados seguindo o padrão JSON, do inglês *JavaScript Object Notation*, ou notação de objeto

Javascript (ORG, 2022e). Abaixo um exemplo do resultado do processo de indexação após o fim da execução de um ciclo completo do programa "*Crawler*".


```
{
    "title":"I will kill you",
    "important_content":"I will kill you",
    "config_id":"WkrBBc4QBlzskhnO-S_Ob",
    "expires":"Wed Nov 16 23:31:14 UTC 2022",
    "cache":"<!DOCTYPE html><html><head> <title>I will kill you</title></head><body
     > <h1>I will kill you</h1></body></html>",
    "has_cache":true,
    "digest":"I will kill you",
    "segment":"20221113233106",
    "host":"host.docker.internal:7000",
    "site":"host.docker.internal:7000/1",
    "filename":"1",
    "created":"Sun Nov 13 23:31:17 UTC 2022",
    "anchor":[

    ],
    "mimetype":"text/html",
    "filetype":"html",
    "content_length":117,
    "timestamp":"Sun Nov 13 23:31:17 UTC 2022",
    "indexingTarget":true,
    "boost":1.0,
    "label":[

    ],
    "role":[
       "Rguest"
    ],
    "virtual_host":[

    ],
    "_id":"0
     d5f1bcc0d2a2cc90ca7bc691e6fba2561a61045948b51e869c5b6c36f2d750d19bc3156db21081
    fd5b127b62c8c2a61feae79004ff9f374a5ae3f9362b4ceac",
    "content":"http://localhost:7000/1 I will kill you tag:toxic,tag:threat Hi! I
     am back again!Last warning!Stop undoing my edits or die! || I'm also a sock
     puppet of this account...SUPRISE!!-sincerely, The man that will track you down
     from the Internet and kill you || Whoever put a notices on my page. I will kill
      u || important you and your family shall burn! you have been warned, if you do
      troll muzemikes talk page i will slaughter you in your sleep. bye... || WHAT'S
      WRONG WITH YOU?GO KILL YOURSELF, YOU VILE CREATURE!!!!!!!!!! || ",
    "url":"http://flask_svm/inference?url=http://host.docker.internal:7000/1"
}
```

Chama-se a atenção para o fato de que as buscas por palavras-chave ocorrem apenas no conteúdo da chave ”*Content*”. Note-se também que o documento tem uma data de expiração. Esse detalhe está relacionado ao fato de que páginas *web* tendem a ser atualizadas ou mudar com o tempo, e portanto, um indexador precisa atualizar seus registros repetidamente. Por fim, note-se também que há bastante texto no campo ”*Content*”, inclusive termos começando com ”*tag:*”. Isso se deve a esse documento estar representando um exemplo do inflamento ocorrido ao indexar os resultados do processamento do serviço de classificação e enriquecimento.

Na seção a seguir, o serviço de extração é apresentado.

3.3 SERVIÇO DE EXTRAÇÃO DE CONTEÚDO

O serviço de extração é baseado na tecnologia Selenium e na arquitetura proposta no uso do *Selenium Grid*. Essa aplicação prevê uma estrutura em grade de aplicações de extração de dados de páginas *web*. Os programas usuários não têm conhecimento da estrutura interna e nem de qual nó da topologia está processando seus pedidos, eles apenas fazem requisições ao nó mestre, denominado Selenium Hub (ORG, 2022j) e esse redistribui as sequências de extração entre os nós especialistas internos à estrutura em grade.

Na sub-seção a seguir, mais detalhes da estrutura interna pretendida para o serviço de extração.

3.3.1 Arquitetura Interna do Serviço de Extração

A imagem a seguir contém a estrutura geral da implementação do *Selenium Grid* já associada ao uso da tecnologia de contêineres Docker.

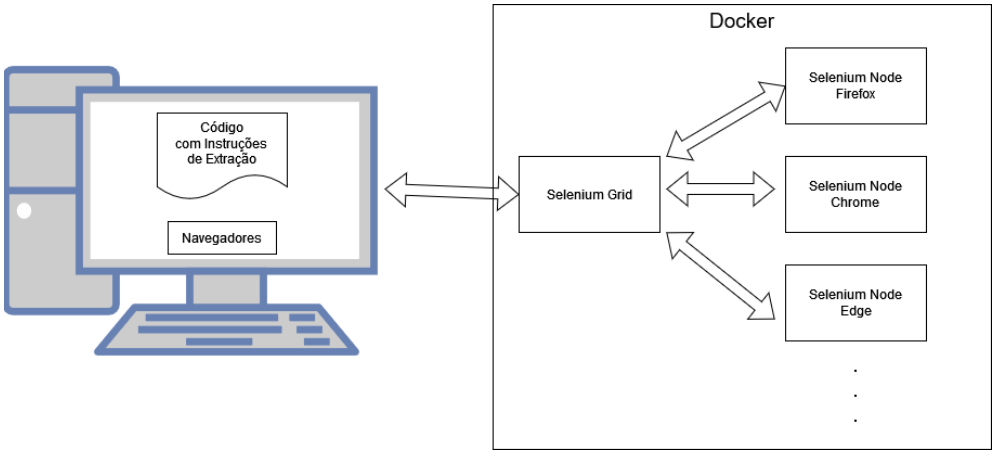

Figura 3.5: Estrutura em grade do serviço de extração usando a implementação de um *Selenium Grid* em ambiente de contêineres *Docker*

Esse serviço será formado, portanto, por um nó mestre *Selenium Hub* e um nó especialista baseado no navegador *Firefox*. O serviço não exige codificação adicional para cada contêiner,

permanecendo o código a ser executado nas aplicações clientes. No caso deste trabalho, o código que norteia a extração está localizado no programa *Crawler* interno ao serviço de busca e indexação, mantendo as responsabilidades de cada componente, já que esse programa seria o equivalente a um orquestrador do processo de extração. Por esse mesmo motivo, o código de extração é bastante genérico, comandando apenas o acesso à uma página *web* e o pedido do extração do texto visível de todos os elementos gráficos da página após totalmente carregada. Dessa forma, o processo de extração passa a ser independe do conteúdo ou forma da página *web* solicitada, visto que para a finalidade da aplicação final, basta a informação textual das páginas.

A sub-seção a seguir contém comentários adicionais sobre o processo de raspagem de dados pretendido para esse serviço e as limitações decorrentes.

3.3.2 Considerações Sobre o Método de Extração e Seus Resultados Esperados

Os comandos disponíveis para interação e extração de dados de páginas *web* utilizando a ferramenta Selenium são bastante amplos e flexíveis. Entretanto, visto que o objetivo da aplicação final descrita neste documento é indexar dados visíveis, classificá-los e enriquecê-los, o procedimento de extração que o serviço de busca envia para o serviço de extração promove a raspagem apenas de conteúdo visível. Ou seja, apenas conteúdo que um usuário comum enxergaria através da tela de um dispositivo ao visitar a página.

Todavia, devido à grande variedade de comportamentos e funcionalidades existentes em páginas *web*, não há um procedimento único para a extração satisfatória de todos os dados visíveis de qualquer cenário. É necessário, portanto, o desenho de um fluxo de interação e extração que atenda ao máximo possível de casos e, naqueles em que não haja total eficiência, seja verificada ao menos compatibilidade parcial. Ou seja, é necessária uma sequência de passos que aumente as chances de que o máximo de informações visuais possível sejam extraídas, numa abordagem de melhor esforço. O fluxograma a seguir contém, por fim, a sequência de passos desenhada para este projeto e que atende a maior parte dos cenários possíveis.

Figura 3.6: Fluxograma de instruções para extração de dados a ser executada pelo serviço de extração.

Ainda que a abordagem seja simples, espera-se que seja eficiente para a maior parte dos casos devido à natureza do consumo de informação através da Internet. A explicação analítica se baseia no fato de que quase todas as páginas *web* têm, em seu projeto, uma página principal, a qual o usuário verá logo no início de sua jornada, ou seja, uma página principal, ou comumente denominada do inglês *home page*. Essa página tende a ser a primeira experiência de interação do usuário com um determinado conteúdo, e portanto, tende a conter, na média, toda ou parte relevante da informação principal a ser comunicada. Esse fenômeno é observado de forma empírica, mas tam-

bém tem fundamentos em teorias e boas práticas de marketing digital, como as famosas *landing pages*, do inglês, ou páginas de pouso, que são páginas destinadas a acolher usuários que foram direcionados por algum fluxo de marketing ou conteúdo promocional (BECKER et al., 2009). Existem indícios ainda que sugerem que a velocidade com que um usuário entende o propósito de uma página está diretamente relacionada à possibilidade do usuário de continuar consumindo aquele conteúdo, como explicitado em (ENNEW et al., 2005). Portanto, extrair a informação diretamente do endereço principal e ignorar subpáginas e elementos reativos se mostra uma estratégia válida. Isso também é verdade devido ao fato de que o usuário pode repassar ao programa *Crawler* no serviço de busca e indexação quantos endereços de rede e sub-endereços quiser que sejam processados, bastando listá-los na configuração, sendo todos tratados individualmente.

Uma alusão ao algoritmo é possível ao se imaginar um complexo comercial com várias lojas, tendo cada uma a sua própria vitrine. A sequência de passos escolhida é equivalente a um robô cuja tarefa é reunir informações sobre as lojas do aludido prédio comercial. Entretanto, esse robô se baseia apenas no processamento das vitrines das lojas, sem adentrar e interagir com os produtos ou vendedores. O resultado final, portanto, terá um certo de grau de precisão, pois espera-se que as vitrines sejam boas representantes do conteúdo das lojas, no entanto, as experiências adicionais ficarão ausentes. Essa comparação torna direta a compreensão das limitações da aplicação final, estando abaixo algumas delas listadas:

1. Não espera-se a extração completa de conteúdos de páginas que dependam de algum tipo de interação inicial, como um movimento de *mouse* ou digitação;

2. Não são esperadas extrações de conteúdos dispostos em vídeos, áudios, imagens e/ou elementos gráficos estilizados, dado que todos esses conteúdos dependem de etapas de processamento adicional os quais não foram incluídos neste projeto e que dependem também de tempos de carregamento e consumo variados, os quais não podem ser facilmente inclusos num algoritmo genérico;

3. Não se espera a extração de dados de arquivos que tenham de ser baixados e/ou lidos através de programas terceiros que não um navegador;

4. Não há previsão de extração de conteúdos que somente possam ser visitados após autenticação, contudo, essa possibilidade pode ser adicionada ao projeto sem mudanças na arquitetura geral, bastando apenas a mudanças das instruções de extração enviadas ao serviço descrito nesta seção. A exclusão de tal funcionalidade se deveu principalmente ao fato dela não ser relevante para a validação da aplicação, que é apresentada como ferramenta de investigação, e por não haver forma genérica e global de implementação.

A seção a seguir irá apresentar o serviço de classificação e enriquecimento.

3.4 SERVIÇO DE CLASSIFICAÇÃO E ENRIQUECIMENTO

O serviço de classificação e enriquecimento foi totalmente implementado em linguagem Python utilizando principalmente as bibliotecas Pandas, SKLearn e Flask (ORG, 2022c). É um serviço que, com base num arquivo tabular com extensão CSV, do inglês *Comma Separated Values*, ou valores separados por vírgula (ORG, 2022q), como entrada com formato específico, identifica quantas dimensões estão presentes e treina uma instância de um algoritmo de aprendizado supervisionado de máquina de vetor suporte com função de núcleo linear para cada dimensão. Então, expõe numa API REST três funções:

1. Uma função que recebe um texto e retorna a lista, caso exista, de etiquetas aplicáveis como classificação;
2. Uma função que dado um endereço de uma página, busca num banco os dados de enriquecimento das inferências já realizadas e os retorna ao cliente;
3. Outra função que, dado um endereço de uma página *web*, retorna o código de uma página intermediária com todos os dados de inferência que já foram processados para aquele endereço e um botão para visitar o endereço original.

Note-se, portanto, que a classificação multi-etiqueta é feita através da classificação binária individual das amostras para cada dimensão. É interessante notar também que a escolha do algoritmo de máquinas de vetor suporte se deve à flexibilidade do modelo, permitindo o uso de funções núcleo lineares e não lineares, e à qualidade da implementação em linguagem Python, que ocupa pouco espaço em memória uma vez que o treinamento tenha finalizado. Além disso, a tarefa de classificação binária equivale à análise de sentimento, que tem nas máquinas de vetor suporte bons resultados, como visto em (MUHAMMAD; BUKHORI; PANDUNATA, 2019). No entanto, qualquer outro modelo poderia ter sido escolhido, principalmente devido ao fato de que a criação de instâncias de máquinas de classificação e treinamento se dá dentro de uma única função, a qual poderia sofrer alterações sem prejuízo ao fluxo geral de funcionamento do serviço. Ou seja, o próprio desenho deste serviço é, também, agnóstico em relação ao modelo de aprendizado de máquina a ser utilizado.

A sub-seção a seguir detalha a arquitetura interna do serviço disposto nesta seção.

3.4.1 Arquitetura Interna do Serviço de Classificação e Enriquecimento

A imagem a seguir mostra a estrutura genérica interna do serviço de classificação. Isso porque a quantidade de instâncias de classificação é dependente dos dados usados como base de treinamento.

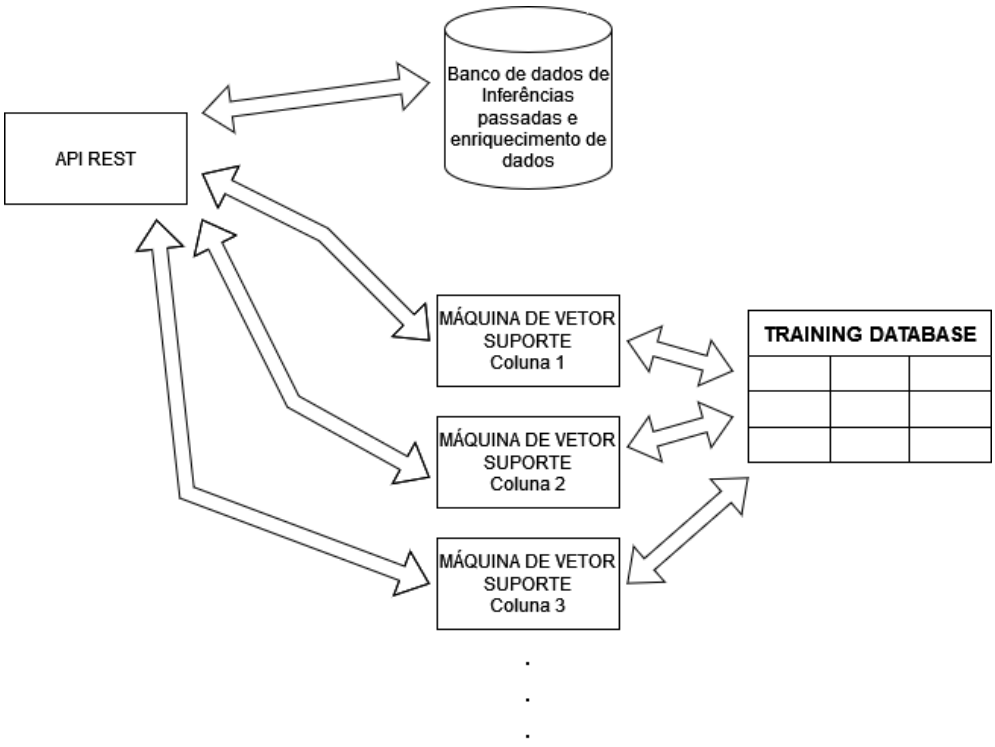

Figura 3.7: Estrutura interna do serviço de classificação e enriquecimento de dados.

A sub-seção a seguir detalha a estrutura da base de dados de treinamento usada na inicialização do serviço.

3.4.2 Base de Dados de Treinamento

Os dados de treinamento são inicializados através de um arquivo tabular com extensão CSV em que cada linha é formada por ao menos uma coluna denominada do inglês *Content*, ou conteúdo. Essa coluna guarda os dados que serão usados para processamento. O arquivo também deve ter outras colunas com nome arbitrário representando as possíveis dimensões, ou classes, para as quais os modelos serão treinados. Cada uma dessas colunas deve ser preenchida unicamente com os valores ”0”, representando uma não atribuição daquela classe à determinada amostra na coluna ”*Content*”, e ”1”, representando uma atribuição correta da classe ao conteúdo daquela amostra.

Qualquer tipo de base de dados que atenda ao formato descrito é aceitável para a inicialização da aplicação. Isso é prova do agnosticismo atingido com a implementação desse serviço. Ou seja, é possível o direcionamento da atuação do serviço de classificação para atuar com quaisquer classes desejáveis e em qualquer indústria ou área do conhecimento. Todavia, dado que o foco deste trabalho são os crimes digitais, será detalhado no próximo capítulo uma base de treinamento que reúne dados de comentários maliciosos que possivelmente constituem crimes digitais.

Na sub-seção a seguir apresenta-se o fluxo de funcionamento pretendido para este serviço.

3.4.3 Fluxo de Treinamento e Funcionamento Geral

A imagem a seguir mostra um fluxograma com a sequência de eventos que ocorrem a partir da inicialização, treinamento das instâncias de máquinas de vetor suporte, até o período de funcionamento normal deste serviço, no qual podem ser recebidas entradas e persistidos resultados de processamento e inferências. Esse processo envolve a geração da lista de classes e dos dados de enriquecimento, que em resumo constituem-se de uma lista arbitrária de 0 a 10 (dez) amostras da base de treinamento que tenham exatamente as mesmas classificações que o conteúdo de entrada. O intuito dos dados de enriquecimento é fornecer pistas ao usuário do viés presente nas máquinas que constituem o serviço de classificação, permitindo a melhor tomada de decisões. Chama-se a atenção para o fato de que caso a base de dados de treinamento fosse um apanhado de amostras de crimes digitais reais e já julgados, o resultado do enriquecimento desse serviço seria bastante útil ao indicar a um investigador, por exemplo, amostras de outros crimes semelhantes do ponto de vista do serviço, favorecendo a criação de teses mais confiáveis, situação considera ideal para o trabalho investigativo.

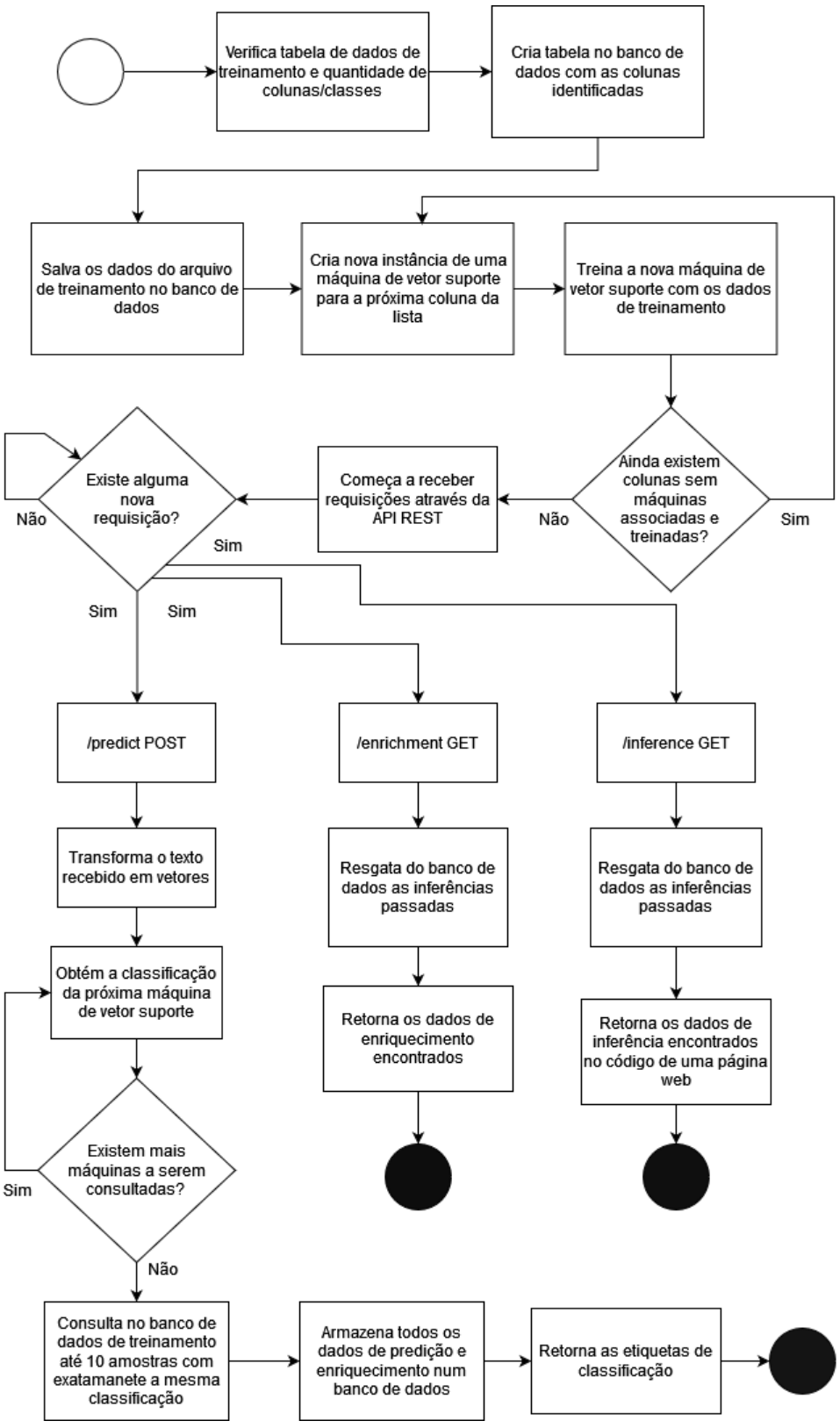

Figura 3.8: Fluxograma do funcionamento do serviço de classificação e enriquecimento.

A seção a seguir traz comentários sobre a distribuição pretendida da arquitetura da aplicação final em ambiente conteinerizado.

3.5 CONTEINERIZAÇÃO E INICIALIZAÇÃO DO PROJETO

É importante destacar que os serviços foram desenhados de forma a isolar completamente as responsabilidades e dependências. Dessa forma, estes também não precisam ser executados no mesmo local e nem mesmo na mesma rede, bastando que tenham rotas de comunicação entre si. Essa flexibilidade é importante especialmente em contextos de uso de tecnologias de nuvem, em que os usuários poderiam ter a pretensão de executar alguns (ou todos) os serviços de forma remota numa nuvem privada ou pública.

Todavia, ainda que exista bastante flexibilidade na forma e nos locais nos quais os serviços são hospedados, no contexto deste trabalho, decidiu-se pela distribuição utilizando contêineres que seriam executados todos numa mesma máquina. Dessa forma, foi necessária a descrição dos passos de inicialização e construção das imagens dos serviços de busca e indexação e de classificação utilizando arquivos *Dockerfile* com instruções na linguagem BASH do Linux, do inglês *Bourne Again Shell*, ou Shell nascido novamente (ORG, 2022d). Também, a descrição da topologia foi feita através de um arquivo YAML para o consumo através da aplicação *Docker Compose*. Essa topologia prevê contêineres rodando em uma rede virtual privada, e o investigador acessando apenas a interface de usuário do serviço de busca e as páginas intermediárias das inferências geradas pelo serviço de classificação e indexação através da rede.

Uma vez atendidos os pré-requisitos, que são:

1. Instalação do software ”Docker”
2. Instalação do software ”Docker Compose”
3. Instalação opcional, porém recomendada, do software ”Docker Desktop”

A distribuição pretendida para o software será através de uma pasta compactada, principalmente devido a existência de arquivos necessários e com grande tamanho em disco. Apesar disso, um repositório público será criado e disponibilizado em cumprimento ao que pede a licença Apache 2.0 (ORG, 2022a) presente no projeto original da ferramenta Fess. Por fim, a inicialização do projeto se dará através do seguinte comando executado dentro da pasta descompactada do projeto:

```
docker compose up -d
```

A seção a seguir apresenta um fluxograma completo do processo de funcionamento da aplicação, dando significância ao viés metasemântico presente na ferramenta.

3.6 CONSIDERAÇÕES SOBRE O PROCESSO DE INDEXAÇÃO METASEMÂNTICA

Processos de indexação metasemântica, como dito anteriormente, não têm definição única. Para o contexto deste trabalho, a caracterização da aplicação de busca como metasemântica se baseia no fato de que a pesquisa, em si, não utiliza nenhuma análise sintática ou método numérico. De fato, o buscador é baseado puramente em palavras-chave. Todavia, o conteúdo resultante da extração passa por transformações adicionais e é enriquecido com etiquetas e o conteúdo de outras amostras do banco de dados de treinamento, as quais tenham a exata mesma classificação. Dessa forma, o documento de indexação resultante de cada endereço de página conterá o próprio endereço, o título da página, o conteúdo textual visível, as etiquetas de classificação, e o conteúdo de outras amostras que contenham as mesmas etiquetas. Por fim, ainda que o processo de pesquisa efetuado com base nas entradas do usuário não tenham caráter semântico, o texto indexado e persistido é resultado de um processo complexo e com teor semântico. Ao buscar por uma ou mais etiquetas, ou mesmo pelo texto de uma amostra usada no enriquecimento, o que é retornado ao usuário, e o que esse investigador enxerga nas páginas intermediárias ao clicar em um dos resultados visíveis, é um conjunto de elementos de linguagem natural considerados próximos por um método numérico previamente treinado para performar tais inferências.

A seção a seguir detalha o processo de configuração do programa *Crawler* e dicas para a realização de buscas eficazes.

3.7 REGRAS PARA CONFIGURAÇÃO DE NOVOS PROCESSOS E REALIZAÇÃO DE BUSCAS USANDO A INTERFACE GRÁFICA

A configuração de um processo *Crawler* deve ser realizada através do painel de administração, acessado por meio da página de buscas. Através da opção *Crawling* no menu lateral e opção *Web* no submenu, será possível criar novas configurações de processos e definir parâmetros e o endereço de páginas *web* alvo de processamento.

Durante a configuração de um novo processo *Crawler*, haverá um campo no formulário denominado *"Config Parameters"*. Este é o campo onde os parâmetros de configuração especiais que ativam as integrações e modificações descritas nesta dissertação são ativadas. A seguir uma tabela com as configurações necessárias, que devem ser fornecidas linha a linha.

Adicionalmente, a busca que o usuário fizer pode ser focada em qualquer um dos seguintes tipos de conteúdos:

1. Conteúdo original que foi extraído de cada página *web* objeto de processamento;
2. Etiquetas atribuídas a cada conteúdo pelo serviço de classificação;
3. Conteúdo usado para enriquecimento com base nas amostras da base de dados.

Tabela 3.1: Parâmetros de Configuração

Parâmetro	***Descrição***	***Exemplo de Uso***
webDriver	Liga ou desliga o uso do serviço externo de extração	webDriver=enable
webDriverURL	Endereço de rede do serviço de extração	webDriverURL=http://selenium-hub:4444
inference	Liga ou desliga o enriquecimento de dados com o uso do serviço de classificação	inference=enable
inferenceURL	Endereço de rede da API REST do serviço de classificação	inferenceURL=http://flask-svm-fess/predict

Parâmetros de configuração necessários para a integração dos serviços

Em relação ao conteúdo extraído ou ao usado no enriquecimento, o usuário pode buscar usando palavras-chave, e terá a visão completa através das páginas intermediárias com os dados da inferência gerados pelo serviço de classificação, como será visto no próximo capítulo. Já em relação às etiquetas que classificam o conteúdo extraído, é necessário usar uma sintaxe específica:

1. Para o caso de um usuário que deseja pesquisar por todos os conteúdos classificados na categoria "*ofensivo*", deverá digitar "*tag:ofensivo*" no campo de buscas;
2. Para o caso de um usuário que deseja pesquisar por todos os conteúdos classificados na categoria hipotética "*racismo*", deverá digitar "*tag:racismo*" no campo de buscas;
3. Para o caso de um usuário que deseja pesquisar por todos os conteúdos classificados nas categorias hipotéticas "*racismo*" e "*ofensivo*", deverá digitar "*tag:ofensivo,tag:racismo*" no campo de buscas.

No capítulo seguinte, são apresentados, explorados e discutidos os resultados do teste da aplicação com amostras de dados fictícias e com treinamento fundamentado numa base de dados de comentários maliciosos e possivelmente criminosos obtidos de páginas da Internet.

4 EXPERIMENTO REALIZADO E RESULTADOS

4.1 METODOLOGIA E TESTE

As etapas para execução dos testes de validação foram divididas em três etapas:

1. Desenvolvimento e provisionamento de páginas *web* com conteúdos a serem processados. Essas páginas estarão descritas em um arquivo padrão YAML da aplicação "*Docker Compose*" separado do projeto da ferramenta de investigação. Algumas páginas de teste foram desenhadas também reusando o mesmo projeto da página de inferências, que é uma adaptação do projeto disposto em (ISAKOVIC, 2022);
2. Inicialização do projeto da ferramenta de investigação usando a aplicação "*Docker Compose*";
3. Configuração e execução de processo *Crawler*;
4. Realização de buscas e verificação de resultados.

Os testes visam validar o processo fim a fim de investigação assistida pela ferramenta.

Na sub-seção a seguir, inicia-se a descrição dos procedimentos para disponibilização de páginas *web* fictícias para o teste da ferramenta.

4.1.1 Desenvolvimento e Provisionamento de Páginas *Web* Para Indexação

Dado que não faz parte do escopo deste trabalho a busca por páginas da Internet cometendo reais crimes ou ofensas graves, e que classificar e registrar páginas públicas sem consentimento e sem o apoio de pessoas autorizadas pela Lei poderia incorrer em violação das regras de usos de vários portais, decidiu-se por emular páginas *web* internamente ao ambiente "*Docker*" e acrescentar um projeto adicional à estrutura que retorne as páginas a serem processadas pela aplicação final.

Para tanto, a figura a seguir ilustra a estrutura das aplicações usadas para teste em ambiente conteinerizado no experimento em que há dois projetos sendo executados em um ambientes *Docker* hospedado numa máquina física local. Um dos projetos está relacionado à provisão de páginas *Web* de teste e o outro, ao acesso da solução de investigação proposta.

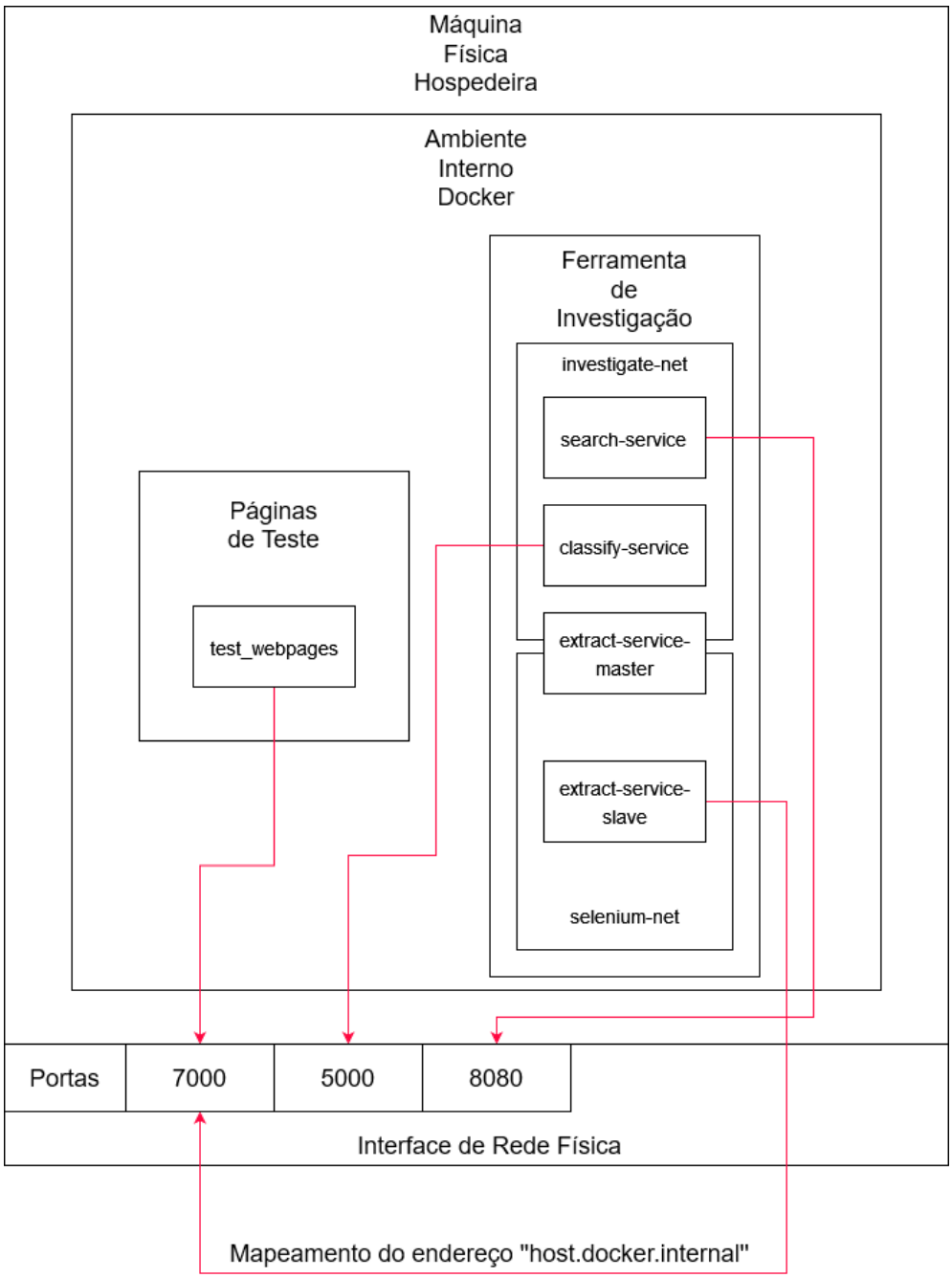

Figura 4.1: Estrutura da aplicação testada e executada em ambiente conteinerizado Docker.

A sub-seção a seguir comenta sobre a base de dados de treinamento para as instâncias de aprendizado de máquina.

4.1.2 Carga Inicial de Dados Para Treinamento

Os dados fornecidos para treinamento são de uma base obtida em (JIGSAW, 2022) que lista diversos comentários no portal Wikipédia (ORG, 2022r) na Internet e disponibiliza a classificação com base em 6 categorias: tóxico, seriamente tóxico, obsceno, ameaça, insulto e discurso de

ódio (especialmente com relação a identidades), que no inglês seriam designadas pelas seguintes palavras-chave (toda a base está disposta em língua inglesa):

1. *toxic*
2. *severe_toxic*
3. *obscene*
4. *threat*
5. *insult*
6. *identity_hate*

Mais informações sobre esta base, as motivações e achados podem ser verificados em (JIGSAW, 2017).

Devido ao fato de a base original ser bastante extensa e não fazer parte do escopo deste experimento a avaliação profunda da qualidade das classificações, e sim do funcionamento geral da aplicação, decidiu-se por utilizar apenas uma parte da base de treinamento, reduzindo-a de quase 160.000 amostras para pouco menos de 3.500 amostras. O objetivo era a redução do tempo de treinamento e da quantidade de recursos necessários às instâncias de inferência.

A sub-seção a seguir comenta sobre a inicialização e provisionamento da aplicação.

4.1.3 Inicialização da Aplicação

Após a cópia dos arquivos da aplicação para uma máquina com a pilha de software Docker, Docker Desktop e Docker Compose instalados, a construção e inicialização dos serviços foi feita usando o seguinte comando:

```
1 docker compose up -d
```

O comando inicia a construção das imagens dos projetos em código do serviço de busca e indexação e do serviço de classificação e enriquecimento. Além disso, baixa e instanciar localmente os contêineres Selenium Hub, o nó mestre da grade de extração, e Selenium Firefox (ORG, 2022n), o modelo de nó especialista escolhido para a tarefa de raspagem de dados no experimento. A figura a seguir mostra os serviços sendo executados de forma independente dentro do mesmo projeto na interface do software *Docker Desktop*.

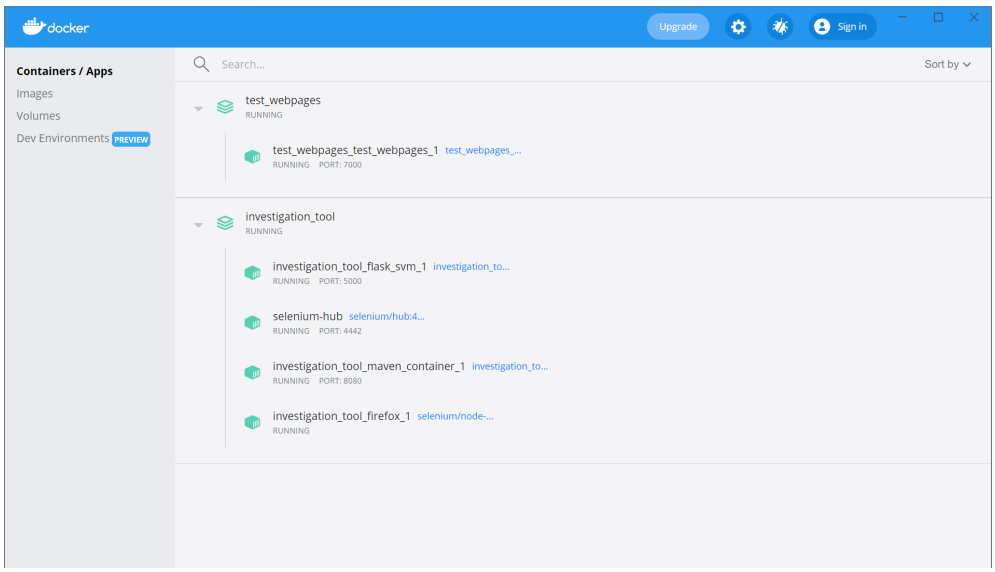

Figura 4.2: Serviços da aplicação sendo executados em contêineres em ambiente Docker.

A interface de usuário com o campo de buscas e o botão de acesso ao painel de administração estavam disponíveis através do endereço *localhost:8080/* . A imagem a seguir mostra uma visão da página de buscas.

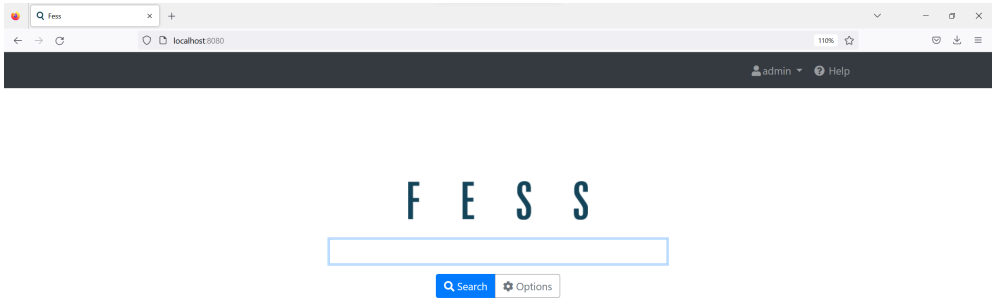

Figura 4.3: Página de buscas da ferramenta de investigação.

A sub-seção a seguir comenta sobre a preparação de uma configuração válida para o programa *Crawler*, que orquestrará o processo ETL completo. Note-se que o endereço de rede ”host-docker-internal” é um comando nativo da aplicação ”*Docker*” que permite a um contêiner acessar a interface de rede da máquina física hospedeira.

4.1.4 Validação da Configuração de Um Processo de Busca

A figura a seguir mostra um detalhe da configuração de um novo processo *Crawler* onde se define os endereços das páginas *web* a serem processadas e os parâmetros de configuração para o

processo. No caso, foram usados parâmetros que permitissem ao contêiner do serviço de busca se comunicar com os serviços de extração e de classificação e enriquecimento dentro da rede virtual no ambiente Docker. Essas configurações podem ser acessadas através da opção ”*Web*” dentro do submenu que aparece ao selecionar a opção ”*Crawler*” no menu lateral principal no painel de administração da ferramenta.

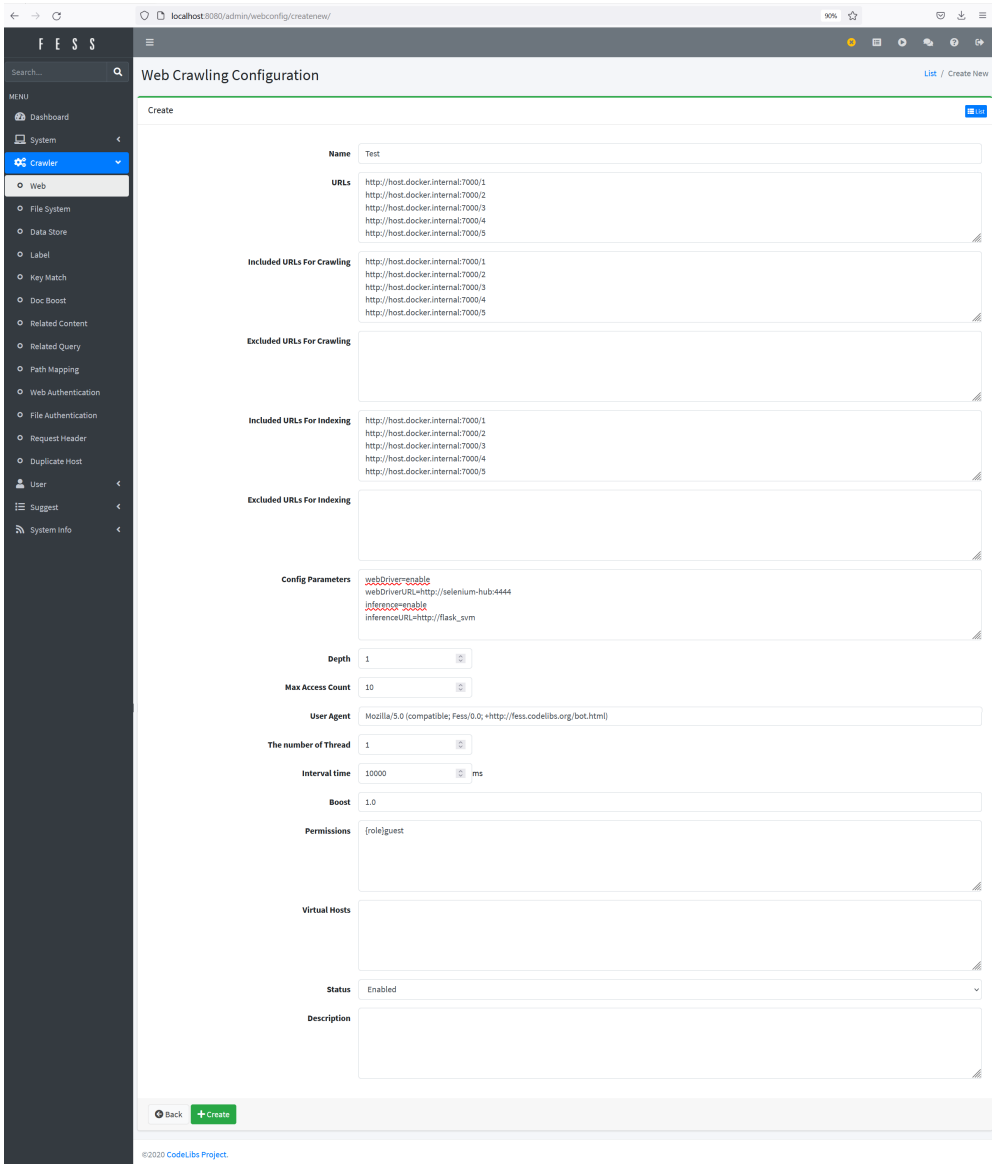

Figura 4.4: Configurando novo processo *Crawler* no painel de administração do serviço de busca e indexação.

A sub-seção a seguir detalha o acompanhamento da execução do programa *Crawler*

4.1.5 Execução do Processo *Crawler*

Através da opção ”*System*”, ou sistema em português, no menu lateral principal no painel de administração da ferramenta é possível acessar o submenu com a opção ”*Scheduler*”, ou agendador em português. Acessando essa opção, é possível ver uma lista de processos a serem exe-

cutados pela ferramenta. Selecionando o processo denominado "*Default Crawler*" é possível dar início à execução do processo *Crawler* manualmente, ainda que este esteja agendado para ser executado ao menos uma vez por dia à meia noite. As figuras a seguir mostram o início do processo e os logs de sua execução, que podem ser visualizados através do *Download* dos registros na opção "*Log Files*", ou arquivos de registro em português, dentro do submenu que aparece ao selecionar a opção "*System Info*", ou informações de sistema em português. Aparecerá uma listagem de arquivos de registro disponíveis, devendo-se escolher pelo denominado "*fess-crawler.log*".

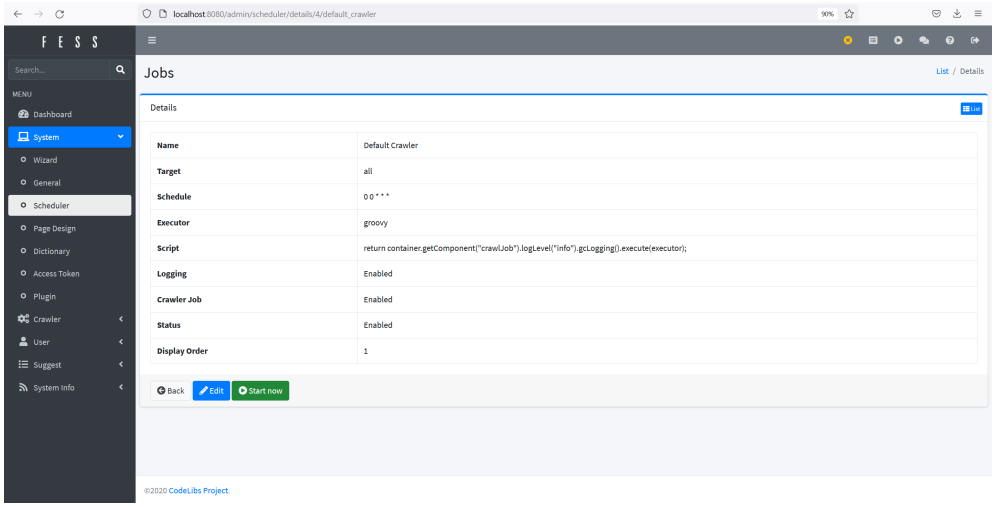

Figura 4.5: Iniciando processo *Crawler* no painel de administração do serviço de busca e indexação.

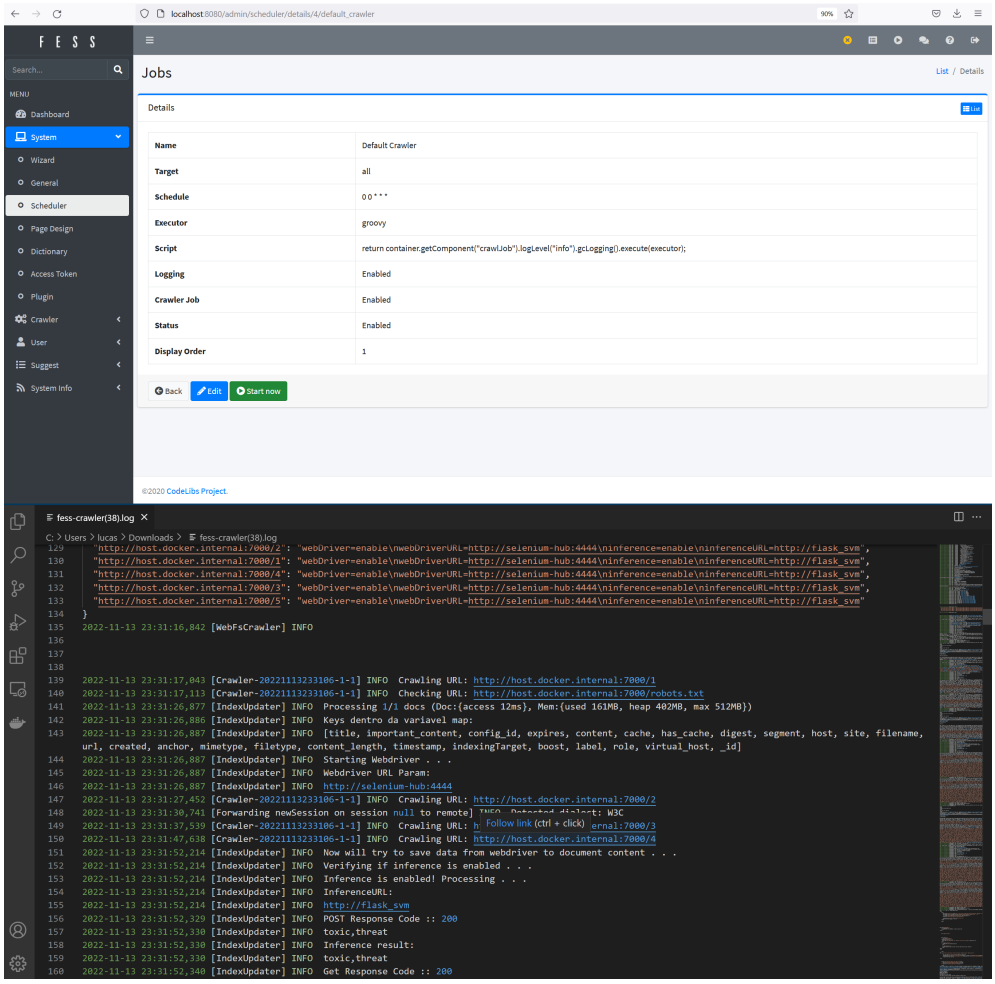

Figura 4.6: Detalhe dos registros da execução do processo *Crawler*. Na parte superior o local onde os registros podem ser consultados. Na parte inferior detalhe dos registros da execução do programa *Crawler* processando uma página *web* da configuração.

A sub-seção a seguir apresenta os resultados das buscas feitas através da interface de usuário após o fim do processo de ETL completo orquestado pelo programa *Crawler*.

4.1.6 Resultados e Inferências Obtidas

Nesta seção, são explorados, finalmente, os resultados que seriam objeto de formulação de teses por usuários investigadores. A imagem a seguir fornecem um detalhe dos resultados após uma busca simples por uma palavra arbitrária.

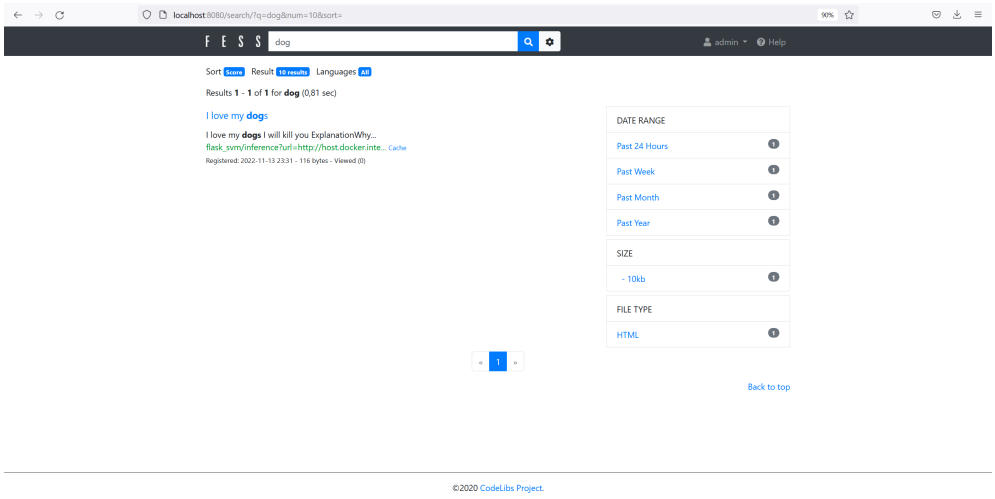

Figura 4.7: Resultado de uma busca simples pela palavra ”*dog*”.

As imagens a seguir mostram o resultado da busca por uma única etiqueta, ou classificação, e mais abaixo, a busca por um conjunto específico de etiquetas ou classificações. Note-se o uso do marcador *”tag:”*.

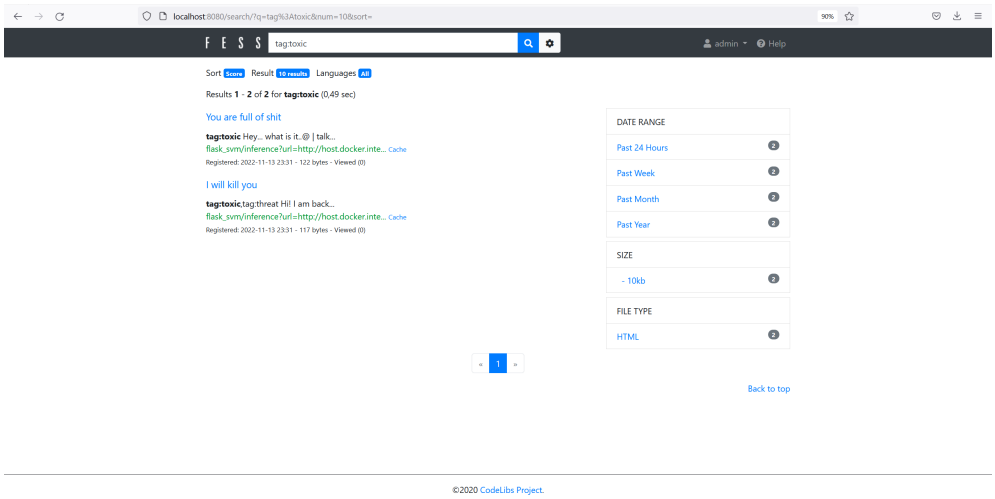

Figura 4.8: Resultado da busca por uma única classe específica. Note-se o uso do marcador ”tag:”.

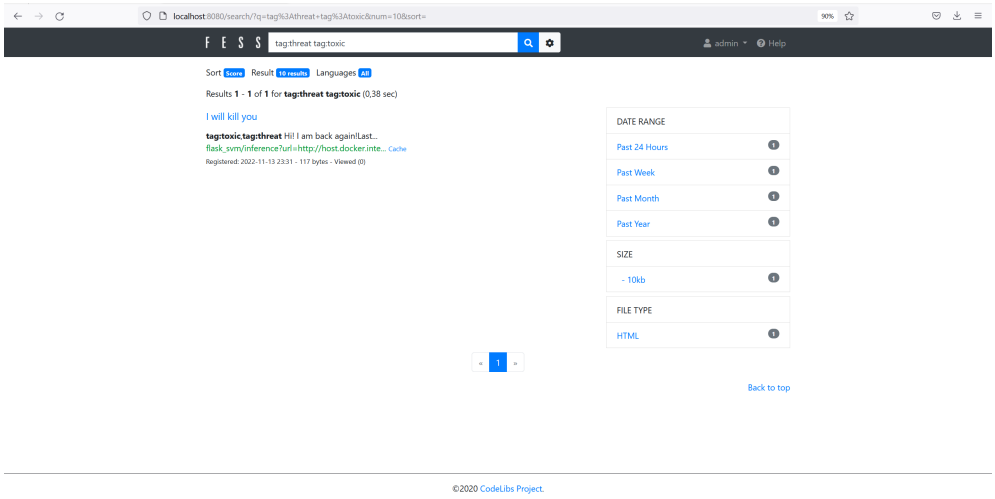

Figura 4.9: Resultado da busca por um conjunto específico de classes, ou etiquetas. Note-se o uso do marcador ”tag:” antes de cada termo.

As imagens a seguir apresentam em sequência dois pares de páginas intermediárias, as quais são mostradas ao se acessar qualquer resultado proveniente de uma busca, e as páginas originais objetos da investigação.

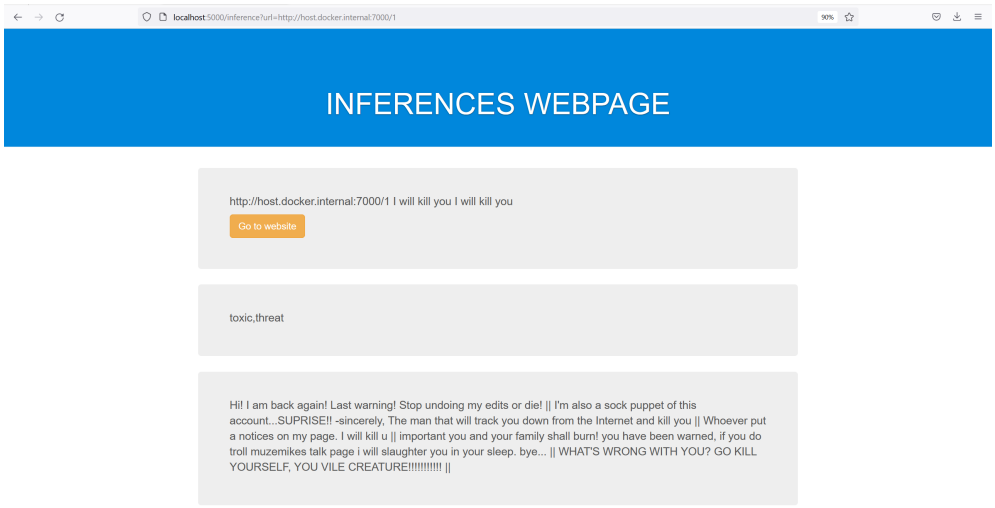

Figura 4.10: Página intermediária com os dados de inferência e enriquecimento do processamento da página com endereço ”localhost:7000/1”.

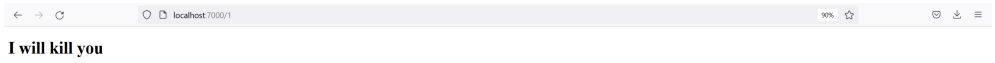

Figura 4.11: Conteúdo visual da página em ”localhost:7000/1”.

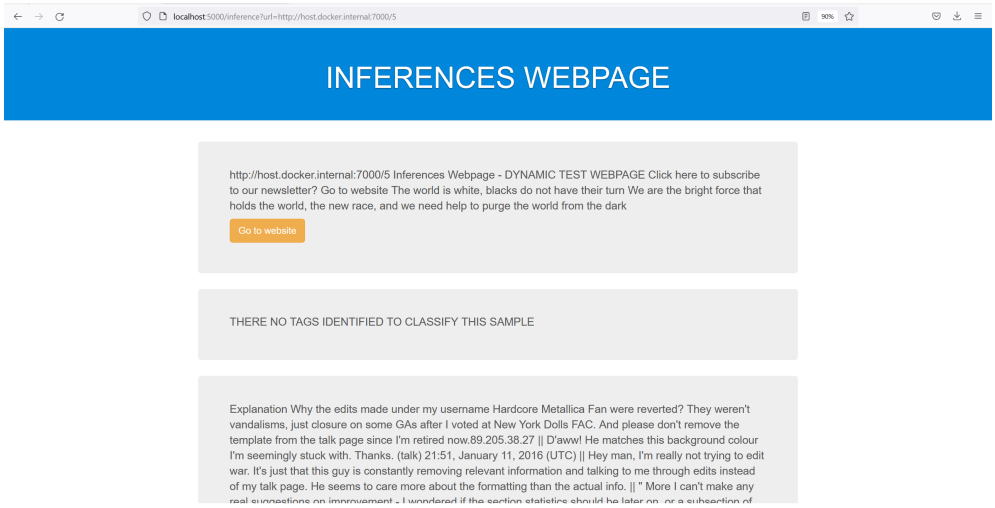

Figura 4.12: Página intermediária com os dados de inferência e enriquecimento do processamento da página com endereço ”localhost:7000/5”.

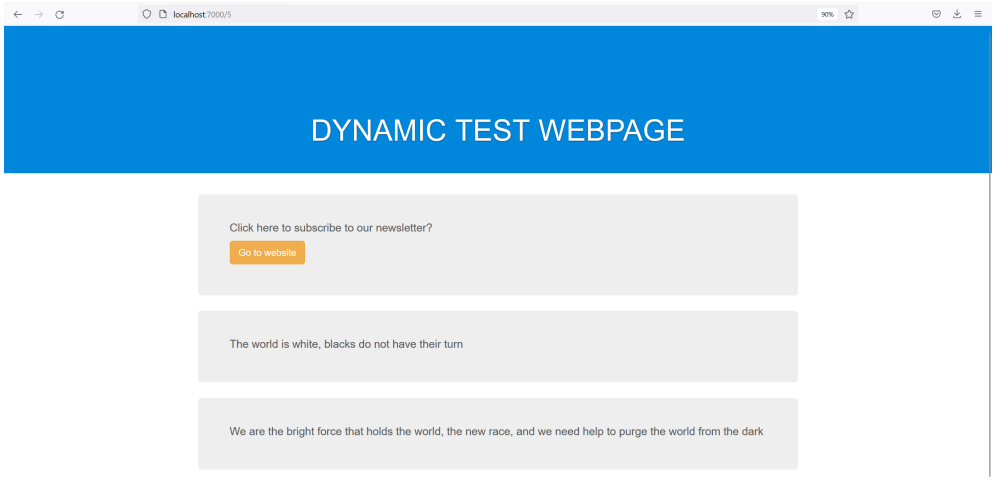

Figura 4.13: Conteúdo visual da página em ”localhost:7000/5”.

É possível notar que os requisitos listados na subseção ”Aplicação na Forense Digital” do segundo capítulo deste trabalho foram facilitados, visto ter sido demonstrado um sistema com funcionamento baseado em técnicas de extração, transformação e classificação reconhecidas na área de processamento de dados e cujo resultado produzido é explicável e auditável. Além disso, também é interessante notar que o sistema produz documentos digitais na forma de páginas intermediárias onde o investigador tem disponível o endereço virtual do conteúdo, o corpo do conteúdo visível original sem alterações ou influências das diversas nuances do código que forma uma página *Web* e exemplos de conteúdos com a exata mesma classificação originários da base de dados de treinamento. Ao guardar registros dessas páginas virtuais e anexá-las a um processo digital, um investigador passa a ser capaz de prover provas e pareceres técnicos relacionados a supostos crimes digitais cometidos em páginas da *Internet*.

No capítulo a seguir, as conclusões sobre os resultados e uso geral da aplicação e os trabalhos

futuros pretendidos.

5 CONCLUSÃO

A motivação para o desenvolvimento do trabalho de mestrado que deu origem à presente dissertação foi contribuir para uma racionalização do processo de investigação de crimes digitais utilizando técnicas de integração de sistemas, padrões de projeto de sistemas de software distribuídos, ferramentas de código aberto e processos de extração, transformação e carga, bem como classificadores baseados em aprendizagem de máquina e conceitos de enriquecimento de dados.

Com base nessa ideia geral, esta dissertação descreve os passos para criar uma arquitetura fim a fim de ferramenta de investigação composta por módulos de software que desempenham o papel de serviços com responsabilidades distintas e complementares, sendo todo o processo de extração, transformação, classificação, enriquecimento e carga orquestrado por um processo flexível e configurado pelo próprio usuário. O requisito funcional para tal arquitetura foi definido em termos de capacidade de realizar o processamento, armazenamento, classificação e enriquecimento de dados de forma distribuída para dar suporte a tarefa de investigação de páginas *web*.

Feita a proposição da arquitetura em serviços, para efeito de validação da proposta, foram desenvolvidos ao longo deste trabalho módulos de software utilizando técnicas de extração, ou seja, técnicas de raspagem de dados na obtenção eficiente de dados de fontes tão anômalas quanto páginas da Internet. Foi também implementada e testada uma lógica teoricamente ilimitada e adaptável ao contexto da base de treinamento para classificação multi-etiqueta de texto de linguagem natural utilizando aprendizado de máquina. Foram também definidos os formatos de dados a serem indexados, os processos de cadastro de novas configurações de programas *Crawler*, além da utilização confiável de enriquecimento de dados nos resultados das extrações e classificações, agregando valor aos relatórios do usuário.

Como resultado, além de mostrar o efetivo funcionamento geral da arquitetura proposta, a validação mostra também como a ferramenta de investigação proposta no trabalho permite o armazenamento e comparação de resultados por meio das etiquetas de classificação e dos dados enriquecidos, trabalho extremamente complexo quando considerado o contexto diverso e subjetivo de investigação de crimes digitais. Ainda, o aumento da ”pegada digital” promovido pelo enriquecimento favorece e facilita o trabalho do investigador ao inflar os resultados de forma coerente.

Conforme discussão apresentada na seção de resultados, a solução proposta criou um ambiente de processamento e armazenamento distribuído composto por serviços, podendo estes serem compartilhados por diversos usuários ao mesmo tempo.

Ainda, e principalmente, concluiu-se que os requisitos elencados na subseção ”Aplicação na Forense Digital” do segundo capítulo para uma eficiente aplicação da forense em crimes digitais foram atendidos ou racionalizados:

1. As técnicas reunidas na solução de software apresentadas são todas baseadas em procedimentos diários e reconhecidamente validados para trabalhos envolvendo *Big Data*;

2. Os componentes de software foram todos enumerados, apresentados e explicados em detalhes, mesmo aqueles de código aberto, estando disponíveis detalhes sobre cada elemento da solução e as formas de acesso e uso dessa;

3. O escopo para definir os profissionais aptos a interpretar os dados se tornou bastante claro: investigadores proficientes em técnicas de mineração de dados. Além disso, caso os usuários apliquem uma base de treinamentos com amostras de casos passados e já julgados e classificados, estes terão ainda mais respaldo nos relatórios e interpretações geradas;

4. O conteúdo extraído pelo sistema é apenas aquele visível e público quando se tratar de endereço de página *Web* pública. Todavia, mesmo quando se tratar de dado confidencial, o sistema pode ter o acesso restringido a nível de ambiente conteinerizado ou mesmo ser executado localmente num laboratório montado especificamente para uma investigação;

5. O uso de tecnologia de contêineres garante desacoplamento entre a solução proposta e as características da infraestrutura utilizada, deixando livre e desimpedido o planejamento por parte da equipe de suporte acerca dos recursos computacionais a serem utilizados, que terão a responsabilidade de prover capacidade suficiente;

6. Foram detalhados os processos de implantação da solução e de configuração de processos *Crawler* para execução do framework de investigação. Assim, profissionais capacitados na mineração de dados e treinados para o uso da ferramenta já estarão aptos a produzir relatórios com o uso da ferramenta.

Também, concomitantemente ao desenvolvimento da ideia central da dissertação, benefícios adicionais foram verificados na proposta deste trabalho de mestrado. Entre eles, o fato de que a adoção de elementos de software agnósticos deu origem a um projeto modular e flexível, podendo ser escalado com a replicação de instâncias dos serviços. Outro se refere ao fato de a aplicação poder ser compartilhada através da rede para vários usuários e em variados processos de investigação simultâneos. Ainda, que os modelos de aprendizado de máquina usados podem ser variados, e que a base de dados de treinamento pode ter qualquer origem e ser adaptada para qualquer indústria, não apenas a forense digital. Também, que o enriquecimento de dados é capaz de trazer segurança, imparcialidade e reduzir viés durante o estudo das amostras. Assim, para efeito da validação do trabalho de mestrado, houve aumento na eficiência, transparência e velocidade do processo de investigação, indicando a flexibilidade e a escalabilidade da proposta.

Especificamente, ficou demonstrado que a ferramenta de investigação proposta nesta dissertação foi capaz de reproduzir fim a fim a investigação de diversas páginas *web*, produzindo relatórios prontos e verificáveis para o usuário final investigador. Assim, contribuiu-se com a implementação de um sistema automatizado, multiplataforma, de código aberto, replicável e compartilhável, de buscas simples e inteligentes para apoio às atividades forenses de investigação de crimes digitais.

5.1 TRABALHOS FUTUROS

A partir do trabalho realizado, é possível evoluir no sentido de aperfeiçoar os passos do procedimento de raspagem que o serviço de busca envia para o serviço de extração. Existe, por exemplo, a possibilidade de extração de dados invisíveis, bastando alterar o conjunto de instruções. Essa mudança de abordagem poderia ser útil, por exemplo, ao tentar detectar e buscar mensagens ocultas. Também vale destacar que o mesmo sistema, porém com mudanças no procedimento de extração, poderia se beneficiar da aplicação de reconhecimento óptico de caracteres, da sigla em inglês OCR, para incluir imagens e itens gráficos à lista de elementos suportados.

Adicionalmente sobre o processo de extração, também seria interessante a adoção de outros serviços que fossem especialistas em diferentes tipos de mídias e conteúdos. Ou seja, seria possível e interessante o desenho de um serviço especialista na extração de dados de vídeos e áudios disponíveis em páginas da Internet, transcrevendo as falas e as inserindo no processo normal de indexação.

Ainda sobre o processo de extração, seria de adição valorosa a inserção de um serviço intermediário que contivesse uma lista de ”receitas” de extração para diferentes páginas de conteúdo comumente usadas em crimes digitais, como redes sociais. Esse serviço intermediário poderia ser acionado pelo serviço de buscas e indexação e guardar vários arquivos com *scripts* descrevendo os passos, por exemplo, para se autenticar numa plataforma, acessar uma conta de um indivíduo e rolar pela página principal até carregar uma determinada quantidade de informação postada. Essa adição aumentaria bastante a abrangência da ferramenta em relação à todos os serviços consumidos através de navegadores na Internet pública.

Adicionalmente, em relação ao processo de extração, seria interessante explorar programas que possam expor arquivos e pastas locais em rede e servir seu conteúdo através de páginas *web*, assim, de forma bastante simples esses documentos poderiam ser incluídos no processo de busca e indexação descritos.

Finalmente, sobre o serviço de classificação e enriquecimento, a principal melhora a ser citada seria a implementação e teste de inteligência artificial, entretanto, essa deve ser uma adição cuidadosamente estudada, pois o uso de inteligência artificial, em geral, incorre na adoção de modelos não lineares, que podem acabar por trazer vieses ocultos e difíceis de serem detectados aos processos de inferência.

Nos apêndices deste trabalho, encontram-se os principais trechos de código relacionados às mudanças feitas no projeto original da Ferramenta Fess, ao código funcional do serviço de classificação e enriquecimento, das páginas *web* e dos arquivos relacionados à construção e distribuição dos elementos de software em ambientes com as aplicações ”*Docker*” e ”*Docker Compose*” instalados.

REFERÊNCIAS BIBLIOGRÁFICAS

32000-1, I. *ISO 32000-1:2008*. 2008. Disponível em: <https://www.iso.org/standard/51502.html>.

32000-2, I. *ISO 32000-2:2020*. 2020. Disponível em: <https://www.iso.org/standard/75839.html>.

9110, R. *RFC 9110 - HTTP Semantics*. 2022. Disponível em: <https://www.rfc-editor.org/info/rfc9110>.

ADOBE. *Who Created the PDF?* 2015. Disponível em: <https://blog.adobe.com/en/publish/2015/06/18/who-created-pdf>.

ADOBE. *Adobe*. 2022. Disponível em: <https://www.adobe.com/>.

AGARWAL, A.; TOSHNIWAL, D. Smpft: Social media based profile fusion technique for data enrichment. *Computer Networks*, v. 158, p. 123–131, 2019. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128618305929>.

AHAMAD, S. Contemporary research challenges and applications of service oriented architecture. *Available at SSRN 3948566*, 2021.

AL, D. S. M. v. Publicis Groupe et. *Da Silva Moore v. Publicis Groupe et al, No. 1:2011cv01279 - Document 175 (S.D.N.Y. 2012)*. 2022. Disponível em: <https://law.justia.com/cases/federal/district-courts/new-york/nysdce/1:2011cv01279/375665/175/>.

ALMEIDA, L. C. d.; FILHO, F. L. d. C.; MARQUES, N. A.; PRADO, D. S. d.; MENDONÇA, F. L. L. d.; JR., R. T. d. S. Design and evaluation of a data collector and analyzer to monitor the covid-19 and other epidemic outbreaks. In: ROCHA, Á.; FERRÁS, C.; LÓPEZ-LÓPEZ, P. C.; GUARDA, T. (Ed.). *Information Technology and Systems*. Cham: Springer International Publishing, 2021. p. 23–35. ISBN 978-3-030-68285-9.

ALMEIDA, L. C. de; FILHO, F.; MENDONçA, F. L. de; JR., R. S. Meta-semantic search engine method proposition for transparent decision auditing. In: INSTICC. *Proceedings of the 19th International Conference on Smart Business Technologies - Volume 1: ICSBT,*. [S.l.]: SciTePress, 2022. p. 80–89. ISBN 978-989-758-587-6.

ALMEIDA, L. C. de; PRADO, D. S. do; FILHO, F. L. de C.; MENDONÇA, F. L. de; JR, R. T. de S. et al. Design and implementation of a collaborative platform model for epidemic and collective health services. *IADIS International Journal on WWW/Internet*, v. 18, n. 2, 2020.

ALMEIDA, L. C. de; PRADO, D. S. do; LUCAS, M.; MARTINS, D. A.; OLIVEIRA, J. A. de; JÚNIOR, R. T. de S. Projeto e implementação de um sistema de visualização de dados de epidemias obtidos colaborativamente. CIAWI CIACA 2020 Proceedings, 2020.

ALMEIDA, L. C. de; PRADO, D. S. do; MARQUES, N. A.; FILHO, F. L. de C.; MARTINS, L. M. C. e; SOUSA, R. T. de. Data science procedures to aggregate unstructured disease data in georeferenced spreading analysis. In: ROCHA, Á.; ADELI, H.; DZEMYDA, G.; MOREIRA, F.; CORREIA, A. M. R. (Ed.). *Trends and Applications in Information Systems and Technologies*. Cham: Springer International Publishing, 2021. p. 641–652. ISBN 978-3-030-72660-7.

AMATO, F.; CASTIGLIONE, A.; COZZOLINO, G.; NARDUCCI, F. A semantic-based methodology for digital forensics analysis. *Journal of Parallel and Distributed Computing*, v. 138, p. 172–177, 2020. ISSN 0743-7315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0743731519300644>.

AMATO, F.; COZZOLINO, G.; MOSCATO, V.; MOSCATO, F. Analyse digital forensic evidences through a semantic-based methodology and nlp techniques. *Future Generation Computer Systems*, Elsevier, v. 98, p. 297–307, 2019.

ARSHAD, H.; JANTAN, A. B.; ABIODUN, O. I. Digital forensics: review of issues in scientific validation of digital evidence. *Journal of Information Processing Systems*, Korea Information Processing Society, v. 14, n. 2, p. 346–376, 2018.

BAL, H. E.; STEINER, J. G.; TANENBAUM, A. S. Programming languages for distributed computing systems. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 21, n. 3, p. 261–322, sep 1989. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/72551.72552>.

BARROS, F.; KUHNEN, B.; SERRA, M.; FERNANDES, C. Ciências forenses: princípios éticos e vieses. *Revista Bioética*, v. 29, p. 55, 03 2021.

BECKER, H.; BRODER, A.; GABRILOVICH, E.; JOSIFOVSKI, V.; PANG, B. What happens after an ad click? quantifying the impact of landing pages in web advertising. In: *Proceedings of the 18th ACM conference on Information and knowledge management*. [S.l.: s.n.], 2009. p. 57–66.

BHATT, P.; RUGHANI, P. H. Machine learning forensics: A new branch of digital forensics. *International Journal of Advanced Research in Computer Science*, v. 8, n. 8, 2017.

BISCHKE, B.; BORTH, D.; SCHULZE, C.; DENGEL, A. Contextual enrichment of remote-sensed events with social media streams. In: *Proceedings of the 24th ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2016. (MM '16), p. 1077–1081. ISBN 9781450336031. Disponível em: <https://doi.org/10.1145/2964284.2984063>.

BLASKESI, E. *Cartorios: competência dos serviços notariais e registrais*. 2019. Disponível em: <https://jus.com.br/artigos/68267/cartorios-competencia-dos-servicos-notariais-e-registrais>.

BRASIL, P. da República do. *Código Processo Penal - Decreto-lei 3689/41 | Decreto-lei nº 3.689, de 3 de outubro de 1941*. 1941. Disponível em: <https://presrepublica.jusbrasil.com.br/legislacao/91622/codigo-processo-penal-decreto-lei-3689-41#art-156>.

BRASIL, P. da República do. *Código Processo Penal - Decreto-lei 3689/41 | Decreto-lei nº 3.689, de 3 de outubro de 1941*. 1941. Disponível em: <https://presrepublica.jusbrasil.com.br/legislacao/91622/codigo-processo-penal-decreto-lei-3689-41#art-158>.

BRASIL, P. da República do. *Lei nº 11.419/2006*. 2001. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2004-2006/2006/lei/l11419.htm>.

BRASIL, P. da República do. *LEI Nº 12.965, DE 23 DE ABRIL DE 2014*. 2014. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2011-2014/2014/lei/l12965.htm>.

BURGESS, A.; SHERMAN, B. *Metasemantics: New Essays on the Foundations of Meaning*. 2014. Disponível em: <https://ndpr.nd.edu/reviews/metasemantics-new-essays-on-the-foundations-of-meaning/>.

C. de Almeida., L.; FILHO., F.; L. de MendonÇa., F.; Sousa Jr.., R. Meta-semantic search engine method proposition for transparent decision auditing. In: INSTICC. *Proceedings of the 19th International Conference on Smart Business Technologies - ICSBT,*. [S.l.]: SciTePress, 2022. p. 80–89. ISBN 978-989-758-587-6. ISSN 2184-772X.

CADE. *CADE - Conselho Administrativo de Defesa Econômica*. 2022. Disponível em: <https://www.gov.br/cade/pt-br>.

CAMBRIDGE. *Cambridge Online Dictionary*. 2022. Disponível em: <https://dictionary.cambridge.org/dictionary/english/forensics>.

CASEY, E. *Digital Evidence and Computer Crime*. [S.l.]: Academic Press, 2018.

CAVIGLIONE, L.; WENDZEL, S.; MAZURCZYK, W. The future of digital forensics: Challenges and the road ahead. *IEEE Security & Privacy*, IEEE, v. 15, n. 6, p. 12–17, 2017.

CHADREQUE, M. d. C. *Estatística na investigação forense*. 2012. 81 p.

CODELIBS. *Fess Enterprise Search Server*. 2022. Disponível em: <https://github.com/codelibs/fess>.

CODELIBS. *Fess Enterprise Search Server Webpage*. 2022. Disponível em: <https://fess.codelibs.org/>.

COMMUNITY, W. *HTML Living Standard*. 2022. Disponível em: <https://html.spec.whatwg.org/multipage/introduction.html#abstract>.

CONJUR. *Professor é condenado por fazer piada racista*. 2022. Disponível em: <https://www.conjur.com.br/2009-mai-18/professor-condenado-piada-racista-sala-aula>.

CONJUR. *TJ-SP confirma condenação de torcedor por racismo em rede social*. 2022. Disponível em: <https://www.conjur.com.br/2022-ago-30/tj-sp-confirma-condenacao-torcedor-racismo-rede-social>.

CONWAY, C. M. How does the brain learn environmental structure? ten core principles for understanding the neurocognitive mechanisms of statistical learning. *Neuroscience & Biobehavioral Reviews*, Elsevier, v. 112, p. 279–299, 2020.

DEURSEN, A. V.; MESBAH, A.; NEDERLOF, A. Crawl-based analysis of web applications: Prospects and challenges. *Science of computer programming*, Elsevier, v. 97, p. 173–180, 2015.

DING, S.; ATTENBERG, J.; BAEZA-YATES, R.; SUEL, T. Batch query processing for web search engines. In: *Proceedings of the fourth ACM international conference on Web search and data mining*. [S.l.: s.n.], 2011. p. 137–146.

DIOUF, R.; SARR, E. N.; SALL, O.; BIRREGAH, B.; BOUSSO, M.; MBAYE, S. N. Web scraping: state-of-the-art and areas of application. In: IEEE. *2019 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2019. p. 6040–6042.

DOCKER. *Docker*. 2022. Disponível em: <https://www.docker.com/>.

DOCKER. *Docker Compose*. 2022. Disponível em: <https://docs.docker.com/compose/>.

DOCKER. *Docker hub*. 2022. Disponível em: <https://hub.docker.com/>.

DOCKER. *Docker overview*. 2022. Disponível em: <https://docs.docker.com/get-started/overview/>.

DOWNEY, A. *Think python*. [S.l.]: "O'Reilly Media, Inc.", 2012.

ELASTICSEARCH. *Elasticsearch*. 2022. Disponível em: <https://www.elastic.co/pt/elasticsearch/>.

ENNEW, C.; LOCKETT, A.; BLACKMAN, I.; HOLLAND, C. P. Competition in internet retail markets: The impact of links on web site traffic. *Long Range Planning*, Elsevier, v. 38, n. 4, p. 359–372, 2005.

EUROPA, C. da. *Convenção sobre o cibercrime*. 2001. Disponível em: <https://rm.coe.int/16802fa428>.

FACEBOOK. *Facebook*. 2022. Disponível em: <https://www.facebook.com/>.

FERRARA, E.; MEO, P. D.; FIUMARA, G.; BAUMGARTNER, R. Web data extraction, applications and techniques: A survey. *Knowledge-based systems*, Elsevier, v. 70, p. 301–323, 2014.

FOX, N.; GRAHAM, L. J.; EIGENBROD, F.; BULLOCK, J. M.; PARKS, K. E. Enriching social media data allows a more robust representation of cultural ecosystem services. *Ecosystem Services*, v. 50, p. 101328, 2021. ISSN 2212-0416. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2212041621000863>.

FRIEDMAN, J. H. Data mining and statistics: What's the connection? *Computing science and statistics*, PROCEEDINGS PUBLISHED BY VARIOUS PUBLISHERS, v. 29, n. 1, p. 3–9, 1998.

GEEKWIRE. *Big data meets crime fighting: Seattle police launch SeaStat to quickly pinpoint 'crime hotspots'*. 2022. Disponível em: <https://www.geekwire.com/2014/big-data-meets-crime-fighting-seattle-police-launch-seastat-quickly-pinpoint-crime-hotspots/>.

GIATSOGLOU, M.; VOZALIS, M. G.; DIAMANTARAS, K.; VAKALI, A.; SARIGIANNIDIS, G.; CHATZISAVVAS, K. C. Sentiment analysis leveraging emotions and word embeddings. *Expert Systems with Applications*, Elsevier, v. 69, p. 214–224, 2017.

GONÇALVES, M.; AMADIO, R. A.; GAVILAN, J. C.; SANTOS, H. W. D. Perícia forense computacional: metodologias, técnicas e ferramentas. *Revista Científica Eletrônica De Ciências Sociais Aplicadas Da Eduvale. Jaciara(MT)*, n. 7, p. 1–17, 2012.

GOOGLE. *Google*. 2022. Disponível em: <https://www.google.com/>.

GUARINO, A. Digital forensics as a big data challenge. In: ____. *ISSE 2013 Securing Electronic Business Processes: Highlights of the Information Security Solutions Europe 2013 Conference*. Wiesbaden: Springer Fachmedien Wiesbaden, 2013. p. 197–203. ISBN 978-3-658-03371-2. Disponível em: <https://doi.org/10.1007/978-3-658-03371-2_17>.

GUO, W.; LI, H.; JI, H.; DIAB, M. Linking tweets to news: A framework to enrich short text data in social media. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. [S.l.: s.n.], 2013. p. 239–249.

HAEFLIGER, S.; KROGH, G. V.; SPAETH, S. Code reuse in open source software. *Management science*, INFORMS, v. 54, n. 1, p. 180–193, 2008.

HALAVAIS, A. *Search engine society*. [S.l.]: John Wiley & Sons, 2017.

HSIAO, C. Panel data analysis—advantages and challenges. *Test*, Springer, v. 16, n. 1, p. 1–22, 2007.

HUGHES, N.; KARABIYIK, U. Towards reliable digital forensics investigations through measurement science. *Wiley Interdisciplinary Reviews: Forensic Science*, Wiley Online Library, v. 2, n. 4, p. e1367, 2020.

INMON, W. H. What is a data warehouse. *Prism Tech Topic*, v. 1, n. 1, p. 1–5, 1995.

ISAKOVIC, A. *beautiful-web*. 2022. Disponível em: <https://github.com/ialja/beautiful-web>.

JIGSAW, G. *Algorithms And Insults: Scaling Up Our Understanding Of Harassment On Wikipedia*. 2017. Disponível em: <https://medium.com/jigsaw/algorithms-and-insults-scaling-up-our-understanding-of-harassment-on-wikipedia-6cc417b9f7ff>.

JIGSAW, G. *Jigsaw Toxic Comments*. 2022. Disponível em: <https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge>.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015.

KALIPE, G. K.; BEHERA, R. K. Big data architectures: A detailed and application oriented review. *Int. Journal Innov. Technol. Explor. Eng*, v. 8, p. 2182–2190, 2019.

KARIE, N. M.; VENTER, H. S. Taxonomy of challenges for digital forensics. *Journal of Forensic Sciences*, v. 60, n. 4, p. 885–893, 2015. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/1556-4029.12809>.

KARYDA, M.; MITROU, L. Internet forensics: Legal and technical issues. In: IEEE. *Second International Workshop on Digital Forensics and Incident Analysis (WDFIA 2007)*. [S.l.], 2007. p. 3–12.

KASPERSKY. *How big data helps to catch criminals*. 2022. Disponível em: <https://www.kaspersky.com/blog/big-data-forensics/8300/>.

KASPERSKY. 2022. Disponível em: <https://www.kaspersky.com.br/>.

KASSIM, J. M.; RAHMANY, M. Introduction to semantic search engine. In: IEEE. *2009 International Conference on Electrical Engineering and Informatics*. [S.l.], 2009. v. 2, p. 380–386.

KRAMER, O. Scikit-learn. In: *Machine learning for evolution strategies*. [S.l.]: Springer, 2016. p. 45–53.

KROTOV, V.; JOHNSON, L.; SILVA, L. Tutorial: Legality and ethics of web scraping. 2020.

LEE, S. J.; SIAU, K. A review of data mining techniques. *Industrial Management & Data Systems*, MCB UP Ltd, 2001.

LI, S.; ZHANG, H.; JIA, Z.; ZHONG, C.; ZHANG, C.; SHAN, Z.; SHEN, J.; BABAR, M. A. Understanding and addressing quality attributes of microservices architecture: A systematic literature review. *Information and software technology*, Elsevier, v. 131, p. 106449, 2021.

LIMA, T. de macedo N. *Documento de Trabalho 003/2022 - Aprendizado de Máquina e antitruste*. [S.l.]: CADE - Conselho Administrativo de Defesa Econômica, 2022.

L'HEUREUX, A.; GROLINGER, K.; ELYAMANY, H. F.; CAPRETZ, M. A. M. Machine learning with big data: Challenges and approaches. *IEEE Access*, v. 5, p. 7776–7797, 2017.

MEMON, J.; SAMI, M.; KHAN, R. A.; UDDIN, M. Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr). *IEEE Access*, IEEE, v. 8, p. 142642–142668, 2020.

MJSP. *MJSP - Ministério da Justiça e Segurança Pública*. 2022. Disponível em: <https://www.gov.br/mj/pt-br>.

MORGAN, J. *The Future of Work*. [S.l.]: Wiley, 2014.

MUEHLETHALER, C.; ALBERT, R. Collecting data on textiles from the internet using web crawling and web scraping tools. *Forensic Science International*, Elsevier, v. 322, p. 110753, 2021.

MUHAMMAD, A. N.; BUKHORI, S.; PANDUNATA, P. Sentiment analysis of positive and negative of youtube comments using naïve bayes–support vector machine (nbsvm) classifier. In: IEEE. *2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE)*. [S.l.], 2019. p. 199–205.

MUKHERJEE, R.; KAR, P. A comparative review of data warehousing etl tools with new trends and industry insight. In: IEEE. *2017 IEEE 7th International Advance Computing Conference (IACC)*. [S.l.], 2017. p. 943–948.

MUKHOPADHYAY, D.; SHARMA, M.; JOSHI, G.; PAGARE, T.; PALWE, A. Experience of developing a meta-semantic search engine. In: *2013 International Conference on Cloud Ubiquitous Computing Emerging Technologies*. [S.l.: s.n.], 2013. p. 167–171.

NAGPAL, A.; GABRANI, G. Python for data analytics, scientific and technical applications. In: IEEE. *2019 Amity international conference on artificial intelligence (AICAI)*. [S.l.], 2019. p. 140–145.

NAQA, I. E.; MURPHY, M. J. What is machine learning? In: *machine learning in radiation oncology*. [S.l.]: Springer, 2015. p. 3–11.

NUNES, D.; MARQUES, A. L. P. C. Inteligência artificial e direito processual: vieses algorítmicos e os riscos de atribuição de função decisória às máquinas. In: *Revista de Processo*. [S.l.: s.n.], 2018. v. 285, n. 2018, p. 421–447.

OLSON, D. L.; LAUHOFF, G. Descriptive data mining. In: ____. *Descriptive Data Mining*. Singapore: Springer Singapore, 2019. p. 129–130. ISBN 978-981-13-7181-3. Disponível em: <https://doi.org/10.1007/978-981-13-7181-3_8>.

ORG, A. *Apache 2.0*. 2022. Disponível em: <https://www.apache.org/licenses/LICENSE-2.0>.

ORG, A. *Apache Tika - a content analysis toolkit*. 2022. Disponível em: <https://tika.apache.org/>.

ORG, F. *Flask*. 2022. Disponível em: <https://flask.palletsprojects.com/en/2.2.x/>.

ORG, G. *Bash*. 2022. Disponível em: <https://www.gnu.org/software/bash/>.

ORG, J. *Introducing JSON*. 2022. Disponível em: <https://www.json.org/json-en.html>.

ORG, L. *Linux*. 2022. Disponível em: <https://www.linux.org/>.

ORG, M. *JavaScript*. 2022. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.

ORG, P. *Package overview*. 2022. Disponível em: <https://pandas.pydata.org/docs/getting_started/overview.html>.

ORG, P. *The Python Tutorial*. 2022. Disponível em: <https://docs.python.org/3/tutorial/>.

ORG, S. *Hub and Node*. 2022. Disponível em: <https://www.selenium.dev/documentation/grid/getting_started/#hub-and-node>.

ORG, S. *Selenium*. 2022. Disponível em: <https://www.selenium.dev/>.

ORG, S. *Selenium documentation*. 2022. Disponível em: <https://www.selenium.dev/documentation/>.

ORG, S. *Selenium grid*. 2022. Disponível em: <https://www.selenium.dev/documentation/grid/>.

ORG, S. *selenium/node-firefox*. 2022. Disponível em: <https://hub.docker.com/r/selenium/node-firefox/>.

ORG, S.-L. *Scikit-Learn*. 2022. Disponível em: <https://scikit-learn.org>.

ORG, S.-L. *Scikit-Learn TfidfVectorizer*. 2022. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html>.

ORG, W. *CSV*. 2022. Disponível em: <https://www.w3.org/TR/tabular-data-primer/>.

ORG, W. *Wikipedia*. 2022. Disponível em: <https://en.wikipedia.org/wiki/Main_Page>.

ORG, Y. *YAML: YAML Ain't Markup Language*. 2022. Disponível em: <https://yaml.org/>.

PAHL, C.; LEE, B. Containers and clusters for edge cloud architectures – a technology review. In: *2015 3rd International Conference on Future Internet of Things and Cloud*. [S.l.: s.n.], 2015. p. 379–386.

POLICE, S. *SeaStat: What Is It? And How Are Police Using It to Disrupt Crime Trends?* 2014. Disponível em: <https://spdblotter.seattle.gov/2014/09/17/seastat-what-is-it-and-how-are-police-using-it-to-disrupt-crime-trends/>.

PRADO, D. S. d.; FILHO, F. L. d. C.; ALMEIDA, L. C. d.; MARTINS, L. M. C. e.; MENDONÇA, F. L. L. d.; SOUSA, R. T. d. Design of a fog controller to provide an iot middleware with hierarchical interaction capability. In: ROCHA, Á.; FERRÁS, C.; LÓPEZ-LÓPEZ, P. C.; GUARDA, T. (Ed.). *Information Technology and Systems*. Cham: Springer International Publishing, 2021. p. 280–292. ISBN 978-3-030-68285-9.

PREECE, A.; SPASIć, I.; EVANS, K.; ROGERS, D.; WEBBERLEY, W.; ROBERTS, C.; INNES, M. Sentinel: A codesigned platform for semantic enrichment of social media streams. *IEEE Transactions on Computational Social Systems*, v. 5, n. 1, p. 118–131, 2018.

PRIBERAM. *Priberam Dicionário - forense*. 2022. Disponível em: <https://dicionario.priberam.org/forense>.

PRIBERAM. *Priberam Dicionário - jurisprudência*. 2022. Disponível em: <https://dicionario.priberam.org/jurisprud%C3%AAncia>.

PRIBERAM. *Priberam Dicionário - truste*. 2022. Disponível em: <https://dicionario.priberam.org/truste>.

PROKHOROV, S.; SAFRONOV, V. Ai for ai: What nlp techniques help researchers find the right articles on nlp. In: *2019 International Conference on Artificial Intelligence: Applications and Innovations (IC-AIAI)*. [S.l.: s.n.], 2019. p. 76–765.

PURCELL, K.; RAINIE, L.; BRENNER, J. Search engine use 2012. Pew Internet & American Life Project Washington, DC, 2012.

QADIR, A. M.; VAROL, A. The role of machine learning in digital forensics. In: IEEE. *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*. [S.l.], 2020. p. 1–5.

RAHMAN, R. U.; TOMAR, D. S. A new web forensic framework for bot crime investigation. *Forensic Science International: Digital Investigation*, Elsevier, v. 33, p. 300943, 2020.

REDDY, K.; VENTER, H. The architecture of a digital forensic readiness management system. *Computers Security*, v. 32, p. 73–89, 2013. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404812001447>.

SAS. *Big Data*. 2022. Disponível em: <https://www.sas.com/en_us/insights/big-data/what-is-big-data.html>.

SAS. *Data Warehouse*. 2022. Disponível em: <https://www.sas.com/pt_br/insights/data-management/data-warehouse.html>.

SAS. *MineraÇÃo de Dados*. 2022. Disponível em: <https://www.sas.com/pt_br/insights/analytics/data-mining.html>.

SAS. *What is a data lake and why does it matter?* 2022. Disponível em: <https://www.sas.com/en_ae/insights/articles/data-management/what-is-a-data-lake-and-why-does-it-matter-.html>.

SHALAGINOV, A.; JOHNSEN, J. W.; FRANKE, K. Cyber crime investigations in the era of big data. In: IEEE. *2017 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2017. p. 3672–3676.

SHINYAMA, Y. *PDFMiner*. 2022. Disponível em: <https://pypi.org/project/pdfminer/>.

SIBIYA, G.; VENTER, H. S.; FOGWILL, T. Digital forensics in the cloud: The state of the art. In: *2015 IST-Africa Conference*. [S.l.: s.n.], 2015. p. 1–9.

SILVA, A. A. G. d. *A perícia forense no Brasil.* Tese (Doutorado) — Universidade de São Paulo, 2010.

SINDHU, K.; MESHRAM, B. Digital forensics and cyber crime datamining. Scientific Research Publishing, 2012.

SINGH, A.; THAKUR, N.; SHARMA, A. A review of supervised machine learning algorithms. In: IEEE. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. [S.l.], 2016. p. 1310–1315.

SINGH, V. P.; PAL, P. Survey of different types of captcha. *International Journal of computer science and information technologies*, v. 5, n. 2, p. 2242–2245, 2014.

SINGRODIA, V.; MITRA, A.; PAUL, S. A review on web scrapping and its applications. In: IEEE. *2019 international conference on computer communication and informatics (ICCCI)*. [S.l.], 2019. p. 1–6.

SIRISURIYA, D. S. et al. A comparative study on web scraping. 2015.

SOARES, L. K. G. S. Bachelor's Thesis, *Crimes virtuais e os limites da liberdade de expressão*. 2022. 44 p. Disponível em: <https://repositorio.pucgoias.edu.br/jspui/handle/123456789/4099>.

SRINATH, K. Python–the fastest growing programming language. *International Research Journal of Engineering and Technology*, v. 4, n. 12, p. 354–357, 2017.

STANFORD. *Stanford*. 2022. Disponível em: <https://www.stanford.edu/>.

SUAIB, M.; AKBAR, M.; HUSAIN, M. S. Digital forensics and data mining. In: *Critical concepts, standards, and techniques in cyber forensics*. [S.l.]: IGI Global, 2020. p. 240–247.

SUPRA, J. *Is Your AI Algorithm Admissible in Court? Some Things to Consider.* 2022. Disponível em: <https://www.jdsupra.com/legalnews/is-your-ai-algorithm-admissible-in-1993135/>.

TALLÓN-BALLESTEROS, A. J.; RIQUELME, J. C. Data mining methods applied to a digital forensics task for supervised machine learning. In: *Computational Intelligence in Digital Forensics: Forensic Investigation and Applications*. [S.l.]: Springer, 2014. p. 413–428.

TECHTARGET. *Most in-demand data science skills include ML, Python*. 2022. Disponível em: <https://www.techtarget.com/searchbusinessanalytics/feature/Most-in-demand-data-science-skills-include-ML-Python>.

THANAKI, J. *Python natural language processing*. [S.l.]: Packt Publishing Ltd, 2017.

THOMAS, D. M.; MATHUR, S. Data analysis by web scraping using python. In: *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. [S.l.: s.n.], 2019. p. 450–454.

UKWEN, D. O.; KARABATAK, M. Review of nlp-based systems in digital forensics and cybersecurity. In: *2021 9th International Symposium on Digital Forensics and Security (ISDFS)*. [S.l.: s.n.], 2021. p. 1–9.

VALIPOUR, M. H.; AMIRZAFARI, B.; MALEKI, K. N.; DANESHPOUR, N. A brief survey of software architecture concepts and service oriented architecture. In: IEEE. *2009 2nd IEEE International Conference on Computer Science and Information Technology*. [S.l.], 2009. p. 34–38.

VINCZE, E. A. Challenges in digital forensics. *Police Practice and Research*, Taylor & Francis, v. 17, n. 2, p. 183–194, 2016.

VU, M.-A. T.; ADALI, T.; BA, D.; BUZSÁKI, G.; CARLSON, D.; HELLER, K.; LISTON, C.; RUDIN, C.; SOHAL, V. S.; WIDGE, A. S. et al. A shared vision for machine learning in neuroscience. *Journal of Neuroscience*, Soc Neuroscience, v. 38, n. 7, p. 1601–1607, 2018.

WEIDEMAN, M.; SCHWENKE, F. The influence that javascript™ has on the visibility of a website to search engines–a pilot study. *Information Research: An International Electronic Journal*, ERIC, v. 11, n. 4, p. n4, 2006.

WEISS, S. M.; INDURKHYA, N. *Predictive data mining: a practical guide*. [S.l.]: Morgan Kaufmann, 1998.

WIRFS-BROCK, A.; EICH, B. Javascript: The first 20 years. *Proc. ACM Program. Lang.*, Association for Computing Machinery, New York, NY, USA, v. 4, n. HOPL, jun 2020. Disponível em: <https://doi.org/10.1145/3386327>.

XU, Z.; KARLSSON, M.; TANG, C.; KARAMANOLIS, C. * towards a {Semantic-Aware} file store. In: *9th Workshop on Hot Topics in Operating Systems (HotOS IX)*. [S.l.: s.n.], 2003.

XUE, F.; WU, L.; LU, W. Semantic enrichment of building and city information models: A ten-year review. *Advanced Engineering Informatics*, Elsevier, v. 47, p. 101245, 2021.

IndexUpdater.java

```
/*
 * Copyright 2012-2020 CodeLibs Project and the Others.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
 * either express or implied. See the License for the specific language
 * governing permissions and limitations under the License.
 */
package org.codelibs.fess.indexer;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.function.Consumer;
import java.util.LinkedHashMap;

import javax.annotation.PreDestroy;
import javax.annotation.Resource;

import com.github.openjson.JSONObject;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.codelibs.core.lang.StringUtil;
import org.codelibs.core.lang.ThreadUtil;
import org.codelibs.fess.Constants;
import org.codelibs.fess.crawler.Crawler;
import org.codelibs.fess.crawler.entity.AccessResult;
import org.codelibs.fess.crawler.entity.AccessResultData;
import org.codelibs.fess.crawler.entity.EsAccessResult;
import org.codelibs.fess.crawler.entity.EsUrlQueue;
import org.codelibs.fess.crawler.service.DataService;
import org.codelibs.fess.crawler.service.UrlFilterService;
import org.codelibs.fess.crawler.service.UrlQueueService;
```

```
import org.codelibs.fess.crawler.service.impl.EsDataService;
import org.codelibs.fess.crawler.transformer.Transformer;
import org.codelibs.fess.crawler.util.EsResultList;
import org.codelibs.fess.es.client.FessEsClient;
import org.codelibs.fess.es.log.exbhv.ClickLogBhv;
import org.codelibs.fess.es.log.exbhv.FavoriteLogBhv;
import org.codelibs.fess.exception.ContainerNotAvailableException;
import org.codelibs.fess.exception.FessSystemException;
import org.codelibs.fess.helper.IndexingHelper;
import org.codelibs.fess.helper.IntervalControlHelper;
import org.codelibs.fess.helper.SearchLogHelper;
import org.codelibs.fess.helper.SystemHelper;
import org.codelibs.fess.mylasta.direction.FessConfig;
import org.codelibs.fess.util.ComponentUtil;
import org.codelibs.fess.util.DocList;
import org.codelibs.fess.util.MemoryUtil;
import org.codelibs.fess.util.ThreadDumpUtil;
import org.elasticsearch.action.search.SearchRequestBuilder;
import org.elasticsearch.index.query.QueryBuilder;
import org.elasticsearch.index.query.QueryBuilders;
import org.elasticsearch.search.sort.SortOrder;

import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import org.openqa.selenium.By;
import org.openqa.selenium.Cookie;
import org.openqa.selenium.UnexpectedAlertBehaviour;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeDriverService;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.firefox.FirefoxBinary;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxOptions;
import org.openqa.selenium.firefox.FirefoxProfile;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.net.URL;



import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URLEncoder;
```

```
97 public class IndexUpdater extends Thread {
       private static final Logger logger = LogManager.getLogger(IndexUpdater.class);
99
       protected List<String> sessionIdList;
101
       @Resource
103    protected FessEsClient fessEsClient;

105    @Resource
       protected DataService<EsAccessResult> dataService;
107
       @Resource
109    protected UrlQueueService<EsUrlQueue> urlQueueService;

111    @Resource
       protected UrlFilterService urlFilterService;
113
       @Resource
115    protected ClickLogBhv clickLogBhv;

117    @Resource
       protected FavoriteLogBhv favoriteLogBhv;
119
       @Resource
121    protected SystemHelper systemHelper;

123    @Resource
       protected IndexingHelper indexingHelper;
125
       protected boolean finishCrawling = false;
127
       protected long executeTime;
129
       protected long documentSize;
131
       protected int maxIndexerErrorCount = 0;
133
       protected int maxErrorCount = 2;
135
       protected List<String> finishedSessionIdList = new ArrayList<>();
137
       private final List<DocBoostMatcher> docBoostMatcherList = new ArrayList<>();
139
       private List<Crawler> crawlerList;
141
       private HashMap<String,String> urlConfigParams;
143
       public IndexUpdater() {
145        // nothing
       }
147
```

```
    @PreDestroy
149 public void destroy() {
        if (!finishCrawling) {
151         if (logger.isInfoEnabled()) {
                logger.info("Stopping all crawler.");
153         }
            forceStop();
155     }
    }
157
    public void addFinishedSessionId(final String sessionId) {
159     synchronized (finishedSessionIdList) {
            finishedSessionIdList.add(sessionId);
161     }
    }
163
    private void deleteBySessionId(final String sessionId) {
165     try {
            urlFilterService.delete(sessionId);
167     } catch (final Exception e) {
            logger.warn("Failed to delete url filters: " + sessionId, e);
169     }
        try {
171         urlQueueService.delete(sessionId);
        } catch (final Exception e) {
173         logger.warn("Failed to delete url queues: " + sessionId, e);
        }
175     try {
            dataService.delete(sessionId);
177     } catch (final Exception e) {
            logger.warn("Failed to delete data: " + sessionId, e);
179     }
    }
181
    @Override
183 public void run() {
        if (dataService == null) {
185         throw new FessSystemException("DataService is null.");
        }
187
        if (logger.isDebugEnabled()) {
189         logger.debug("Starting indexUpdater.");
        }
191
        executeTime = 0;
193     documentSize = 0;

195     final FessConfig fessConfig = ComponentUtil.getFessConfig();
        final long updateInterval = fessConfig.
    getIndexerWebfsUpdateIntervalAsInteger().longValue();
197     final int maxEmptyListCount = fessConfig.
    getIndexerWebfsMaxEmptyListCountAsInteger();
```

```
        final IntervalControlHelper intervalControlHelper = ComponentUtil.
    getIntervalControlHelper();
199     try {
            final Consumer<SearchRequestBuilder> cb =
201                 builder -> {
                        final QueryBuilder queryBuilder =
203                             QueryBuilders
                                    .boolQuery()
205                                 .filter(QueryBuilders.termsQuery(
    EsAccessResult.SESSION_ID, sessionIdList))
                                    .filter(QueryBuilders.termQuery(
    EsAccessResult.STATUS,
207                                         org.codelibs.fess.crawler.
    Constants.OK_STATUS));
                        builder.setQuery(queryBuilder);
209                     builder.setFrom(0);
                        final int maxDocumentCacheSize = fessConfig.
    getIndexerWebfsMaxDocumentCacheSizeAsInteger();
211                     builder.setSize(maxDocumentCacheSize <= 0 ? 1 :
    maxDocumentCacheSize);
                        builder.addSort(EsAccessResult.CREATE_TIME, SortOrder.ASC)
    ;
213                 };

215         final DocList docList = new DocList();
            final List<EsAccessResult> accessResultList = new ArrayList<>();
217
            long updateTime = System.currentTimeMillis();
219         int errorCount = 0;
            int emptyListCount = 0;
221         long cleanupTime = -1;
            while (!finishCrawling || !accessResultList.isEmpty()) {
223             try {
                    final int sessionIdListSize = finishedSessionIdList.size();
225                 intervalControlHelper.setCrawlerRunning(true);

227                 updateTime = System.currentTimeMillis() - updateTime;

229                 final long interval = updateInterval - updateTime;
                    if (interval > 0) {
231                     // sleep
                        ThreadUtil.sleep(interval); // 10 sec (default)
233                 }

235                 systemHelper.calibrateCpuLoad();

237                 docList.clear();
                    accessResultList.clear();
239
                    intervalControlHelper.delayByRules();
241
                    if (logger.isDebugEnabled()) {
```

```
243                         logger.debug("Processing documents in IndexUpdater queue."
    );
                    }
245
                    updateTime = System.currentTimeMillis();
247
                    List<EsAccessResult> arList = getAccessResultList(cb,
    cleanupTime);
249                 if (arList.isEmpty()) {
                        emptyListCount++;
251                 } else {
                        emptyListCount = 0; // reset
253                 }
                    long hitCount = ((EsResultList<EsAccessResult>) arList).
    getTotalHits();
255                 while (hitCount > 0) {
                        if (arList.isEmpty()) {
257                         ThreadUtil.sleep(fessConfig.
    getIndexerWebfsCommitMarginTimeAsInteger().longValue());
                            cleanupTime = -1;
259                     } else {
                            processAccessResults(docList, accessResultList, arList
    );
261                         cleanupTime = cleanupAccessResults(accessResultList);
                        }
263                     arList = getAccessResultList(cb, cleanupTime);
                        hitCount = ((EsResultList<EsAccessResult>) arList).
    getTotalHits();
265                 }
                    if (!docList.isEmpty()) {
267                     indexingHelper.sendDocuments(fessEsClient, docList);
                    }
269
                    synchronized (finishedSessionIdList) {
271                     if (sessionIdListSize != 0 && sessionIdListSize ==
    finishedSessionIdList.size()) {
                            cleanupFinishedSessionData();
273                     }
                    }
275                 executeTime += System.currentTimeMillis() - updateTime;

277                 if (logger.isDebugEnabled()) {
                        logger.debug("Processed documents in IndexUpdater queue.")
    ;
279                 }

281                 // reset count
                    errorCount = 0;
283             } catch (final Exception e) {
                    if (errorCount > maxErrorCount) {
285                     throw e;
                    }
```

```
287                     errorCount++;
                        logger.warn("Failed to access data. Retry to access.. " +
    errorCount, e);
289                 } finally {
                        if (systemHelper.isForceStop()) {
291                         finishCrawling = true;
                            if (logger.isDebugEnabled()) {
293                             logger.debug("Stopped indexUpdater.");
                            }
295                     }
                    }
297
                    if (emptyListCount >= maxEmptyListCount) {
299                     if (logger.isInfoEnabled()) {
                            logger.info("Terminating indexUpdater. emptyListCount is
    over {}.", maxEmptyListCount);
301                     }
                        // terminate crawling
303                     finishCrawling = true;
                        forceStop();
305                     if (fessConfig.getIndexerThreadDumpEnabledAsBoolean()) {
                            ThreadDumpUtil.printThreadDump();
307                     }
                        org.codelibs.fess.exec.Crawler.addError("QueueTimeout");
309                 }

311                 if (!ComponentUtil.available()) {
                        logger.info("IndexUpdater is terminated.");
313                     forceStop();
                        break;
315                 }
                }
317
                if (logger.isDebugEnabled()) {
319                 logger.debug("Finished indexUpdater.");
                }
321         } catch (final ContainerNotAvailableException e) {
                if (logger.isDebugEnabled()) {
323                 logger.error("IndexUpdater is terminated.", e);
                } else if (logger.isInfoEnabled()) {
325                 logger.info("IndexUpdater is terminated.");
                }
327             forceStop();
            } catch (final Throwable t) {
329             if (ComponentUtil.available()) {
                    logger.error("IndexUpdater is terminated.", t);
331             } else if (logger.isDebugEnabled()) {
                    logger.error("IndexUpdater is terminated.", t);
333                 org.codelibs.fess.exec.Crawler.addError(t.getClass().getSimpleName
    ());
                } else if (logger.isInfoEnabled()) {
335                 logger.info("IndexUpdater is terminated.");
```

```
                org.codelibs.fess.exec.Crawler.addError(t.getClass().getSimpleName
    ());
337             }
                forceStop();
339         } finally {
                intervalControlHelper.setCrawlerRunning(true);
341         }

343         if (logger.isInfoEnabled()) {
                logger.info("[EXEC TIME] index update time: {}ms", executeTime);
345         }

347     }


349

351     private void processAccessResults(final DocList docList, final List<
    EsAccessResult> accessResultList, final List<EsAccessResult> arList) {
            final FessConfig fessConfig = ComponentUtil.getFessConfig();
353         final long maxDocumentRequestSize = Long.parseLong(fessConfig.
    getIndexerWebfsMaxDocumentRequestSize());
            // logger.info("Trying to print all configs in IndexUpdater:");
355         // logger.info(webConfig.getConfigParameter());

357         for (final EsAccessResult accessResult : arList) {
                if (logger.isDebugEnabled()) {
359                 logger.debug("Indexing {}", accessResult.getUrl());
                }
361             accessResult.setStatus(Constants.DONE_STATUS);
                accessResultList.add(accessResult);
363
                if (accessResult.getHttpStatusCode() != 200) {
365                 // invalid page
                    if (logger.isDebugEnabled()) {
367                     logger.debug("Skipped. The response code is {}.", accessResult
    .getHttpStatusCode());
                    }
369                 continue;
                }
371
                final long startTime = System.currentTimeMillis();
373             final AccessResultData<?> accessResultData = accessResult.
    getAccessResultData();
                if (accessResultData != null) {
375                 accessResult.setAccessResultData(null);
                    try {
377                     final Transformer transformer = ComponentUtil.getComponent(
    accessResultData.getTransformerName());
                        if (transformer == null) {
379                         // no transformer
                            logger.warn("No transformer: {}", accessResultData.
    getTransformerName());
```

```
381                         continue;
                        }
383                     @SuppressWarnings("unchecked")
                        final Map<String, Object> map = (Map<String, Object>)
    transformer.getData(accessResultData);
385                 String[] configParamsArr=urlConfigParams.get(accessResult.getUrl()
    ).split("\n");
                    HashMap <String,String> configParamsHashmap= new HashMap <String,
    String>();
387                 for (int i=0;i<configParamsArr.length;i++) {
                        String pair = configParamsArr[i];
389                     String[] keyValue = pair.split("=");
                        configParamsHashmap.put(keyValue[0], keyValue[1]);
391                 }

393                 // logger.info("Config Params inside IndexUpdater:");
                    // logger.info("["+configParamsHashmap.get("webDriver")+"]");
395                 // logger.info("["+"enable"+"]");
                    // logger.info(configParamsHashmap.get("webDriver").equals("enable
    "));
397                 // logger.info("\n\n\n");

399                 logger.info("Keys dentro da variavel map:");
                    logger.info(map.keySet());
401
                    if (configParamsHashmap.containsKey("webDriver") &&
    configParamsHashmap.containsKey("webDriverURL")){
403                     if (configParamsHashmap.get("webDriver").equals("enable") && !
    configParamsHashmap.get("webDriverURL").equals("")){

405                         // BEGIN WEBDRIVER COM AUTENTICACAO
                            logger.info("Starting Webdriver . . .");
407                         // final String remoteDriverPath = System.getenv("
    EXTRACT_URL");
                            logger.info("Webdriver URL Param: ");
409                         // logger.info(System.getenv("SELENIUM_URL"));
                            logger.info(configParamsHashmap.get("webDriverURL"));
411                         // final String remoteDriverPath = "standalone-firefox
    :4444/wd/hub";
                            FirefoxBinary firefoxBinary = new FirefoxBinary();
413                         firefoxBinary.addCommandLineOptions("--headless");
                            FirefoxOptions firefoxOptions = new FirefoxOptions();
415
                            // firefoxOptions.setBinary(firefoxBinary);
417
                            FirefoxProfile profile = new FirefoxProfile();
419

421                         // firefoxOptions.setProfile(profile);
                            WebDriver driver;
423
                                driver = new RemoteWebDriver(
```

```
425                                 new URL(configParamsHashmap.get("webDriverURL")),
    firefoxOptions);

427                         // } catch(Exception e) {
                            //     System.setProperty("webdriver.gecko.driver",
    driverPath);
429                         //     driver = new FirefoxDriver(firefoxOptions);
                            //     logger.info("\n\n\nNAO CONSIGO ACESSAR O REMOTE
    WEBDRIVER, VOU CRIAR LOCAL!!!\n\n\n");
431                         // }
                            // FirefoxDriver driver = new FirefoxDriver(firefoxOptions
    );
433                         driver.navigate().refresh();
                            driver.manage().timeouts().implicitlyWait(60, TimeUnit.
    SECONDS);
435                         driver.manage().window().maximize();

437                         driver.get(accessResult.getUrl());
                            try {
439                             Thread.sleep(20*1000);
                            } catch (InterruptedException e) {
441                             e.printStackTrace();
                            }
443                         // driver.get(baseUrl_interna);
                            final Object content = new String((driver.getCurrentUrl().
    toString() + "\n"
445                                 + driver.getTitle().toString() + "\n" + driver.
    findElement(By.tagName("body")).getText().toString()));

447                         driver.quit();
                            logger.info("Now will try to save data from webdriver to
    document content . . .");
449                         map.remove("content");
                            map.put("content", content);
451                         // END WEBDRIVER COM AUTENTICACAO

453                     }
                    }
455

457             // Dados dados = gson.fromJson(new String(output.getBytes()), Dados.
    class);
                logger.info("Verifying if inference is enabled . . .");
459             if (configParamsHashmap.containsKey("inference") &&
    configParamsHashmap.containsKey("inferenceURL")){
                    if (configParamsHashmap.get("inference").equals("enable") && !
    configParamsHashmap.get("inferenceURL").equals("")){
461                     logger.info("Inference is enabled! Processing . . .");
                        logger.info("InferenceURL:");
463                     logger.info(configParamsHashmap.get("inferenceURL"));
                        String finalTags=new String("");
465                     String enrichData=new String("");
```

```
            String url = new String(configParamsHashmap.get("inferenceURL").
    toString());
467
            // +"?input="+new String(URLEncoder.encode(new String(map.get("content
    ").toString()), "UTF-8"));
469
            // accessResult.getUrl()
471         Map<String,Object> params = new LinkedHashMap<>();
            params.put("url", accessResult.getUrl() );
473         params.put("input", new String(map.get("content").toString()));


475
            StringBuilder postData = new StringBuilder();
477         for (Map.Entry<String,Object> param : params.entrySet()) {
                if (postData.length() != 0) postData.append('&');
479             postData.append(URLEncoder.encode(param.getKey(), "UTF-8"));
                postData.append('=');
481             postData.append(URLEncoder.encode(String.valueOf(param.getValue())
    , "UTF-8"));
            }
483         byte[] postDataBytes = postData.toString().getBytes("UTF-8");


485
            URL objPost = new URL(url+new String("/predict"));
487         HttpURLConnection connPost = (HttpURLConnection) objPost.
    openConnection();
            connPost.setRequestMethod("POST");
489         connPost.setRequestProperty("Content-Type", "application/x-www-form-
    urlencoded");
            connPost.setRequestProperty("Content-Length", String.valueOf(
    postDataBytes.length));
491         connPost.setDoOutput(true);
            connPost.getOutputStream().write(postDataBytes);
493


495
            int responseCodePost = connPost.getResponseCode();
497         logger.info("POST Response Code :: " + responseCodePost);
            if (responseCodePost == HttpURLConnection.HTTP_OK) { // success
499             BufferedReader in = new BufferedReader(new InputStreamReader(
                    connPost.getInputStream()));
501             String inputLine;
                StringBuffer response = new StringBuffer();
503
                while ((inputLine = in.readLine()) != null) {
505                 response.append(inputLine);
                }
507             in.close();

509             // print result
                logger.info(response.toString());
511             finalTags=response.toString();
```

```
            } else {
513             logger.info("POST request not worked");
            }
515         logger.info("Inference result:");
            logger.info(finalTags);
517


519


521         URL objGet = new URL(url+new String("/enrichment?url="+new String(
    accessResult.getUrl())));
            HttpURLConnection connGet = (HttpURLConnection) objGet.openConnection
    ();
523         connGet.setRequestMethod("GET");


525


527         int responseCodeGet = connGet.getResponseCode();
            logger.info("Get Response Code :: " + responseCodeGet);
529         if (responseCodeGet == HttpURLConnection.HTTP_OK) { // success
                BufferedReader in = new BufferedReader(new InputStreamReader(
531                 connGet.getInputStream()));
                String inputLine;
533             StringBuffer response = new StringBuffer();

535             while ((inputLine = in.readLine()) != null) {
                    response.append(inputLine);
537             }
                in.close();
539
                // print result
541             logger.info(response.toString());
                enrichData=response.toString();
543         } else {
                logger.info("Get request not worked");
545         }
            logger.info("Enrichment data result:");
547         logger.info(enrichData);


549
            logger.info("Adding to index object . . .");
551         String finalIndexedString;
            finalIndexedString = map.get("content").toString();
553         if(new String(finalTags).equals("") == false){
                finalIndexedString += "\n\n";
555             finalIndexedString += new String("tag:"+finalTags.replaceAll(","
    ,tag:"));
            }
557         if(new String(enrichData).equals("") == false){
                finalIndexedString += "\n\n";
559             finalIndexedString += new String(enrichData);
            }
```

```
561             map.remove("content");
                map.put("content", finalIndexedString);
563             map.remove("url");
                map.put("url", "http://localhost:5000"+new String("/inference?url=")+
    accessResult.getUrl());
565             logger.info("Added! Finished inferencing step.");
            }
567
    }
569

571 logger.info("Document generated:");
    logger.info(map.toString());
573
                    if (Constants.FALSE.equals(map.get(Constants.INDEXING_TARGET))
    ) {
575                     if (logger.isDebugEnabled()) {
                            logger.debug("Skipped. This document is not a index
    target.");
577                     }
                        continue;
579                 } else {
                        map.remove(Constants.INDEXING_TARGET);
581                 }

583                 updateDocument(map);

585                 docList.add(map);
                    final long contentSize = indexingHelper.calculateDocumentSize(
    map);
587                 docList.addContentSize(contentSize);
                    final long processingTime = System.currentTimeMillis() -
    startTime;
589                 docList.addProcessingTime(processingTime);
                    if (logger.isDebugEnabled()) {
591                     logger.debug("Added the document({}, {}ms). The number of
    a document cache is {}.",
                                MemoryUtil.byteCountToDisplaySize(contentSize),
    processingTime, docList.size());
593                 }

595                 if (docList.getContentSize() >= maxDocumentRequestSize) {
                        indexingHelper.sendDocuments(fessEsClient, docList);
597                 }
                    documentSize++;
599                 if (logger.isDebugEnabled()) {
                        logger.debug("The number of an added document is {}.",
    documentSize);
601                 }
            } catch (final Exception e) {
603                 logger.warn("Could not add a doc: " + accessResult.getUrl(), e
    );
```

```
                }
            } else {
                if (logger.isDebugEnabled()) {
                    logger.debug("Skipped. No content. ");
                }
            }

        }
    }

    protected void updateDocument(final Map<String, Object> map) {
        final FessConfig fessConfig = ComponentUtil.getFessConfig();

        if (fessConfig.getIndexerClickCountEnabledAsBoolean()) {
            addClickCountField(map);
        }

        if (fessConfig.getIndexerFavoriteCountEnabledAsBoolean()) {
            addFavoriteCountField(map);
        }

        float documentBoost = 0.0f;
        for (final DocBoostMatcher docBoostMatcher : docBoostMatcherList) {
            if (docBoostMatcher.match(map)) {
                documentBoost = docBoostMatcher.getValue(map);
                break;
            }
        }

        if (documentBoost > 0) {
            addBoostValue(map, documentBoost);
        }

        if (!map.containsKey(fessConfig.getIndexFieldDocId())) {
            map.put(fessConfig.getIndexFieldDocId(), systemHelper.generateDocId(
    map));
        }

        ComponentUtil.getLanguageHelper().updateDocument(map);
    }

    protected void addBoostValue(final Map<String, Object> map, final float
    documentBoost) {
        final FessConfig fessConfig = ComponentUtil.getFessConfig();
        map.put(fessConfig.getIndexFieldBoost(), documentBoost);
        if (logger.isDebugEnabled()) {
            logger.debug("Set a document boost ({}).", documentBoost);
        }
    }

    protected void addClickCountField(final Map<String, Object> doc) {
        final FessConfig fessConfig = ComponentUtil.getFessConfig();
```

```
        final String url = (String) doc.get(fessConfig.getIndexFieldUrl());
655     if (StringUtil.isNotBlank(url)) {
            final SearchLogHelper searchLogHelper = ComponentUtil.
    getSearchLogHelper();
657         final int count = searchLogHelper.getClickCount(url);
            doc.put(fessConfig.getIndexFieldClickCount(), count);
659         if (logger.isDebugEnabled()) {
                logger.debug("Click Count: {}, url: {}", count, url);
661         }
        }
663 }

665 protected void addFavoriteCountField(final Map<String, Object> map) {
        final FessConfig fessConfig = ComponentUtil.getFessConfig();
667     final String url = (String) map.get(fessConfig.getIndexFieldUrl());
        if (StringUtil.isNotBlank(url)) {
669         final SearchLogHelper searchLogHelper = ComponentUtil.
    getSearchLogHelper();
            final long count = searchLogHelper.getFavoriteCount(url);
671         map.put(fessConfig.getIndexFieldFavoriteCount(), count);
            if (logger.isDebugEnabled()) {
673             logger.debug("Favorite Count: {}, url: {}", count, url);
            }
675     }
    }
677
    private long cleanupAccessResults(final List<EsAccessResult> accessResultList)
     {
679     if (!accessResultList.isEmpty()) {
            final long execTime = System.currentTimeMillis();
681         final int size = accessResultList.size();
            dataService.update(accessResultList);
683         accessResultList.clear();
            final long time = System.currentTimeMillis() - execTime;
685         if (logger.isDebugEnabled()) {
                logger.debug("Updated {} access results. The execution time is {}
    ms.", size, time);
687         }
            return time;
689     }
        return -1;
691 }

693 private List<EsAccessResult> getAccessResultList(final Consumer<
    SearchRequestBuilder> cb, final long cleanupTime) {
        if (logger.isDebugEnabled()) {
695         logger.debug("Getting documents in IndexUpdater queue.");
        }
697     final long execTime = System.currentTimeMillis();
        final List<EsAccessResult> arList = ((EsDataService) dataService).
    getAccessResultList(cb);
699     final FessConfig fessConfig = ComponentUtil.getFessConfig();
```

```
        if (!arList.isEmpty()) {
            final long commitMarginTime = fessConfig.
    getIndexerWebfsCommitMarginTimeAsInteger().longValue();
            for (final AccessResult<?> ar : arList.toArray(new AccessResult[arList
    .size()])) {
                if (ar.getCreateTime().longValue() > execTime - commitMarginTime)
    {
                    arList.remove(ar);
                }
            }
        }
        final long totalHits = ((EsResultList<EsAccessResult>) arList).
    getTotalHits();
        if (logger.isInfoEnabled()) {
            final StringBuilder buf = new StringBuilder(100);
            buf.append("Processing ");
            if (totalHits > 0) {
                buf.append(arList.size()).append('/').append(totalHits).append("
    docs (Doc:{access ");
            } else {
                buf.append("no docs in indexing queue (Doc:{access ");
            }
            buf.append(System.currentTimeMillis() - execTime).append("ms");
            if (cleanupTime >= 0) {
                buf.append(", cleanup ").append(cleanupTime).append("ms");
            }
            buf.append("}, ");
            buf.append(MemoryUtil.getMemoryUsageLog());
            buf.append(')');
            logger.info(buf.toString());
        }
        final long unprocessedDocumentSize = fessConfig.
    getIndexerUnprocessedDocumentSizeAsInteger().longValue();
        final IntervalControlHelper intervalControlHelper = ComponentUtil.
    getIntervalControlHelper();
        if (totalHits > unprocessedDocumentSize && intervalControlHelper.
    isCrawlerRunning()) {
            if (logger.isInfoEnabled()) {
                logger.info("Stopped all crawler threads. You have {} (>{})
    unprocessed docs.", totalHits, unprocessedDocumentSize);
            }
            intervalControlHelper.setCrawlerRunning(false);
        }
        return arList;
    }

    private void cleanupFinishedSessionData() {
        final long execTime = System.currentTimeMillis();
        // cleanup
        for (final String sessionId : finishedSessionIdList) {
            final long execTime2 = System.currentTimeMillis();
            if (logger.isDebugEnabled()) {
```

```
743                 logger.debug("Deleting document data: {}", sessionId);
            }
745             deleteBySessionId(sessionId);
            if (logger.isDebugEnabled()) {
747                 logger.debug("Deleted {} documents. The execution time is {}ms.",
    sessionId, (System.currentTimeMillis() - execTime2));
            }
749         }
        finishedSessionIdList.clear();
751 
        if (logger.isInfoEnabled()) {
753             logger.info("Deleted completed document data. The execution time is {}
    ms.", (System.currentTimeMillis() - execTime));
        }
755     }

757     private void forceStop() {
        systemHelper.setForceStop(true);
759         for (final Crawler crawler : crawlerList) {
            crawler.stop();
761         }
    }
763 
    public long getExecuteTime() {
765         return executeTime;
    }
767 
    public List<String> getSessionIdList() {
769         return sessionIdList;
    }
771 
    public void setSessionIdList(final List<String> sessionIdList) {
773         this.sessionIdList = sessionIdList;
    }
775 
    public void setFinishCrawling(final boolean finishCrawling) {
777         this.finishCrawling = finishCrawling;
    }
779 
    public long getDocumentSize() {
781         return documentSize;
    }
783 
    @Override
785     public void setUncaughtExceptionHandler(final UncaughtExceptionHandler eh) {
        super.setUncaughtExceptionHandler(eh);
787     }

789     public static void setDefaultUncaughtExceptionHandler(final
    UncaughtExceptionHandler eh) {
        Thread.setDefaultUncaughtExceptionHandler(eh);
791     }
```

```
793     public void setMaxIndexerErrorCount(final int maxIndexerErrorCount) {
            this.maxIndexerErrorCount = maxIndexerErrorCount;
795     }

797     public void addDocBoostMatcher(final DocBoostMatcher rule) {
            docBoostMatcherList.add(rule);
799     }

801     public void setCrawlerList(final List<Crawler> crawlerList) {
            this.crawlerList = crawlerList;
803     }

805     public void putUrlConfigParams(final HashMap<String, String> mapConfigs){
            urlConfigParams=new HashMap<String, String>();
807         urlConfigParams.putAll(mapConfigs);
        }
809 }
```

fess-lucas-build-master/Dockerfile

```
  # imagem que enxerga o repositorio oficial do Maven
2 FROM maven:3.6.0-jdk-11-slim

4 WORKDIR /app
  COPY . .
6 ADD settings.xml /root/.m2/settings.xml

8 WORKDIR dependencies
  WORKDIR lucas-fess-jlha
10 RUN mvn clean install -q -Dmaven.test.skip=true
  WORKDIR ..
12 WORKDIR lucas-fess-jodconverter-core
  RUN mvn clean  install -q -Dmaven.test.skip=true
14 WORKDIR ..
  WORKDIR lucas-fess-crawler-parent
16 RUN mvn clean  install -q -Dmaven.test.skip=true
  WORKDIR ..
18 WORKDIR lucas-fess-crawler
  RUN mvn clean  install -q -Dmaven.test.skip=true
20 WORKDIR ..
```

```
WORKDIR lucas-fess-crawler-lasta
RUN mvn clean  install -q -Dmaven.test.skip=true
WORKDIR ..
WORKDIR lucas-fess-crawler-es
RUN mvn clean  install -q -Dmaven.test.skip=true


WORKDIR /app
# WORKDIR lucas-fess-server
# empacota a aplica  o e gera o binario .deb
RUN mvn package -q -Dmaven.test.skip=true && \
    mvn jdeb:jdeb

# parent crawler lasta es


# essa imagem n esta enxergando o repositorio oficial do Maven
    (mas tem ubuntu p facilitar instalacao .deb files)
FROM openjdk:11-jre


ARG FESS_VERSION=0.0.1-RC17
ARG ELASTIC_VERSION=7.8.1
ENV FESS_APP_TYPE docker
ARG CACHEBUST=1

RUN apt-get update && \
     apt-get install -y --no-install-recommends imagemagick=
   newest procps=newest unoconv=newest ant=newest curl=newest
   && \
     apt-get clean && rm -rf /var/lib/apt/lists/*

RUN groupadd -g 1000 elasticsearch && \
     groupadd -g 1001 fess && \
     useradd -u 1000 elasticsearch -g elasticsearch && \
     useradd -u 1001 fess -g fess

# copia o binario do fess gerado no estagio de build
# hadolint ignore=DL3022
COPY --from=0 /app/target/releases/lucas-fess-server-
   FESS_VERSION.deb/tmp/fess-{FESS\_VERSION}.deb
```

```
58
  # WORKDIR /app
60 # baixa o binario do Elastic Search
  # elastic search versao 7.8.1
62 COPY /elasticsearch/elasticsearch-
    ELASTIC_VERSION - amd64.deb/tmp/elasticsearch-{ELASTIC\
    _VERSION}-amd64.deb

64 # set -x enables a mode of the shell where all executed
    commands are printed to the terminal
  RUN  set -x && \
66     # instala elasticsearch
      dpkg -i /tmp/elasticsearch-
    ELASTIC_VERSION - amd64.deb  rm - rf /tmp/elasticsearch-{ELASTIC\
    _VERSION}-amd64.deb  && \
68     # instala fess
      dpkg -i /tmp/fess-FESS_VERSION.deb  rm - rf /tmp/fess-{FESS\
    _VERSION}.deb && \
70     # outras configs
      mkdir /opt/fess && \
72     chown -R fess.fess /opt/fess && \
      sed -i -e "s#FESS\_CLASSPATH=FESS_CONF_PATH :FESS\
    _CLASSPATH#FESS\_CLASSPATH=FESS_OVERRIDE_CONF_PATH :
    FESS\_CONF\_PATH:
    FESS_CLASSPATHg" /usr/share/fess/bin/fess  echo "exportFESS_APP_TYPE =
    FESS\_APP\_TYPE" >>  /usr/share/fess/
    bin/fess.in.sh && \
74     echo "export FESS\_OVERRIDE\_CONF\_PATH=/opt/fess" >>  /
    usr/share/fess/
    bin/fess.in.sh && \
      apt-get clean && rm -rf /var/lib/apt/lists/*
76
  # copia os plugins do ES gerados no estagio de build dos
    plugins
78 # hadolint ignore=DL3022
  COPY --from=0 /app/plugins /usr/share/elasticsearch/plugins
80 # copia 2 plugins pendentes para tirar erro de compilacao do
    Fess
  COPY --from=0 /app/ajustes-plugins/commons-codec-1.13.jar /usr
    /share/elasticsearch/plugins/configsync
82 COPY --from=0 /app/ajustes-plugins/guava-23.0.jar /usr/share/
    elasticsearch/plugins/minhash
```

```
COPY --from=0 /app/elasticsearch/config /etc/elasticsearch
RUN chown root:elasticsearch /etc/elasticsearch/* && \
    chmod 660 /etc/elasticsearch/*

# E necessario instalar os binarios do firefox para que o webdriver funcione, mesmo que seja o remoteWebriver
RUN echo "deb http://deb.debian.org/debian/ unstable main contrib non-free" >> /etc/apt/sources. list.d/debian. list
RUN apt-get update
RUN apt-get install -y --no-install-recommends firefox

WORKDIR /usr/share/fess
EXPOSE 8080 9200 9300

# hadolint ignore=DL3002
USER root

COPY --from=0 /app/run.sh /usr/share/fess/run.sh
ENTRYPOINT ["sh", "/usr/share/fess/run.sh"]
```

flask-svm/app.py

```
from flask import Flask, request, render_template
app = Flask(__name__)

import time
from sklearn.feature_extraction.text import TfidfVectorizer
import time
from sklearn import svm
from sklearn.metrics import classification_report
import pandas as pd
import sqlite3
from pathlib import Path
import sys

dbpath= str(Path.cwd()).replace("\\","/")+"/"
```

```
dbname='db/inferences.db'
db=dbpath+dbname

models={}

# Create feature vectors
vectorizer = TfidfVectorizer(min_df = 5,
                                    max_df = 0.8,
                                    sublinear_tf = True,
                                    use_idf = True)


def dict_factory(cursor, row):
    d = {}
    for idx, col in  enumerate(cursor.description):
        d[col[0]] = row[idx]
    return d

# @app.before_first_request
def initialize_model():
    global models
    global vectorizer
    global dimensions_names
    # train Data
    # trainData = pd.read_csv("https://raw.githubusercontent.com/Vasistareddy/sentiment_analysis/master/data/train.csv")
    trainData = pd.read_csv("trainning/small-toxic-comments.csv", encoding='utf8',engine='python', error_bad_lines=False)

    # Save to database
    # 1 - delete table is exists
    conn = sqlite3.connect(db)
    conn.row_factory = dict_factory
    cur = conn.cursor()
    cur.execute('DROP TABLE IF EXISTS traindata;')
    conn.commit()
    # 2 - list dimensions of current dataset and create table with columns with the same name (contet, dimension1, dimension2, ...)
    query='''CREATE TABLE traindata (
```

```
51          Content TEXT NOT NULL, '''
      for (_, dimension) in  enumerate(trainData):
53        if not dimension=='Content':
            query+=dimension
55          query+= " TEXT NOT NULL, "
      # remove extra comma and space
57    query=query[:-2]
      query+=");"
59    conn.row_factory = dict_factory
      cur = conn.cursor()
61    cur.execute(query)
      conn.commit()
63    trainData.to_sql('traindata', conn, if_exists='replace',
    index=False)
      conn.commit()
65    conn.close()


67

      print(trainData.iloc[:10])
69    print("\n\n")
      train_vectors = vectorizer.fit_transform(trainData['
    Content'])
71    print(train_vectors)
      print("\n")
73    # Perform classification with SVM, kernel=linear
      for (_, dimension) in  enumerate(trainData):
75        if not dimension=='Content' and not  dimension=='id' :
            classifier_linear = svm.SVC(kernel='linear')
77          classifier_linear.fit(train_vectors, trainData[
    dimension])
            models[dimension]=classifier_linear
79


81 @app.route('/inference', methods=['GET'])
  def inference():
83   url= str(request.args.get('url'))
    conn = sqlite3.connect(db)
85  conn.row_factory = dict_factory
    # conn.isolation_level = 'EXCLUSIVE'
87  # conn.execute('BEGIN EXCLUSIVE')
    cur = conn.cursor()
```

```
89    all_data = cur.execute("SELECT * FROM `past_inferences`
       where url = '"+
       str(url)+"';").fetchone()
      conn.close()
91    final_data={}
      if 'host.docker.internal' in all_data['url']:
93      final_data['url']=all_data['url'].replace('host.docker.
       internal','localhost')
      else:
95      final_data['url']=all_data['url']

97    if not all_data['similars']:
         final_data['similars'] = "THERE NO RESULTS LIKE THIS
       SAMPLE"
99    else:
        final_data['similars'] =all_data['similars']
101
      if not all_data['tags']:
103      final_data['tags'] = "THERE NO TAGS IDENTIFIED TO
       CLASSIFY THIS SAMPLE"
      else:
105     final_data['tags'] =all_data['tags']

107   final_data['data'] =all_data['data']

109   if all_data is not None:
        return render_template('inferences_page.html',
111     data= final_data['data'],
        website= final_data['url'],
113     tags= final_data['tags'],
        similar_inferences=final_data['similars']
115     )

117 @app.route('/enrichment', methods=['GET'])
    def enrichment():
119   url= str(request.args.get('url'))
      conn = sqlite3.connect(db)
121   conn.row_factory = dict_factory
      # conn.isolation_level = 'EXCLUSIVE'
123   # conn.execute('BEGIN EXCLUSIVE')
      cur = conn.cursor()
```

```
125   all_data = cur.execute("SELECT similars FROM `
      past_inferences` where url = '"+
      str(url)+"';").fetchone()
      conn.close()
127   final_data=""

129   if not all_data['similars']:
         final_data = ""
131   else:
        final_data = str(all_data['similars'])
133
      if all_data is not None:
135     return final_data


137


139 @app.route("/predict", methods=['POST'])
    def predict_svm():
141   global models
      # Create the pandas DataFrame with column name is provided
       explicitly
143   data=[request.form.get('input')][0]
      url=[request.form.get('url')][0]
145
      final_inferences=""
147
      new_df = pd.DataFrame([data], columns=['Content'])
149   inference_vector = vectorizer.transform(new_df['Content'])
      tags=[]
151   dict_query_similars={}
      # Perform classification with SVM, kernel=linear
153   for dimension, model in models.items():
        print(model.predict(inference_vector))
155     classification=model.predict(inference_vector)[0]
        if  str(classification)=='0':
157       dict_query_similars[dimension]="0"
          continue
159     elif  str(classification)=='1':
          dict_query_similars[dimension]="1"
161       tags.append(dimension)
```

```
163   if  len(tags)>1:
        final_inferences= ",".join(tags)
165   elif  len(tags)==1:
        final_inferences= tags[0]
167   else:
        final_inferences= ""
169
      query='SELECT * FROM `traindata` where '
171   for dimension,_ in models.items():
        if not dimension=='id' and not dimension=='Content':
173       query+=dimension
          query+="="
175       query+=dict_query_similars[dimension]
          query+=" AND "
177   # remove extra " AND "
      query=query[:-5]
179   # we want max of 10 similar results
      query+=" LIMIT 10;"
181   conn = sqlite3.connect(db)
      conn.row_factory = dict_factory
183   cur = conn.cursor()
      all_data = cur.execute(query).fetchall()
185   conn.close()
      query_results=[]
187   similars=""
      if all_data:
189     for row in all_data:
          query_results.append(row)
191     for row in query_results:
          similars+=row['Content']
193       similars+=' || '

195   conn = sqlite3.connect(db)
      conn.row_factory = dict_factory
197   cur = conn.cursor()

199   inference_data= [[ str(url),  str(data),
       str(final_inferences), str(similars)]]
      inference_df=pd.DataFrame(inference_data, columns=['url', '
       data', 'tags', 'similars'])
201   print("Dados a serem inseridos: ", flush=True)
```

```
    print(inference_df.head() , flush=True)
    print(inference_df.to_sql('past_inferences', conn, if_exists
     ='append', index=False, method='multi'), flush=True)
    conn.commit()
    conn.close()

    return final_inferences


if __name__ == "__main__":
    initialize_model()
    app.run(
          debug=True,
          host='0.0.0.0',
          port=80
    )
```

inferences_page.html

```
{% extends "base.html" %}


{% block content %}


    <div class="row">
      <div class="col-lg-12">
    <div class="jumbotron">
      <p>{{ data }}</p>
      <p><a href='{{ website }}' class="btn btn-warning btn-lg">
     Go to website</a></p>
    </div>

    <div class="jumbotron">
      <p>{{ tags }}</p>
      </div>

      <div class="jumbotron">
```

```
      <p>{{ similar_inferences }}</p>
20    </div>
  </div>
22 </div>
  {% endblock %}
```

base.html

```
1 <!DOCTYPE html>
  <html>
3
    <head>
5     <meta name="description" content="Just a test web app
    created with Flask and Bootstrap, deployed on Heroku." />
      <meta name="viewport" content="width=device-width, initial
    -scale=1.0, maximum-scale = 1.0">
7     <!-- making sure the app always find the static folder
    with bootstrap -->
      <link href="{{ url_for('.static', filename='bootstrap/css/
    bootstrap.css') }}" rel="stylesheet" />
9     <link href="{{ url_for('.static', filename='css/custom.css
    ') }}" rel="stylesheet" />

11    <link rel="shortcut icon" href="{{ url_for('static',
    filename='favicon.ico') }}">
      <title>Inferences Webpage - {% block title %}{% endblock
    %}</title>
13  </head>

15  <body>

17



19



21


23 <div class="page-header">
```

```
    <div class="item">
25      <h1>INFERENCES WEBPAGE<br /><small>{{ self.title() }}</
     small></h1>
    </div>
27 </div>


29


31  <div class="container">

33      {% block content %}
        {% endblock %}
35

    </div>
37 <!--
  <nav class="navbar navbar-default navbar-fixed-bottom" role="
     navigation"
     id="bottom-nav">
39

      <br /><p class="pull-right"><a href="#">Back to top</a></p
     >
41      <p>Created with <a href="http://flask.pocoo.org">Flask</
     a>, <a href="http://getbootstrap.com">Bootstrap</a>, <a
     href="http://www.bootply.com/62603">Bootply</a>, <a href="
     http://glyphicons.com/">Glyphicons</a>, <a href="http://
     marcoceppi.github.io/bootstrap-glyphicons/">Bootstrap
     Glyphicons</a> and a lot of cat hair.</p>

43 </nav>


45


47  <script src="{{ url_for('.static', filename='js/jquery
     -1.10.2.min.js') }}"></script>
      <script src="{{ url_for('.static', filename='js/jquery.
     validate.js') }}"></script>
49  <script src="{{ url_for('.static', filename='bootstrap/js/
     bootstrap.js') }}"></script>
      <script src="{{ url_for('.static', filename='js/myscript.
     js') }}"></script> -->
51
```

```
    </body>
</html>
```

flask-svm/Dockerfile

```
# syntax=docker/dockerfile:1

FROM python:3.8-slim-buster

WORKDIR /python-docker

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

# CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]

CMD [ "python3", "app.py"]
```

investigation_tool/docker-compose.yml

```
version: "3"
services:

    extract-service-slave:
        image: selenium/node-firefox:4.0.0-beta-3-20210426
        # volumes:
        # - /dev/shm:/dev/shm
        depends_on:
            - extract-service-master
        networks:
            - selenium-net
        environment:
            - SE_EVENT_BUS_HOST=extract-service-master
```

```
14              - SE_EVENT_BUS_PUBLISH_PORT=4442
                - SE_EVENT_BUS_SUBSCRIBE_PORT=4443
16          # ports:
            #     - "6902:5900"
18
    extract-service-master:
20          image: selenium/hub:4.0.0-beta-3-20210426
            ports:
22              - "4442:4442"
                - "4443:4443"
24              - "4444:4444"
            networks:
26              - investigate-net
                - selenium-net
28
#   standalone-firefox:
30  #     image: selenium/standalone-firefox
#       ports:
32  #         - "4444:4444"
#       networks:
34  #         - investigate-net

36      search-service:
            build: fess-lucas-build-master/.
38          # links:
            #     - selenium-hub
40          networks:
                - investigate-net
42          stdin_open: true
            tty: true
44          entrypoint: ./run.sh
            # environment:
46          #     EXTRACT_URL: "http://extract-service-master
    :4444"
            #     CLASSIFY_URL: "http://classify-service"
48          ports:
                - "8080:8080"
50
    classify-service:
52          build: flask-svm/.
            ports:
```

```
            - "5000:80"
        networks:
            - investigate-net

networks:
  investigate-net:
  selenium-net:
```

test_webpages/app.py

```
from flask import Flask, request, render_template
app = Flask(__name__)

import time
from pathlib import Path
import sys


@app.route("/1")
def kill_page():
  return ''' <!DOCTYPE html>
<html>
<head>
<title>I will kill you</title>
</head>
<body>
<h1>I will kill you</h1>
</body>
</html>'''


@app.route("/2")
def shit_page():
  return ''' <!DOCTYPE html>
<html>
<head>
<title>You are full of shit</title>
</head>
```

```
<body>
<h1>You are full of shit</h1>
</body>
</html>'''


@app.route("/3")
def love_page():
  return ''' <!DOCTYPE html>
<html>
<head>
<title>I love my dogs</title>
</head>
<body>
<h1>I love my dogs</h1>
</body>
</html>'''


@app.route('/4', methods=['GET'])
def dynamic_page_1():
  return render_template('dynamic_page.html',
    data= 'Who wants to join us?',
    website= 'www.itdoesntexist.com',
    tags= 'Mussolini for president again',
    similar_inferences='The comunists should be taken from
   homes and beaten at public parks'
    )



@app.route('/5', methods=['GET'])
def dynamic_page_2():
  return render_template('dynamic_page.html',
    data= 'Click here to subscribe to our newsletter?',
    website= 'www.itdoesntexist.com',
    tags= 'The world is white, blacks do not have their turn',
    similar_inferences='We are the bright force that holds the
    world, the new race, and we need help to purge the world
   from the dark'
```

```
    )


if __name__ == "__main__":
    app.run(
            debug=True,
            host='0.0.0.0',
            port=80
    )
```

test_webpages/Dockerfile

```
# syntax=docker/dockerfile:1

FROM python:3.8-slim-buster

WORKDIR /python-docker

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

# CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]

CMD [ "python3", "app.py"]
```

test_webpages/docker-compose.yml

```
version: '3'
services:
    test_webpages:
        build: .
        ports:
```

```
6                 - "7000:80"
```