Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Ferramenta para busca e categorização automática de materiais didáticos

Rodolpho Nazareth Rossete de Souza

JUIZ DE FORA
AGOSTO, 2022

Ferramenta para busca e categorização automática de materiais didáticos

Rodolpho Nazareth Rossete de Souza

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Fabrício Martins Mendonça

JUIZ DE FORA

AGOSTO, 2022

Ferramenta para busca e categorização automática de materiais didáticos

Rodolpho Nazareth Rossete de Souza

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Fabrício Martins Mendonça
Doutor em Ciência da Informação

Jairo Francisco de Souza
Doutor em Informática

Victor Ströele de Andrade Menezes
Doutor em Engenharia de Sistemas e Computação

Juiz de Fora

05 DE AGOSTO, 2022

Dedico este trabalho aos meus pais, Delma e João Bosco, quem são minha base e o motivo desta conquista.

Resumo

O volume de dados disponibilizados na web e a facilidade de publicação dos mesmos têm levado a um crescimento acelerado e desestruturado de materiais educacionais no contexto web. As informações estão espalhadas na web, muitas vezes, de forma não organizada, causando um certo desconforto ao usuário durante suas pesquisas, consumindo um grande tempo de busca e afetando de alguma forma o processo de aprendizagem. Acrescenta-se ainda o fato de que recursos educacionais na web são cada vez mais necessários como forma de suporte ao processo de ensino e aprendizagem. Nesse contexto, emerge a necessidade de ferramentas que possam otimizar o tempo e auxiliar na obtenção de materiais educacionais. O presente trabalho apresenta o desenvolvimento de uma plataforma computacional que possibilita buscas e classificações de forma automática a conteúdos educacionais disponibilizados na internet, a fim de auxiliar professores e alunos nesse processo. Como metodologia de pesquisa foi realizado um levantamento de trabalhos relacionados ao tema de classificação de documentos existentes na web com a finalidade de apoiar o processo de ensino e aprendizagem em diferentes áreas do conhecimento, além de propor uma arquitetura modular como base para desenvolvimento da ferramenta. Tal arquitetura é composta pelos módulos de interface usuária, comunicação, indicação de conteúdos, obtenção de dados, persistência de dados e categorização. Os resultados encontrados mostram que a ferramenta desenvolvida minimiza o esforço de busca por materiais educacionais pelo usuário, além de trazer uma arquitetura modular capaz de agregar novas features no processo de classificação. Foi possível também, entender melhor o comportamento dos métodos de aprendizagem de máquina utilizados para classificação dos documentos educacionais, os quais usaram os algoritmos de *Naive Bayes*, *Decision Tree* e *Support Vector Machine*. Nos testes de acurácia realizados pela pesquisa, o algoritmo *Naive Bayes* obteve os melhores resultados, independentemente das classes (categorias) e conjuntos de dados utilizados nos testes.

Palavras-chave: Monografia, Aprendizagem de Maquina,Naive Bayes, Support Vec-

tor Machine, SVM, Decision Tree, Processamento de Linguagem Natural, PLN, Categorização.

Abstract

The volume of data available on the web and the ease of publishing them have led to an accelerated and unstructured growth of educational materials in the web context. Information is spread on the web, often in an unorganized way, causing some discomfort to the user during their research, consuming a lot of search time and somehow affecting the learning process. Added to this is the fact that educational resources on the web are increasingly necessary as a way of supporting the teaching and learning process. In this context, the need for tools that can optimize time and assist in obtaining educational materials emerges. The present work presents the development of a computational platform that allows searches and classifications automatically to educational content available on the internet, in order to assist teachers and students in this process. As a research methodology, a survey of works related to the topic of classification of existing documents on the web was carried out in order to support the teaching and learning process in different areas of knowledge, in addition to proposing a modular architecture as a basis for the development of the tool. This architecture is composed of user interface modules, communication, content indication, data collection, data persistence and categorization. The results found show that the developed tool minimizes the effort of searching for educational materials by the user, in addition to bringing a modular architecture capable of adding new features in the classification process. It was also possible to better understand the behavior of the machine learning methods used to classify educational documents, which used the *Naive Bayes*, *Decision Tree* and *Support Vector Machine* algorithms. In the accuracy tests carried out by the research, the *Naive Bayes* algorithm obtained the best results, regardless of the classes (categories) and data sets used in the tests.

Keywords: Monograph, Machine Learning, Naive Bayes, Support Vector Machine, SVM, Decision Tree, Natural Language Processing, NLP, Categorization.

Agradecimentos

Agradeço imensamente meus pais, Delma e João Bosco, que não mediram esforços para que eu alcançasse esse sonho, de me formar . Obrigada por toda renúncia, luta e força, vocês são meu alicerce. Amo muito vocês

Agradeço também a minha noiva, Alessandra, por todos os momentos de incentivo e por nunca desistir de mim, eu te amo.

Aos meus amigos João Victor e Maxwell, que estiveram do meu lado em todos os momentos durante a graduação, agradeço imensamente a por todo companheirismo, partilha e apoio.

Não poderia deixar de agradecer ao meu orientador Fabrício que sempre esteve comigo durante todo este percurso para a conclusão do curso. Muito obrigado por toda a ajuda e conhecimento.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

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

Madre Teresa de Calcutá

Conteúdo

Lista de Figuras **8**

Lista de Tabelas **9**

Lista de Abreviações **10**

1 Introdução **11**
1.1 Apresentação do Tema e sua contextualização 11
1.2 Problema 12
1.3 Justificativa 13
1.4 Objetivos 13

2 Fundamentação Teórica **15**
2.1 Processamento de Linguagem Natural 15
2.2 Abordagens Estatísticas 17
2.3 Web Crawler 19
2.4 Machine Learning 20
2.4.1 Algoritmos de aprendizado supervisionado 22
2.5 Trabalhos Relacionados 23

3 Metodologia **30**
3.1 Classificação da Pesquisa 30
3.2 Tecnologias utilizadas 31
3.3 Arquitetura Proposta 31
3.4 Aquisição dos conteúdos web 36
3.5 Categorização do conteúdo 37

4 Resultados **40**
4.1 Análise da Ferramenta Desenvolvida 40
4.2 Classificadores 43

5 Conclusões **48**

Bibliografia **51**

Lista de Figuras

2.1 Exemplo análise de sintaxe 16
2.2 Fluxo de execução de um Web Crawler 20
2.3 Decision Tree criada pelo aluno como exemplo 23
2.4 Gráfico SVM criado pelo aluno como exemplo 24

3.1 Arquitetura proposta pelo aluno 32
3.2 *Select box* para escolha de categoria 32
3.3 *Select box* com as opções de categoria 33
3.4 Lista com as páginas retornadas após a classificação 33
3.5 Fluxo de captura de informações para o site Só História 37
3.6 Métodos de *Machine Learning* utilizados 38
3.7 Endpoint criado para acessar o módulo de classificação 38
3.8 Fluxo de recebimento e retorno dos dados já classificados 39

4.1 *Select box* para escolha de categoria 40
4.2 *Select box* com as opções de categoria 41
4.3 Lista com as páginas retornadas para a categoria de Grécia Antiga 41
4.4 Lista com as páginas retornadas para a categoria de Idade Média 42
4.5 Exemplo de amostragem usando a técnico de *k-fold* 43
4.6 Acurácia dos métodos de classificação para todas as classes 44
4.7 Matriz de confusão gerada após os testes. 45
4.8 Acurácia dos métodos de classificação divididos em subdomínios 46

5.1 Nova arquitetura proposta pelo aluno 49

Lista de Tabelas

2.1 Tabela Trabalhos Relacionados 25

4.1 Tabela comparativa com as acurácias dos métodos de *Machine Learning* . 47

Lista de Abreviações

DCC Departamento de Ciência da Computução

UFJF Universidade Federal de Juiz de Fora

1 Introdução

1.1 Apresentação do Tema e sua contextualização

A Internet hoje se tornou a principal forma de difusão de informações, se tornando uma forma rápida e prática de propagar recursos importantes (SANTOS, 2010).

Como apresentado por Santana et al. (2020), a Web pode ser considerada a fonte mais completa e abrangente de materiais educacionais do mundo e, no cenário atual, o uso destes recursos educacionais disponíveis são cada vez mais necessários no processo de ensino e aprendizagem, aumentando as buscas por essas informações.

Porém a procura por esses materiais se torna muitas vezes custosa, pelo fato das informações estarem espalhadas pela rede de uma forma, às vezes, não muito organizadas. Esta situação acaba, por muito, causando uma certa confusão ao usuário durante as suas pesquisas, consumindo um grande tempo e afetando diretamente na fluidez de sua aprendizagem (WEBER et al., 2015; NASTASE; STRUBE, 2008).

Por este motivo se vê necessário a criação de recursos computacionais que possam otimizar o tempo e auxiliar na obtenção de tais informações, agrupando esses dados em um só local e categorizando as informações requeridas na busca, a fim de atender as necessidades do usuário final e melhorar a experiência de uso.

Um dos objetivos deste trabalho é o desenvolvimento de uma plataforma que consiga realizar buscas e classificações, de forma automática, a conteúdos educacionais disponibilizados na internet, auxiliando professores e alunos durante a obtenção destes materiais, como suporte no processo de ensino e aprendizagem.

A plataforma proposta pode ser caracterizada como um software de busca que permite realizar consultas em bases de dados abertas que contenham recursos educacionais, classificando através de categorias referentes ao assunto que cada recurso contem, através de métodos de aprendizagem de máquina. Para que essa categorização possa ser realizada existe a necessidade inicial de se realizar um pré-processamento das informações obtidas. Esta etapa inicial é feita utilizando técnicas de Processamento de Linguagem

Natural (PLN), com normalização dos recursos para que a classificação dos mesmos possa ser feita durante o fluxo de categorização.

O presente trabalho também visa a realização de testes comparativos entre métodos de classificação automática dos documentos coletados nas base de dados, trazendo buscando trazer um melhor entendimento do comportamento destes métodos.

1.2 Problema

Com o grande número de conteúdos de dados distribuídos na web e a necessidade de se obter informações de forma mais ágil e eficiente, surge uma grande demanda por meios de realizar a busca destes dados.

Este problema é enfrentado por muitas áreas, entre elas a área educacional, que se viu muito impactada com o surgimento da pandemia do novo coronavírus COVID-19, no início do ano de 2020. Este acontecimento acabou forçando as redes de ensino públicas e privadas a suspenderem as aulas de formas presenciais, criando uma demanda muito grande por conteúdos educacionais contidos na Internet (CORDEIRO, 2020).

Atualmente, a tarefa de busca por materiais educacionais tem se tornado cada mais um ponto de interesse entre os estudos realizados, como apresentado por Medeiros et al. (2021) que realiza um mapeamento sobre os principais recursos educacionais disponibilizado, validando a qualidade dos dados trazidos por essas plataformas.Estas informações disponibilizadas muitas das vezes são apresentadas de forma desorganizada e distribuída. Esse cenário atual vem gerando dificuldades na obtenção de informações com boa qualidade, que possam ser efetivas no auxílio ao processo de ensino e aprendizagem.

Esta situação acaba causando um certo desconforto ao usuário durante as suas pesquisas, consumindo um grande tempo e afetando diretamente na fluidez de sua aprendizagem e proporcionando uma experiência ruim durante o uso de muitas plataformas por parte dos usuários.

1.3 Justificativa

Tendo em vista os problemas ocasionados pelo crescimento do número de dados, apresenta-se a necessidade de recursos computacionais que possam otimizar o tempo e auxiliar na obtenção de tais informações. Como soluções possíveis para este problema, pode-se citar alguns métodos para processamento e mineração de dados contidos na web, como por exemplo tratativas usando Processamento de Linguagem Natural e *Machine Learning*.

Como apresentado por Marante et al. (2020) que realiza um mapeamento da principais pesquisas que realizam alguma forma de recomendação de conteúdos educacionais, mostrando a importância de ferramentas e métodos que possam trazer ao usuário uma forma fácil e simples de obter um conteúdo relevante

Estas abordagens auxiliam na captura das informações e organização das mesmas, trazendo uma melhor experiencia de uso ao aluno e professor. Tomando como base estes recursos computacionais podemos pensar em uma ferramenta que possa, de forma automática agrupar esses dados em um só local, fazendo com que as informações requeridas sejam categorizadas e estruturadas a fim de atender as necessidades do usuário final, melhorando a experiência de uso no processo de recuperação da informação.

Além da criação de uma ferramenta pode se realizar uma analise das melhores formas de se categorizar e estruturar os dados obtidos da internet.

Soluções computacionais, como o software de busca e recomendação proposto nesta pesquisa e analise dos métodos de classificação são formas de agilizar e melhorar o acesso a materiais confiáveis e de maior qualidade por parte tanto de professores como de alunos, visando assim fornecer um suporte computacional ao processo de ensino e aprendizagem no contexto da web.

1.4 Objetivos

A partir da análise de textos e artigos que utilizam métodos para categorização e identificação de *tags* em *datasets* de diferentes contextos, o presente trabalho foi conduzido com o propósito de desenvolver uma solução computacional a fim de auxiliar e agilizar o processo de busca por conteúdos educacionais, proporcionando a alunos e professores

uma forma mais confiável e eficiente de se obter materiais educacionais.

Para que isso possa ser possível, esta pesquisa tem como **objetivo geral** desenvolver uma ferramenta (software) de busca capaz de identificar e agrupar, de forma automática, categorias de textos educacionais, utilizando técnicas de Processamento de Linguagem Natural, Aprendizagem de Máquina, a fim de auxiliar professores e alunos durante as etapas de busca por materiais educacionais.

Tem-se como objetivo específicos da pesquisa os seguintes:

- Desenvolver uma ferramenta de busca para centralizar as consultas por documentos educacionais, categorizando-os a partir de sub áreas existentes nas disciplinas, de forma a auxiliar docentes e estudantes durante as pesquisas realizadas.
- O desenvolvimento de uma arquitetura modular, onde os módulos pertencentes a ela sejam independentes entre si, fazendo com que cada módulo tenha um objetivo específico dentro da arquitetura.
- Entendimento dos métodos adequados para a tarefa de classificação de materiais educacionais, realizando testes comparativos entre os métodos estudados durante o desenvolvimento do trabalho.

Desta forma, presente trabalho esta organizado da seguinte forma: No capítulo 2 são abordados os principais conceitos necessários para o entendimento do trabalho. No capítulo 3 serão apresentado apresentados as atividades desenvolvidas juntamente com as etapas metodológicas criadas para o desenvolvimento. Já no capítulo 4 são apresentadas as atividades desenvolvidas para a criação da arquitetura e da ferramenta, juntamente com o fluxo de teste dos métodos de categorização. E por fim, o capítulo 5 são trazidas as conclusões, juntamente com trabalhos futuros

2 Fundamentação Teórica

Neste capítulo são abordados os principais conceitos necessários para um melhor entendimento do trabalho desenvolvido no decorrer desta monografia.

Os conceitos apresentados, a seguir, tem como objetivo fundamentar as técnicas e termos utilizados durante a criação e processamento de textos em linguagem natural, acrescentando, de forma significativa, uma base para o estudo que será apresentado durante o desenvolvimento deste trabalho.

Assim, o capítulo de fundamentação teórica visa trazer conceitos e técnicas que são usados para o desenvolvimento do trabalho. Este capítulo foi dividido nas seguintes subseções: 2.1 Processamento de Linguagem Natural, 2.2 Abordagens Estatísticas, 2.3 Web Crawler, 2.4 Machine Learning e 2.5 Trabalhos Relacionados.

2.1 Processamento de Linguagem Natural

O Processamento de Linguagem Natural constitui o tratamento e análise dos diversos métodos de comunicação humana, como por exemplo sons, palavras, sentenças e discursos, proporcionando ao computador se comunicar em linguagem humana, levando em conta vários níveis da linguagem, Morfológico, Fonético/Fonológico e Semântico (GONZALEZ; LIMA, 2003):

- **Morfológico:** Estrutura e formação de uma palavra, analisada de forma isolada das demais. A morfologia é o primeiro estágio da análise, uma vez que a entrada foi recebida. Ele analisa as maneiras pelas quais as palavras se dividem em seus componentes e como isso afeta seu status gramatical Reshamwala, Mishra e Pawar (2013). E como isso afeta o estado gramatical da palavra e como ela é formada (Substantivos, Verbos e Pronomes).

  Ex: Compraram - O morfema, **aram**, que é a menor unidade que apresenta significado, indicando o número (plural) e o tempo verbal;

- **Sintaxe:** Semelhante a análise morfológica, a sintaxe tem como propósito o estudo da palavra, porém olhando dentro do contexto de uma sentença, podendo assim rotular as palavras contidas na sentença. Como apresentado na Figura 2.1

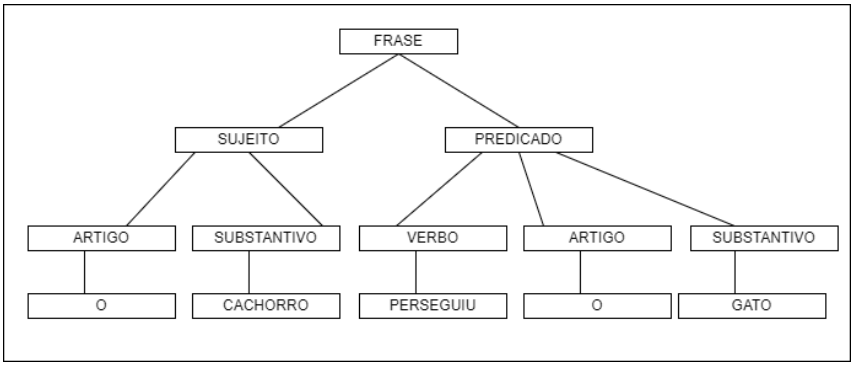


Figura 2.1: Exemplo análise de sintaxe

- **Fonético e Fonológico:** Forma do som produzido a partir da fala. Baseando-se em três tipos de regras na análise fonológica: Regras fonéticas - sons das palavras faladas, Regras Fonêmicas - variação da pronúncia quando as palavras são faladas juntas e Regras Prosódicas - acentuação e entonação ao longo da frase (LIDDY, 2001).

- **Semântico**: Relação de palavras e seus significados que combinadas trazem sentidos em uma sentença. Enquanto a sintaxe corresponde ao estudo de como as palavras agrupam-se para formar estruturas em nível de sentença, a semântica está relacionada ao significado, não só de cada palavra, mas também do conjunto resultante delas (GONZALEZ; LIMA, 2003).

Como apresentado por Liddy (2001), o Processamento de Linguagem Natural é uma gama de técnicas computacionais teoricamente motivadas para analisar e representar textos que ocorrem naturalmente em um ou mais níveis de análise linguística.

Dentro da área de Processamento de Linguagem Natural muitas técnicas para o processamento dos textos analisados são usados como a **etiquetagem** das palavras do texto. Um etiquetador gramatical, conhecido como *Part-of-speech tagger (POS-Tagger)*, tem como objetivo colocar etiquetas nas palavras analisadas, identificando a qual categoria gramatical as palavras etiquetadas pertencem.

Outra abordagem utilizada dentro de PLN é a **normalização** das palavras existentes no texto, dividindo-se em *Stemming* e *Lemmatization*.

Stemming consiste na redução das palavras, que possuem radicais em comum, a uma forma comum ao radical, sendo retirados os afixos oriundos da derivação e flexão deste radical em comum. Como por exemplo, palavras como *dirigiu* e *dirigiremos* seriam reduzidas para a forma comum, *dirigi* (GONZALEZ; LIMA, 2003).

Já *Lemmatization* é redução da palavra para a forma canônica, levando verbos para o infinitivo e os adjetivos e substantivos para a forma masculina e singular, como por exemplo as palavras *construção* e *construiremos*, ao se aplicar *Lemmatization* seria reduzidas para *construir* e *construção*

Outra técnica utilizada é a eliminação de *StopWords*, que são as palavras que não trazem tanta importância durante a analise de um corpus, como por exemplo: artigos, conectivos e preposições.

2.2 Abordagens Estatísticas

Como apresentado por Zerbinatti (2010), a utilização de abordagens estatísticas se propõe a realizar uma análise do corpus, identificando a frequências de aparições de palavras no texto, identificação de grupos de ocorrências de palavras e a probabilidade de aparição das palavras, criando assim modelos matemáticos caracterizando o texto analisado.

Uma das técnicas mais usadas para a identificação da frequência e relevância de termos em um texto é chamado TF-IDF(*term frequency–inverse document frequency*), que consiste em encontrar a relevância de um determinado termo em um conjunto de textos. O TF-IDF consiste no inverso da frequência de aparições de um determinado termo em um conjunto de textos, revelando assim a importância do termo no texto (RAMOS et al., 2003).

TF-IDF é a combinação de duas abordagens, *Term-frequency* e *Inverse Document Frequency*. Onde TF é usada para mensurar a frequência de um determinado termo dentro de um documento (QAISER; ALI, 2018) . Como exemplo, podemos supor um documento “D1”, que possui 10.000 palavras e a palavra “Guerra” aparece 8 vezes o documento “D1”, logo a frequência do termo “Guerra” em “D1” é calculada da seguinte

forma:

TF = 8/10.000 = 0,0008

Como visto, o cálculo de TF não consegue levar em conta a importância de um determinado termo dentro de um documento, tratando as palavras analisadas como o mesmo peso, não importando se o termo analisado é um *stopword* ou não. Tomando como exemplo, caso a palavra "de" aparecer em um documento 800 vezes e não possui muita relevância dentro do texto, com o IDF, o termo não terá um menor peso na análise do documento.

Considerando 10 documentos na qual o termo "Guerra" está presente em 5 desses documentos, então a frequência inversa do documento pode ser calculada como (QAISER; ALI, 2018):

IDF = $\log_e(10/5) = 0,3010$

Tendo em vista os conceitos de TF e de IDF, quanto maior ocorrência de uma palavra nos documentos maior será a frequência (TF) de termo e menor ocorrência de palavra nos documentos dará maior importância (IDF) para aquela palavra-chave pesquisada em determinado documento.

Assim, a definição de TD-IDF não é nada mais do que multiplicação da frequência do termo (TF) e da frequência do documento inverso (IDF), sendo calculado da seguinte forma (QAISER; ALI, 2018):

TF-IDF = 0,0008 * 0,3010 = 0.0002408

Outra técnica utilizada durante o processo de extração de instâncias é a busca pela similaridade existente entre os termos do corpus, auxiliando na validação das instâncias obtidas durante o processo de extração, tendo em vista que existe uma similaridade entre as palavras extraídas.

Segundo Feldman, Sanger et al. (2007) a similaridade cosseno é uma das medidas de similaridade mais populares na Mineração de Texto. Quando os documentos textuais

são representados por vetores de termos, a similaridade entre eles pode ser obtida pela correlação entre os seus respectivos vetores.Quando essa correlação é quantificada como o cosseno do ângulo entre dois vetores, chamamos esse valor de similaridade cosseno.

Quando a similaridade cosseno entre dois vetores (documentos) é próxima de 1, significa que os vetores formam um ângulo próximo de 0 graus, logo os documentos são semelhantes. Já quando a similaridade é próxima de 0 graus, significa que os vetores formam um ângulo próximo de 90 graus. Isso indica que os componentes dos vetores não são semelhantes e, portanto, os documentos também não são semelhantes (SOARES, 2017).

2.3 Web Crawler

Web crawlers, se caracteriza como um software ou *script* que tem como função navegar pelas estruturas de páginas web, de forma automatizada, a fim de recuperar as informações contidas nessas páginas (KAUSAR; DHAKA; SINGH, 2013).

As informações obtidas através do *crawler* são armazenadas de forma centralizada para que as mesmas possam ser processadas e indexadas para serem utilizadas de diversas formas em aplicações futuras (DHENAKARAN; SAMBANTHAN, 2011).

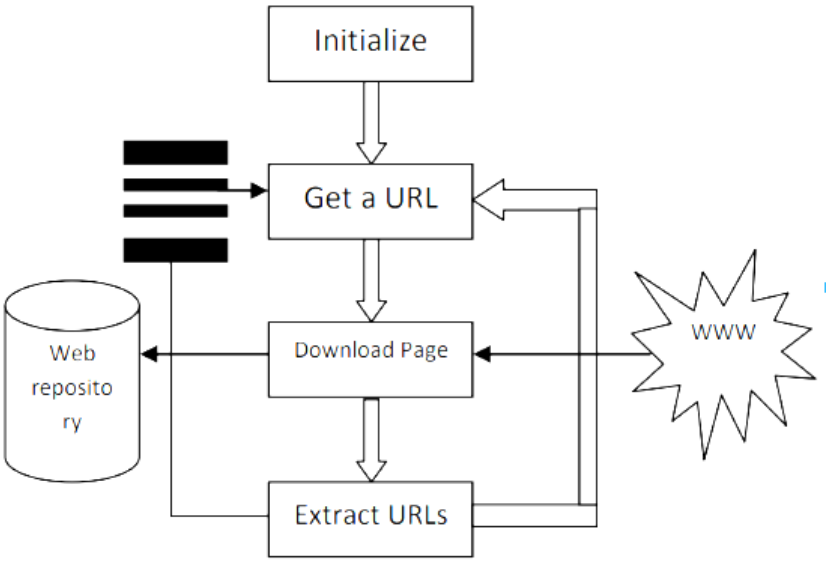
Na Figura 2.2, apresentada por Kausar, Dhaka e Singh (2013) é possível entender melhor o fluxo realizado durante a execução de um *Web Crawler*.

Em Kausar, Dhaka e Singh (2013) são apresentadas 3 (três) técnicas usadas para se desenvolver um *web crawler*, que são :

- ***General Purpose Crawling:*** Este tipo de *crawler* coleta as informações de um conjunto de URL's e os possíveis links pertencentes às páginas. Podendo desta forma gerar um grande conjunto de dados, pelo fato deste *crawler* recuperar todo o conteúdo das páginas.

- ***Focused Crawling:*** Esta técnica tem como característica recuperar apenas documentos de um tópico específico, reduzindo o número de informações coletadas. Desta forma é feita uma busca por conjunto de dados específicos e direcionados a aplicação a qual o *crawler* pertence.

Figura 2.2: Fluxo de execução de um Web Crawler



Fonte: KAUSAR; DHAKA; SINGH (2013)

- ***Distributed Crawling:*** Realiza um múltiplo processamento e obtenção do conteúdo de várias páginas *web*

2.4 Machine Learning

A necessidade de se classificar é uma tarefa existente em quase todas as áreas das ciências, automatizar essa tarefa é imprescindível para impulsionar o progresso científico Rudin e Wagstaff (2014). Para tanto *Machine Learning* (ML) é uma técnica comumente usada.

Machine Learning é uma subárea da Inteligência Artificial que tem apresentado um crescimento enorme nas últimas décadas, tratando-se de algoritmos matemáticos, estatísticos e computacionais que são capazes de realizar tarefas de inferências a partir de exemplos, sendo um meio ideal para a realização de automações de processos.

Em *Machine Learning* , os algoritmos são desenvolvidos para descobrir relações complexas desconhecidas entre variáveis de um grande conjunto de dados, e esses algoritmos produzem saídas com base no que descobriram ou "aprenderam" do conjunto de dados.

Existem muitos algoritmos de *Machine Learning* que diferem uns dos outros no método que usam para descobrir padrões em um conjunto de dados. A seleção de algoritmos de ML depende de vários fatores, como tamanho do conjunto de dados, número e tipo

de variáveis de entrada, tipo de resultado e poder computacional necessário (BALOGLU; LATIFI; NAZHA, 2021).

Como apresentado por Naqa e Murphy (2015), *Machine Learning* é um ramo da Inteligência Artificial em que os algoritmos computacionais são projetados para emular a inteligência humana aprendendo com o ambiente ao redor.

Técnicas baseadas em aprendizado de máquina têm sido aplicadas com sucesso em diversos campos, desde reconhecimento de padrões, visão computacional, engenharia de naves espaciais, finanças, entretenimento e biologia computacional até aplicações biomédicas e médicas.

Machine Learning traz uma contribuição muito grande na área de Processamento de Linguagem Natural, onde existe uma interdisciplinaridade entre a inteligência artificial e o processamento linguístico (EDGCOMB; ZIMA, 2019).

As técnicas de *Machine Learning* podem se dividir em diversos métodos de utilização, onde são segmentadas em: algoritmos de aprendizado supervisionados, não supervisionados e aprendizado semi supervisionados.

Em **algoritmos supervisionados** o algoritmo é treinado a partir de dados pré-definidos e/ou rotulados previamente. Desta forma, o programa é capaz de tomar sua própria decisão de novos conjuntos de dados processados (MAHESH, 2020).

Já em **algoritmos não supervisionados** de aprendizagem de máquina, como também apresentado por Mahesh (2020), diferente de algoritmos supervisionados, não utiliza um conjunto de dados categorizados de forma prévia como forma de treinamento. Ele é capaz de encontrar automaticamente padrões a partir de um conjunto de dados, cabendo ao algoritmo encontrar semelhanças entre os grupos de dados e conseguir agrupá-los nas categorias mais coerentes para aqueles dados.

Algumas técnicas de aprendizado não supervisionado são apresentadas em Mahesh (2020), como o algoritmos de *k-Means* e *Principal Component Analysis*.

O algoritmo de *k-Means* é um algoritmo de clusterização que avalia e clusteriza os dados de acordo com suas características importantes dentro de uma aplicação, classificando os dados a partir de um número de *clusters* definidos (CAMBRONERO; MORENO, 2006).

Principal Component Analysis (PCA) é uma técnica multivariada que analisa uma tabela de dados na qual as observações são descritas por diversas variáveis quantitativas dependentes inter correlacionadas. Seu objetivo é extrair as informações importantes da tabela, representá-la como um conjunto de novas variáveis ortogonais chamadas de componentes principais e exibir o padrão de similaridade das observações e das variáveis como pontos em mapas (ABDI; WILLIAMS, 2010).

Os algoritmos de **aprendizagem semi supervisionados** utilizam uma combinação de estratégias dos métodos supervisionados e não supervisionados. É utilizado de forma semelhante aos métodos de aprendizado supervisionado, porém os dados são diferentes, geralmente, é formado por uma pequena quantidade de dados rotulados, que normalmente têm custo mais elevado, e por um volume maior de dados não rotulados. Mahesh (2020) apresenta alguns métodos de aprendizagem semi supervisionados.

Como foco principal para este trabalho foram utilizados técnicas e algoritmos de **aprendizagem supervisionada**, desta forma, na seção 2.4.1 são detalhados os algoritmos utilizados para o desenvolvimento do trabalho

2.4.1 Algoritmos de aprendizado supervisionado

Alguns dos algoritmos de aprendizagem de máquina supervisionados mais conhecidos são:

- **Decision Tree:** a árvore de decisão é uma estrutura de decisão modelada na forma de um grafo de escolha, onde os resultados são apresentados na forma de uma árvore. Desta forma, os nós folhas representam uma escolha/evento e as arestas marcam o caminho usado para que as decisões/condições pudessem ser tomadasMahesh (2020), como no exemplo apresentado na Figura 2.3

- **Naive Bayes:** o classificador de Naive Bayes é baseado no Teorema de de Bayes, que consiste em uma fórmula matemática usada para o cálculo da probabilidade de um acontecimento tendo em vista a ocorrência de um outro evento já observado. Assim, o teorema de Bayes precisa ter como referência o resultado de eventos anteriores para calcular a probabilidade dos futuros eventos, condicionados aos eventos anteriores.

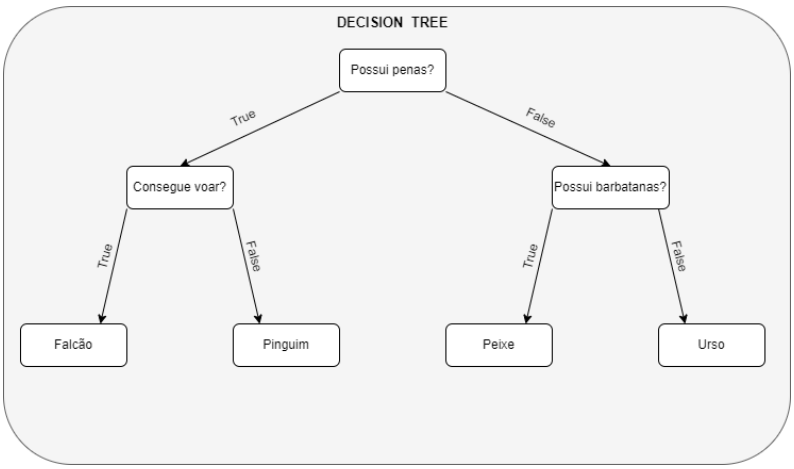


Figura 2.3: Decision Tree criada pelo aluno como exemplo

Sendo a probabilidade calculada a partir da equação 2.1

$$P(A|B) = \frac{P(B|A) * P(A)}{(P(B)} \tag{2.1}$$

- **Support Vector Machine:** esta estratégia de classificação usa um conjunto de técnicas de aprendizagem de máquina supervisionada que analisam a massa de dados, reconhecendo padrões. Tomando como entrada um conjunto de dados classificados e prediz, a partir da entrada, a qual classe o novo conjunto de dados pertence.

  O SVM é então uma representação, um conjunto de pontos no espaço, na qual esses pontos são mapeados visando dispô-los em seus possíveis conjuntos.

  Assim, uma SVM busca encontrar uma linha de separação, entre dados de duas classes. Essa linha busca maximizar a distância entre os pontos mais próximos em relação a cada uma das classes, como na Figura 2.4.

2.5 Trabalhos Relacionados

Para a busca de trabalhos relacionados foi criada uma frase de busca que foi utilizada em bases de dados como **Scopus**, **IEEE** e **Google Scholar**. Esta *String* consiste na junção de termos chaves associados ao tema proposto neste trabalho de pesquisa.

Desta forma, a busca de artigos relacionados usou a seguinte palavra de busca:

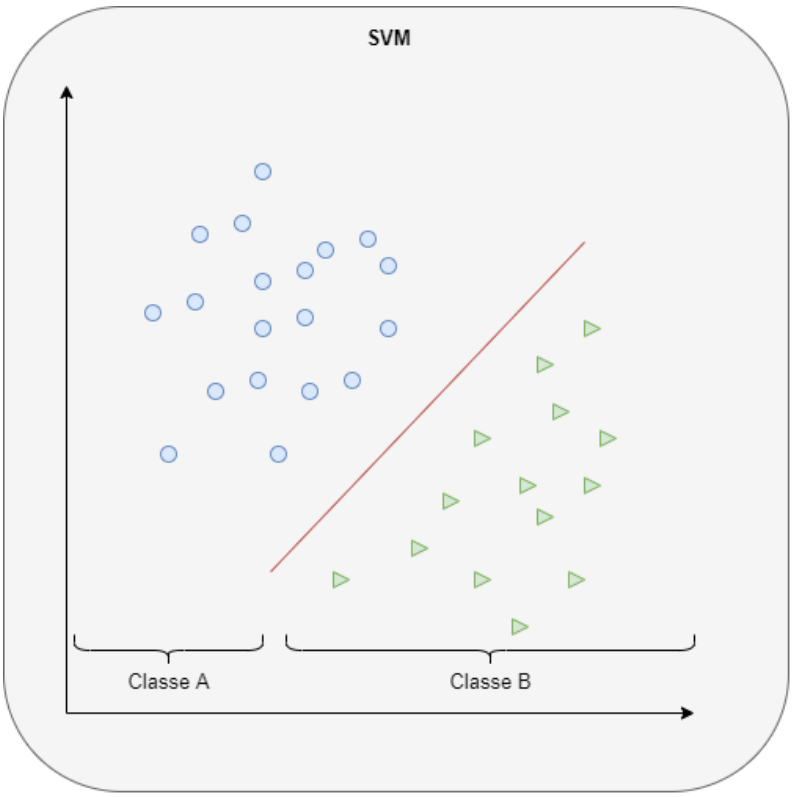


Figura 2.4: Gráfico SVM criado pelo aluno como exemplo

((*"categorization"* OR *"classification"*) AND (*"educational documents"* OR *"educational texts"* OR *"education text"* OR *"pedagogical documents"*) AND (*"machine learning"*))

Os artigos apresentados serviram de base para o correto desenvolvimento deste trabalho. A seguir são apresentados alguns trabalhos que mais se assemelham ou trazem algum direcionamento para o desenvolvimento dos classificadores para os textos educacionais. Os dados relevantes sobre os artigos podem ser visto na Tabela 2.1

Muitos trabalhos encontrados durante esta fase de pesquisa buscam realizar alguma forma de classificação e/ou recomendação de materiais educacionais, visando melhorar a forma como esses dados são utilizados dentro de um contexto específico da área da educação.

Isso pode ser visto no trabalho apresentado por Júnior, Araújo e Dorça (2020) que tem como objetivo a criação de um sistema de recomendação de matérias educacionais complementares em um contexto de sala de aula. Desta forma ele propõe o desenvolvimento de uma forma de recomendação personalizada de materiais adicionais a partir do

Tabela 2.1: Tabela Trabalhos Relacionados

Artigo	Domínio	Métodos	Dataset usado
Gupta, Agarwal e Jain (2019)	Gênero de Livros	*Principle Component Analysis*	Dados vindos da enciclopédia *Encyclopedia of Needlework*
Patil e Pawar (2012)	Homepage de páginas web	*Naive Bayes*	Dados coletados de paginas web governamentais
Júnior, Araújo e Dorça (2020)	Recomendação de matérias educacionais complementares	*Algoritmo Genético*	Informações vindas dos materiais ministrados em sala de aula e vindos da *Wikipédia* e *Youtube*
Apuk e Nuçi (2021)	Categorização de vídeos de plataforma de cursos online	*K-Nearest Neighbour* e *Long Short-Term Memory*	Os dados usados são coletados da plataforma *Coursera*
Woogue, Pineda e Maderazo (2017)	Categorização de paginas web educacionais	*Logistic Regression*, *Linear SVM*,*Multinomial Naive Bayes*, *k-Nearest Neighbor*, *Decision Trees*, *Random Forest* e *Multilayer Perceptron*	As informações utilizadas vem de sites web educacionais, com a Wikipedia *Coursera*

conteúdo de uma aula e as preferências do estudante.

O sistema de recomendação proposto por Júnior, Araújo e Dorça (2020) utiliza materiais de repositórios locais, através do qual o professor que ministra a aula consegue cadastrar todo o conteúdo abordado durante a aula e utiliza também paginas web (*Wikipédia* e *Youtube*) para recomendar os materiais que mais se adéquam à necessidade do aluno.

Para o desenvolvimento deste sistema o autor propõe uma arquitetura que, inicialmente espera que o docente cadastre todos os dados referentes a aula ministrada por ele, além da utilização desses dados cadastrados pelo professor, conteúdos vindos da *Wikipédia* e *Youtube* também são usados como material complementar a ser recomendado para o aluno.

Em seguida, o perfil dos alunos pertencentes a turma é gerado. Esse perfil é gerado a partir de um conjunto de dados inseridos pelo aluno através de um questionário,

mapeando assim as preferências dele.

E como forma de recomendação o autor utiliza Algoritmo Genético para poder entender as necessidades do aluno e assim conseguir recomendar o material que melhor se encaixa com as suas preferências.

Já em Apuk e Nuçi (2021) é feito um estudo sobre dois métodos de categorização diferentes, comparando um método convencional de *Machine Learning* com um método de *Deep Learning*, visando a categorização de conteúdos pedagógicos. Os algoritmos usados durante a pesquisa são *K-Nearest Neighbour* e *Long Short-Term Memory*

Os autores visam principalmente realizar a classificação de conjuntos de dados pertencentes a plataformas onlines que possuem um grande número de cursos, como por exemplo *Udemy*, *Coursera*, *Khan Academy*, dentre outras, que em um contexto geral, apresentam seus conteúdos através de video aulas.

Como forma de testar os métodos escolhidos, Apuk e Nuçi (2021) utilizaram um conjunto de 12.032 vídeos coletados da plataforma de cursos *Coursera*, que possui mais de 200 cursos diferentes. Utilizando esses dados, os autores processa os vídeos a fim de transcrever o áudio dos vídeos, obtendo assim o texto escrito. Os textos obtidos passam por uma etapa de normalização dos dados, na qual todas as palavras em letras maiúsculas são transformadas em letras minúsculas, as *stopwords* são retiradas e as palavras são *lemmatizadas*, passando-as para o seu radical. Após a etapa de normalização, a base de dados está pronta para ser usada nos algoritmos selecionados que classificam os documentos em três categorias diferentes, sendo eles, um contexto mais geral, um contexto específico e por último um contexto voltado para um curso específico.

Após os testes os autores conseguiram chegar a um resultado no qual o método *K-Nearest Neighbours* se saiu melhor se comparado com *Long Short-Term Memory*, entendendo que o número de categorias e a quantidade de documentos usados durante os teste acabam afetando diretamente na acurácia obtida pelos métodos.

Em Gupta, Agarwal e Jain (2019) os autores buscam realizar a classificação de gênero de livros utilizando técnicas de processamento de linguagem natural.

O dataset utilizado para o desenvolvimento da pesquisa foi coletado do website da enciclopédia *Encyclopedia of Needlework*, totalizando em aproximadamente 3600 livros a

serem classificados. Tendo em vista que muitos destes textos possuem palavras e sentenças semelhantes, os autores propuseram utilizar *bag of sense*, com a ajuda da *WordNet* para encontrar termos semelhantes para essas palavras, reduzindo assim as dimensões da matriz de vetorização a ser usada pelo classificador.

Em uma etapa de pré-processamento, os livros com mais de um gênero são removidos do conjunto de dados e são gerados os *tokens* a partir das palavras e sentenças. Usando os *tokens* gerados é calculado o TF-IDF para que a matriz seja montada e assim usada como os pesos para cada *token* pertencente ao dataset.

Considerando que, pela existência de muitos *tokens*, a matriz de pesos criada se torna muito esparsa, os autores propuseram o uso do *Principle Component Analysis (PCA)* para poder reduzir a dimensionalidade da matriz. Assim, a matriz criada é dividida em duas partes, um conjunto para treinamento, utilizando 80% do dataset, e outro para os testes, usando os 20% restantes, para isso foi usada *Decision Tree Classifier*.

Na fase de testes e avaliação dos resultados foi alcançada uma acurácia de aproximadamente 81.18% utilizando um conjunto de dados para testes rotulados e uma acurácia de 92.88% usando um grupo de dados não rotulados na fase de pré-processamento.

Em Woogue, Pineda e Maderazo (2017) realiza-se um estudo sobre diferentes métodos de *Machine Learning* a fim de classificar as páginas web pertencentes ao domínio educacional. A classificação das páginas é feita utilizando um conjunto de classes previamente selecionadas. Esse conjunto é composto por 9 (nove) categorias diferente, sendo elas: Arquitetura e Belas Artes & Design, Matemática e Ciências, Artes (Psicologia, Literatura, História), Medicina, Política (Direito e Governança), Negócios e Economia, Nutrição / Dieta / Saúde, Engenharia de Software / Programação / Tecnologia e Outros. Desta forma, os classificadores utilizados durante o estudo associam as páginas as suas categorias especificas.

Os dados utilizados durante o desenvolvimento dos testes foram retirados, de forma manual, de páginas web que possuem informações que pertencem as 9 categorias criadas, gerando assim um conjunto com aproximadamente 150 páginas para cada categoria usada.

Esse conjunto de dados foi normalizado utilizando técnicas de Processamento

de Linguagem Natural, retirando todas as *stopwords* existentes nos textos e também é realizada uma etapa de *lemmatização* a fim de reduzir palavras semelhantes para o mesmo radical.

Em seguida esses dados são utilizados como fonte de treinamento para os métodos de *Machine Learning* que foram utilizados e comparados durante este trabalho. Para os testes foram utilizados 7 métodos de classificação diferentes, que são: *Logistic Regression*, *Linear SVM*, *Multinomial Naive Bayes*, *k-Nearest Neighbor*, *Decision Trees*, *Random Forest* e *Multilayer Perceptron*.

Utilizando estes algoritmos durante os testes os autores chegaram ao resultado que *Logistic Regression*, *Linear SVM* conseguindo alcançar os melhores valores de acurácia dentre os métodos utilizados durante os testes

Em Patil e Pawar (2012) realizou-se uma pesquisa, fora do contexto educacional, sobre o uso do algoritmo de *Naive Bayes* para a classificação de forma automática de páginas web. Tendo em vista a estrutura utilizada em páginas web, inicialmente os autores realizam a retirada de todas as tags HTML utilizadas nas páginas coletadas para o estudo realizado. Em seguida, todas as *stopwords* são retiradas, a fim de limpar ainda mais os documentos que serão classificados pelos algoritmos de*Machine Learning* usados para a classificação dos dados das páginas.

Como base de dados os autores utilizam *homepages* de sites e os dados contidos nestas páginas são pré-classificados em cerca de 10 categorias, na qual cada página possui sua categoria específica, podendo variar entre páginas relacionadas a cuidados com saúde, esportes, computação, etc.

Como forma de teste, os autores quebram seu conjunto de documentos, utilizando uma parte para se testar e uma outra parte para treinar o algoritmo de *Naive Bayes* a ser utilizado para o teste. patil2012automated utiliza um conjunto de informações contendo um total de 4887 documentos.

Durante os testes o número de documentos usados para treino é variado a cada teste executado, gerando assim, um número pequeno de dados usados como fonte de treino, uma acurácia muito baixa, ou seja, quando utilizado apenas 50 documentos, acurácia chega perto dos 45%, já quando o número de documentos para treino aumenta para cerca

de 450 a acurácia também aumenta, para 89%.

3 Metodologia

No presente capítulo serão apresentados as atividades desenvolvidas juntamente com as etapas metodológicas criadas para o desenvolvimento.

Inicialmente foi feito um levantamento dos principais estudos voltados para classificação de documentos existentes, criando assim uma base para o desenvolvimento das etapas seguintes do trabalho. Em seguida foi idealizada uma arquitetura composta por módulos responsáveis por partes distintas da plataforma desenvolvida.

Desta forma o capítulo de metodologia visa esclarecer as etapas de desenvolvimento da pesquisa desenvolvida. Este capítulo de TCC foi dividido nas seguintes subseções: 3.1 - Classificação da Pesquisa, 3.2 - Tecnologias Utilizadas, 3.3 - Arquitetura Proposta, 3.4 - Captura dos conteúdos web e, por fim, 3.5 - Categorização do conteúdo.

3.1 Classificação da Pesquisa

A metodologia usada para a realização do projeto se dá desde a etapa de busca de trabalhos relacionados, analisando as principais técnicas de categorização, até o desenvolvimento de uma nova plataforma que possa categorizar e compartilhar os documentos educacionais de forma automática. Considerando tal metodologia de pesquisa, pode-se classificar este trabalho conforme os parágrafos a seguir.

Quanto aos objetivos de pesquisa apresentados por Pereira et al. (2018) a pesquisa desenvolvida se caracteriza como **exploratória** pelo fato de buscar maiores informações sobre trabalhos que visam a coleta e categorização de documentos.

Pode ser também classificada como **pesquisa bibliográfica**, já que o presente trabalho visa absorver conceitos e técnicas existentes em artigos relacionados como base fundamental para a pesquisa, sem perder o viés científico do trabalho.

Quanto à abordagem do problema esta pesquisa pode ser classificada a partir do que é apresentado em (CRESWELL; CRESWELL, 2003; PEREIRA et al., 2018). Desta forma tal pesquisa se caracteriza como **quantitativa e qualitativa**, já que realiza

análises e testes realizados do projeto de forma numérica através de gráficos, comparando a acurácia dos métodos de classificação utilizados no presente trabalho, além de um comparativo de experiência dos usuários ao acessar a plataforma desenvolvida e compará-la com as ferramentas tradicionais de busca, atualmente.

Este trabalho pode ser também caracterizado como uma **pesquisa aplicada**, por desenvolver um algoritmo aplicando técnicas de categorização vista nos artigos de referência

3.2 Tecnologias utilizadas

Para o desenvolvimento dos módulos implementados na ferramenta de busca proposta foram utilizadas as linguagens de programação **Java**, **Python** e **JavaScript**, além de frameworks (Angular e Django) e bibliotecas (Spring) que proporcionam a criação de cada módulo. Isso se dá pelo fato dos módulos se comunicarem através de serviços **REST**, proporcionando a criação de módulos com tecnologias diversas.

Para o desenvolvimento da interface (front-end) que usuários finais podem acessar, optou-se por um design simplista e auto-explicativo, utilizando para tal como tecnologias o framework Angular através da linguagem JavaScript. Para a customização da interface foram utilizados recursos das linguagens de marcação HTML e CSS, além do framework Bootstrap.

3.3 Arquitetura Proposta

Outro diferencial é que artefato produzido por esta pesquisa refere-se à arquitetura proposta para o software de busca desenvolvido. Tal arquitetura pode ser utilizada por outras pesquisas que optarem por conduzir um trabalho na mesma área aqui explorada.

A arquitetura desenvolvida e apresentada tem como principal finalidade a criação de um recurso que possa agregar e disponibilizar informações de diversas áreas do conhecimento, tendo em vista que por ser uma arquitetura modular o seu uso pode ser associado a diversas áreas educacionais. Podendo ser criados novos módulos que possam ser conectados a arquitetura inicial, criando assim uma ferramenta mais robusta e com uma gama

maior de informações.

A arquitetura proposta é formada por 6 (seis) módulos, cada um deles responsável por uma tarefa específica da ferramenta. A arquitetura e seus componentes podem ser vistos como apresentados na Figura 3.1, a seguir.

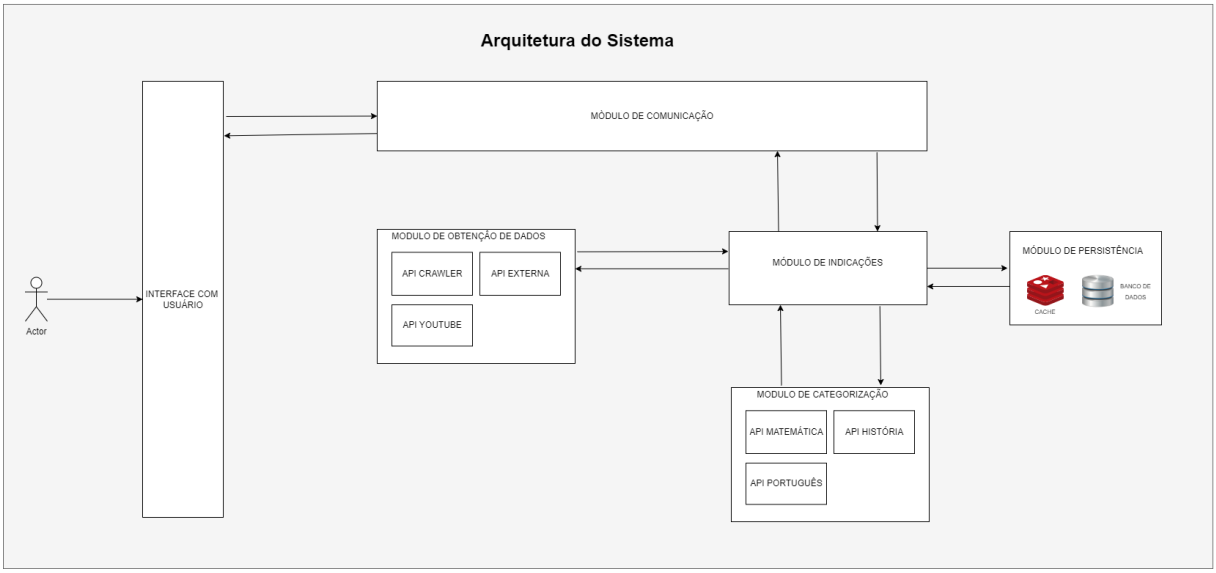


Figura 3.1: Arquitetura proposta pelo aluno

Interface com Usuário:este módulo é responsável pela interação direta da ferramenta com o usuário. É a parte visual do software proposto, onde o professor ou aluno poderá realizar suas buscas e visualizar os resultados das buscas feitas. Ele foi desenvolvido utilizando o framework Angular que utiliza a linguagem JavaScript juntamente com HTML e CSS.

A interface proposta possui duas telas, sendo a primeira responsável pela seleção de qual categoria da disciplina se deseja realizar a consulta e a segunda responsável por apresentar ao usuário, através de uma tabela, às páginas web retornadas a partir da categoria selecionada na primeira tela, como apresentado nas Figuras 3.2, 3.3 e 3.4.

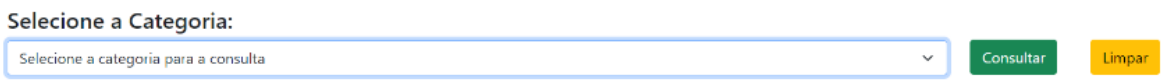


Figura 3.2: *Select box* para escolha de categoria

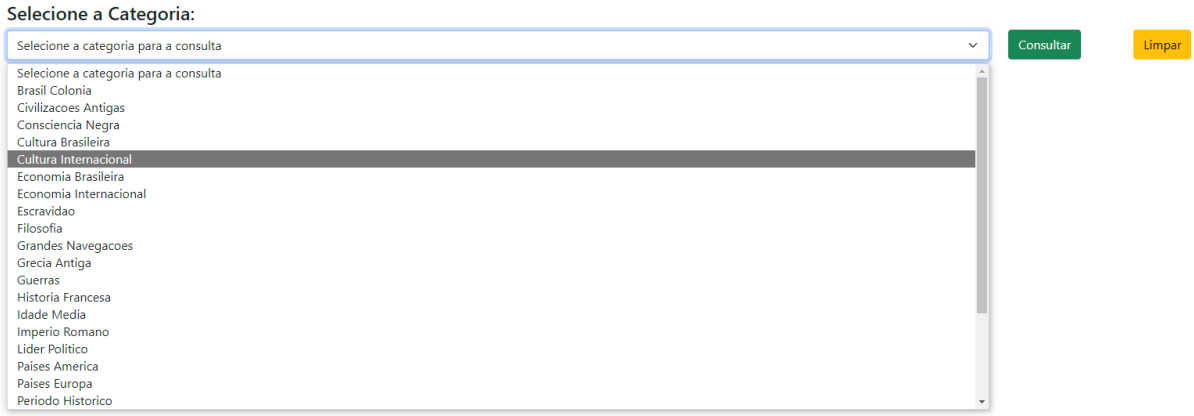


Figura 3.3: *Select box* com as opções de categoria

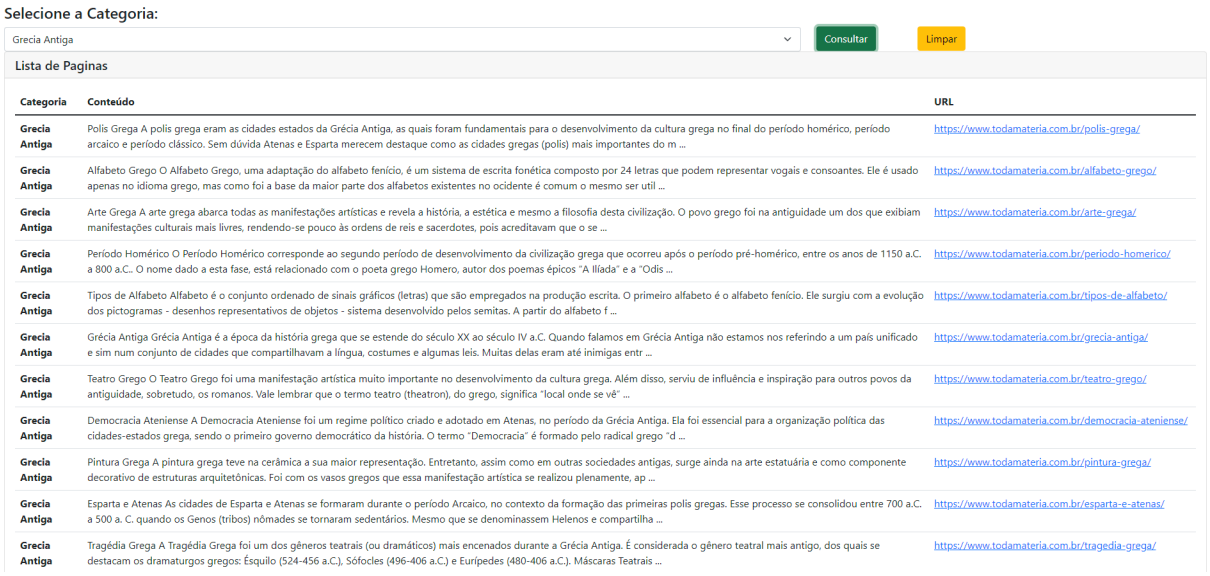


Figura 3.4: Lista com as páginas retornadas após a classificação

Módulo de Comunicação: foi desenvolvido para gerenciar a comunicação entre a interface com o usuário e os demais módulos, utilizando a Linguagem *Java* e o framework *Spring* como um facilitador para o desenvolvimento da API.

O *framework Spring*[1] possui uma série de ferramentas que auxiliam na criação de um projeto, neste caso, auxiliando na criação do módulo de comunicação, proporcionando uma forma fácil de criar uma estrutura capaz de realizar a comunicação entre a interface do usuário e os demais módulos deste projeto. Esta comunicação é feita a partir de *endpoints* REST disponibilizados pelo serviço, que são capazes de receber tanto as requisições solicitando a busca pela categoria desejada pelo usuário quanto a requisição responsável pela resposta após a realização da consulta pela categoria.

[1]https://spring.io/

Módulo de obtenção de dados: neste módulo é feita a obtenção das informações que serão classificadas. Para que isso aconteça, o módulo recupera os dados contidos em páginas web e após a obtenção dessas informações, elas passam por uma etapa de normalização dos textos, utilizando técnicas de PLN, para que o processo de categorização possa ser realizado.

Neste módulo de obtenção de dados foi utilizado o *framework Selenium*[2], que é comumente usado em testes automatizados e funcionais, que visam a interação com a interface. Por isso o Selenium foi usado para a criação de web crawlers responsáveis por recuperar informações contidas em sites da web de forma automática e mais simples.

Nesta fase do projeto foram criados 2 (dois) *crawlers* capazes de recuperar as informações relacionadas à disciplina de História. Informações estas que são usadas para o treinamento e como base de conteúdos a serem indicados para os alunos e professores. As informações utilizadas pertencem as páginas da web, Só História[3] e Toda Matéria[4].

Após a obtenção do conteúdo dos sites através dos crawlers, os dados são processados pelo módulo de indicações, que pré-processa o conteúdo das páginas, realizando alguns processos de PLN, como a retirada de *stopwords* para que somente termos possivelmente relevantes continuem nos documentos analisados.

Este módulo de indicações também foi desenvolvido utilizando a linguagem Java e a biblioteca *Apache OpenNLP*[5] responsável pela realização de todas as etapas de Processamento de Linguagem Natural.

Módulo de Categorização: a etapa executada no módulo de categorização é responsável por realizar a classificação dos recursos obtidos na etapa de obtenção de dados. Inicialmente este módulo irá passar por uma etapa de treinamento se baseando nas *tags* previamente selecionadas, baseando-se em classes contidas na Wikipédia. Desta forma, algoritmos de aprendizagem de máquina são executados a fim de obter a melhor classificação das informações obtidas.

O fluxo existente neste módulo foi criado usando a linguagem *Python* através da

[2]https://www.selenium.dev/
[3]https://www.sohistoria.com.br/
[4]https://www.todamateria.com.br/
[5]https://opennlp.apache.org/

biblioteca *Scikit-Learn*[6], que possui um conjunto de ferramentas de Machine Learning de código aberto, possuindo diversas técnicas e algoritmos que podem ser usados durante a tentativa de classificação de dados.

Para este módulo foram desenvolvidos 3 (três) métodos de classificação que utilizam técnicas de classificação distintas, a fim de gerar uma forma comparativa entre os métodos. Os classificadores foram criados usando os algoritmos de *Support Vector Machine*, *Decision Tree* e *Naive Bayes*.

Ambos os classificadores criados utilizam como fonte de treinamento um *dataset* já categorizado que foi retirado dos sites Só História e Toda Matéria. Esse *dataset* foi previamente categorizado para que os algoritmos de classificação pudessem utilizá-lo para a etapa de treinamento.

Para que o treinamento pudesse ser realizado o dataset passou por uma etapa onde o peso de cada palavra foi calculado utilizando TF-IDF, proporcionando aos algoritmos uma melhor forma de interpretar a relevância de cada palavra no contexto de sua categoria.

Módulo de Indicações: neste módulo também foi usado o Spring como framework para agilizar o desenvolvimento. Ele realiza a comunicação entre os outros fluxos de extração de dados e categorização. Ele estrutura as informações retornadas pelo módulo de obtenção de dados e encaminha para que a classificação do conteúdo possa ser feita. Desta forma, estes recursos poderão ser retornados para o usuário de forma organizada.

O módulo realiza requisições para o fluxo de categorização, passando todos os dados obtidos através dos *crawlers* e pré-processados através deste módulo de indicações.

Módulo de Persistência: este módulo do sistema fica responsável por receber e persistir os conteúdos já categorizados. Desta forma, o fluxo de busca por um conteúdo específico fica mais ágil. Para que isso pudesse ser realizado, foi criada uma tabela no banco de dados que armazena as informações das páginas e a quais categorias elas pertencem. O *Redis*[7] foi usado como tecnologia de cache para poder agilizar a recuperação dessas informações.

Assim, quando existe a necessidade de realizar uma nova busca por uma categoria

[6]https://scikit-learn.org/stable/
[7]https://redis.io/

específica existem 2 (dois) fluxos possíveis a serem realizados por este módulo: inicialmente é validado na base de dados se a busca por esta categoria já foi realizada em algum momento, caso a busca não tenha sido realizado, todo o fluxo de obtenção de dados e categorização é realizado, caso contrário - busca já tenha sido feita -, o módulo busca essas informações na base de dados.

Assim o fluxo de busca acaba se tornando mais rápido e com esse mesmo propósito o Redis é utilizado, armazenando em cache as informações das consultas mais recentes.

Mais detalhes da arquitetura proposta serão explicados nas seções abaixo.

3.4 Aquisição dos conteúdos web

Inicialmente este trabalho teve como objetivo realizar a obtenção de dados da disciplina de Matemática, porém durante os estudos voltados para as informações contidas nas páginas, foi possível notar que os textos dessa disciplina possuem muitos termos que acabam se repetindo dentro das categorias existente dessa área de estudo, dificultando os algoritmos de classificação conseguirem categorizar os dados obtidos em categorias.

Como exemplo deste problema podemos citar a duas categorias *"fração"* e *"equação"*, que durante a análise dos dados foi possível identificar que muitos termos eram comuns a essas duas categorias, dificultando a classificação e treinamento dos algoritmos.

A partir desta conclusão,foi escolhida a área de História pelo fato da mesma possuir textos mais extensos e com mais conteúdos para serem recuperados através do *crawler*. Além de possuir uma gama enorme de categorias a serem atribuídas aos dados coletados nas páginas.

Para o desenvolvimento do projeto foram utilizadas duas plataformas web voltadas para a disciplina de História, As plataformas usadas foram o site Só História e Toda Matéria. Ambas as páginas possuem um material que aborda a maioria das informações lecionadas na disciplina de História, descritas de uma forma mais clara e simples.

Com esse conceito em vista, foi desenvolvido um fluxo utilizando a linguagem Java responsável por percorrer as páginas web e extrair as informações necessárias para a busca das categorias.

Como apresentado na figura 3.5, foi criado um fluxo que percorre a estrutura de

HTML da página web buscando, de forma automática, os conteúdos a serem usados no fluxo de classificação em categorias.

Para a navegação da página foi utilizado o framework *Selenium* que é comumente usado para a realização de teste automatizados de aplicações webs. Através desse framework é possível capturar e percorrer as estruturas das aplicações, recuperando os dados importantes.

```
private void crowlPage(String url, int depth, List<PageComponent> components ) {
    try {
        if(url.contains(".zip")) {      //Caso a url a ser acessa possuir a extensão .zip, retorna
            return;
        }
        Document document = Jsoup.connect(url).userAgent(USER_AGENT).get();

        if(depth == 1) {                //Valida se o atual elemento é uma página com conteúdo, através da profundidade acessada

            Element element = document.getElementById("content");

            PageComponent pageComponent = new PageComponent();
            pageComponent.setUrl(url);
            pageComponent.setContent(getPageContent(element));
            if(!components.contains(pageComponent)) {
                components.add(pageComponent);  //Adiciona o conteudo da pagina atual na list com os demais conteudos
            }

        }else {
            Elements linksOnPage = getUrls(document);
            for (Element page : linksOnPage) {
                crowlPage(page.attr("abs:href"), 1, components);    //Chama a função de forma recursiva passando a url da pagina que possui o conteudo
            }
        }

    } catch (HttpStatusException e) {
        if(e.getStatusCode() == HttpStatus.NOT_FOUND.value()) {
            log.warn("Falied to access page {}", url);
        }
    } catch (Exception e) {
    throw new BusinessException(MessageFormat.format("ERRO: {0}, URL: {1}", e.getMessage(), url));
}

}
```

Figura 3.5: Fluxo de captura de informações para o site Só História

3.5 Categorização do conteúdo

Para a categorização dos conteúdos extraídos pelo fluxo de extração foram criados três tipos de categorizadores, utilizando algoritmos de aprendizagem de máquina supervisionado, os quais são: *SVM*, *Naive Bayes* e *Decision Tree*. Na figura 3.6 é possível visualizar os métodos de classificação criados.

O desenvolvimento do fluxo que utiliza esses algoritmos foi desenvolvido em Python, utilizando a biblioteca Sklearn, que possui um os principais algoritmos e métodos utilizados em *Machine Learning*.

Para o acesso a este módulo de classificação foi criado um endpoint REST utilizando o framework *Django*[8] que possibilita a criação de uma aplicação web através de

[8]https://www.djangoproject.com/

```
def classify_content_history_svm(self, train_content, targets, content_categorization):

    print("--------------->   SVM    <-----------------------")

    clf = SVC()

    clf.fit(train_content, targets)

    return clf.predict(content_categorization)


def classifier_contente_naive_bayes(self, train_content, targets, content_categorization):
    print("--------------->    Naive Bayes <-----------------------")

    clf = GaussianNB()
    clf.fit(train_content.toarray(), targets)
    return clf.predict(content_categorization.toarray())


def classifier_contente_decision_tree(self, train_content, targets, content_categorization):
    print("--------------->    Decision Tree   <-----------------------")

    clf = tree.DecisionTreeClassifier()
    clf.fit(train_content, targets)

    return clf.predict(content_categorization)
```

Figura 3.6: Métodos de *Machine Learning* utilizados

requisições HTTP realizadas para URLs criadas na aplicação. A URL criada para essa requisição pode ser observada na figura 3.7.

```
from django.urls.conf import path
from app import views
from django.contrib import admin


urlpatterns = [
    path('categorizations-history/', views.categorization_content_history)
]
```

Figura 3.7: Endpoint criado para acessar o módulo de classificação

Para o treinamento dos algoritmos foram utilizados um conjunto de dados obtidos a partir das páginas web usadas pelo *crawler* no módulo de obtenção de dados, essas informações foram previamente classificados, de forma manual, a partir da identificação e análise das classes que cada documento pertencia, como por exemplo *Império Romana*, *Brasil Colônia*, *Grandes Navegações*, dentre outras classes. Esses documentos, já rotulados, são utilizados pelos algoritmos tendo em vista que ambos os métodos de classificação são técnicas de aprendizado supervisionado.

Inicialmente os algoritmos foram treinados utilizando um conjunto de dados contendo cerca de 27 classes e 1200 documentos como base de testes, porém após alguns testes foi obtido uma taxa de acurácia muito baixa se comparado com textos base usados como fundamentação para o projeto. Para que esse problema de classificação fosse resolvido, as 27 categorias foram divididas em subcategorias, na qual elas pudessem ter uma maior afinidade e podendo ter assim um número menor de classes a serem usadas nos treinamentos dos algoritmos. Dessa forma os arquivos de treinamento também foram divididos a partir dessas subcategorias, melhorando a assertividade dos algoritmos durante os testes.

Assim os dados de treinamento são carregados pelos algoritmos através de um arquivo *.txt* que possui os dados e a quais categorias cada linha do arquivo pertence.

Por fim, após o treinamento do método de categorização, o conteúdo que se deseja classificar é enviado para o algoritmo que irá especificar a qual categoria a informação pertence.

Na figura 3.8 é apresentado o fluxo que recebe os dados da requisição, contendo as informações a serem categorizadas, e em seguida envia essas informações para o método classificador.

```
@api_view(['POST'])
def categorization_content_history(request):
    classifier = Classifier()
    if request.method == 'POST':
        print("Classifier content")
        request_data = JSONParser().parse(request)
        contents = []
        for element in request_data:                    # Montagem dos que serão enviados para o metodo de classficação
            content_page = PageContent(**element)
            contents.append(content_page)

        return_values_svm = classifier.do_classification(contents)      # Chamada do classifier
        serializer = PageContentSerializer(return_values_svm, many=True)
        serializer.data
        return JsonResponse(serializer.data, status=status.HTTP_200_OK, safe=False, content_type="application/json") # Retorno da chamada
```

Figura 3.8: Fluxo de recebimento e retorno dos dados já classificados

4 Resultados

Neste capítulo serão apresentadas as atividades desenvolvidas para a criação da arquitetura e da ferramenta, juntamente com o fluxo de teste dos métodos de categorização realizados como forma comparativa e de entendimento destes métodos.

Desta forma o presente capítulo apresenta os resultados obtidos e foi dividido nas seguintes subseções: subseção 4.1 - Análise da Ferramenta Desenvolvida, subseção 4.2 - Classificadores.

4.1 Análise da Ferramenta Desenvolvida

A interface implementada para o trabalho consiste em duas telas (vide Figuras 4.1 e 4.2): a primeira tela responsável pela seleção da disciplina e sub-área desta disciplina e a segunda tela responsável pela exibição dos dados retornados por todo o fluxo de categorização realizado. Esta tela de exibição contém os dados da página, juntamente com a sua categoria e o link para o acesso a página retornada.

Na interface criada é possível o usuário selecionar qual categoria ele deseja buscar. Assim a busca será realizada usando a categoria selecionada (vide Figura 4.2).

Figura 4.1: *Select box* para escolha de categoria

Após a consulta é possível visualizar a interface (tela) dos resultados pesquisados na ferramenta desenvolvida, que apresenta os dados recuperados em uma tabela HTML composta por um breve **resumo** do conteúdo retornado, a **URL** que levará o usuário para a página que possui o conteúdo completo e o **conteúdo** da categoria selecionada, como apresentado nas figuras 4.3 e 4.4.

Uma análise prévia da usabilidade da ferramenta desenvolvida a considera como

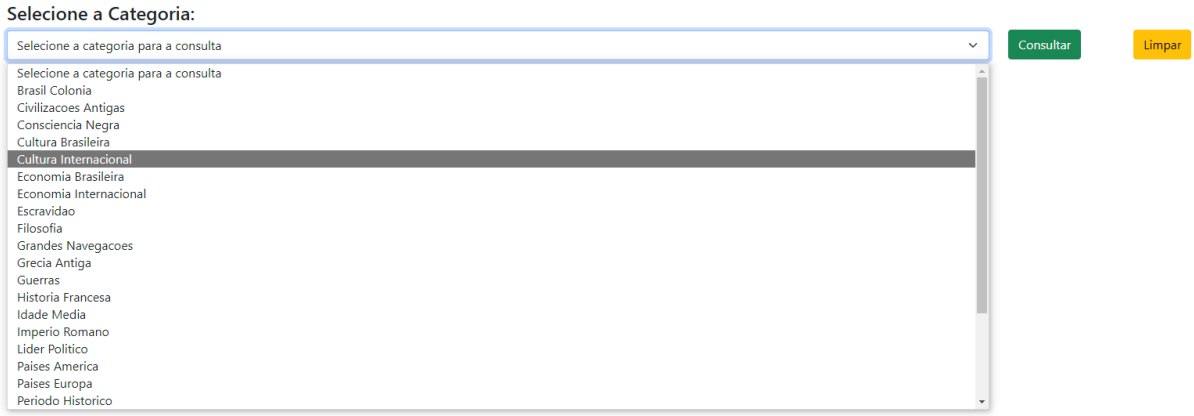


Figura 4.2: *Select box* com as opções de categoria

simples e de fácil compreensão por seus usuários finais, levando em consideração campos de pesquisa simplificados e filtros de busca que se deseja executar.

A apresentação dos resultados também é clara e resumida, onde os dados resultantes da categorização são apresentados de forma simples e contendo um resumo das informações contidas no site que possui o conteúdo inteiro.

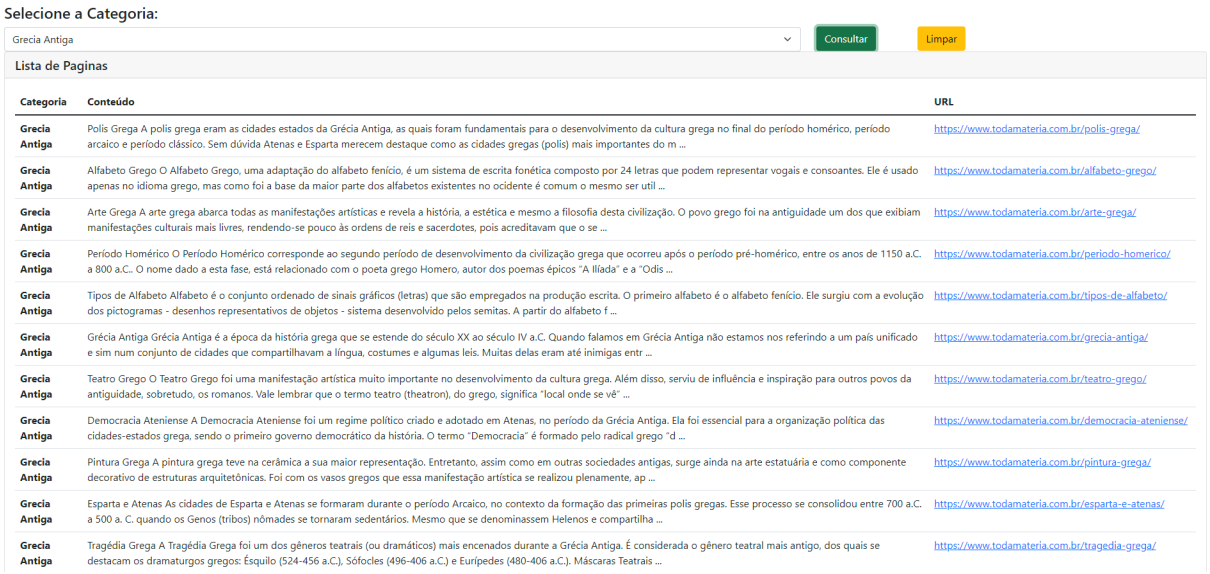


Figura 4.3: Lista com as páginas retornadas para a categoria de Grécia Antiga

As informações apresentadas na tabela com a resposta da consulta possui também o link que redireciona o usuário para a página principal do site que possui a informação completa, agilizando o acesso do usuário ao conteúdo pesquisado.

A ferramenta implementada abstrai algumas informações que muitas das vezes não são de interesse do usuário, trazendo a ele uma forma mais estruturada e clara de acessar os dados contidos nas páginas, sem que o mesmo tenha que ficar navegando nas

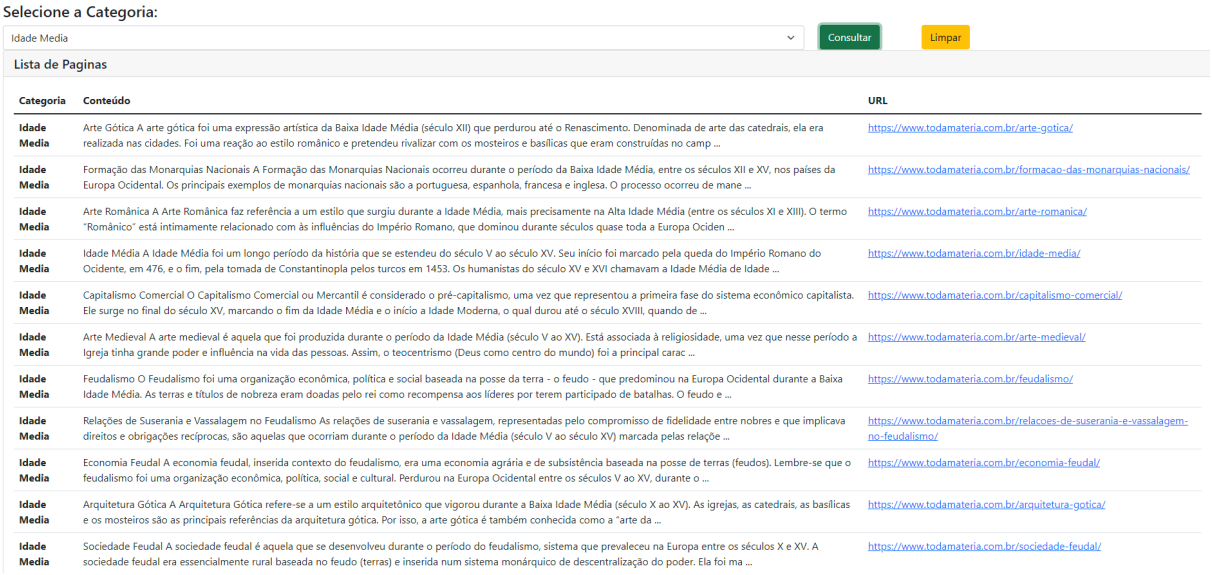


Figura 4.4: Lista com as páginas retornadas para a categoria de Idade Média

páginas à procura do conteúdo desejado.

A ferramenta desenvolvida tem como principal finalidade criar um ambiente no qual o usuário possa realizar a consulta por conteúdos e categorias específicas dentro de uma área específica, no caso neste trabalho materiais educacionais.

Tomando como base as páginas usadas para a captura das informações (Só História e Toda Matéria) como base comparativa para as buscas, a ferramenta se comporta de uma forma mais rápida e de fácil entendimento quando comparada com essas outras páginas pelo fato de o usuário não ter que ficar navegando pela página até que consiga encontrar o material que deseja.

Tendo em vista a arquitetura desenvolvida, através do módulo de persistência dos dados categorizados, a busca se torna mais ágil pelo fato dos dados categorizados já estarem salvos na base de dados, por consequência, criando uma base de dados que agregará várias informações vindas de páginas web coletadas pelos *crawlers*, gerando assim uma base de dados que pode ser utilizado por outras aplicações futuras.

A arquitetura, por utilizar um conceito modular, onde cada *feature* criada fica contida em módulos separados, onde cada um fica responsável por uma tarefa específica, possibilitando a arquitetura absorver novas *features* que possam ser criadas futuramente, trazendo novas funcionalidades para ela a ferramenta

4.2 Classificadores

Foram utilizados algoritmos para a classificação de conteúdos, no software desenvolvido, baseados em três métodos de aprendizagem de máquina, a saber: *Naive Bayes*, *Support Vector Machine* e *Decision Tree*. O objetivo de usar três métodos foi encontrar aquele mais eficaz para se categorizar os dados extraídos das páginas web utilizadas.

Para se calcular a acurácia de cada algoritmo foi criado um fluxo de teste na qual os textos usados como forma de treinamento dos algoritmos foram segmentados baseando-se na técnica de validação cruzada. A validação cruzada é um método de re-amostragem de dados para avaliar a capacidade de generalização de modelos preditivos e evitar o *overfitting* (Berrar (2019), um cenário que ocorre quando, nos dados de treino, o modelo tem um desempenho excelente, porém quando utilizado os dados de teste o resultado é ruim.

A validação cruzada tem como ideia particionar os dados em conjuntos, onde um conjunto é utilizado para treino e outro conjunto é utilizado para teste e avaliação do desempenho do modelo.

No caso dos testes executados a forma de validação utilizada foi o *K-fold*, que consiste em dividir a base de dados de forma aleatória em K subconjuntos, onde uma parte do conjunto é usada como base de teste e a outra como base de treinamento para os algoritmos, conforme apresentado no exemplo da Figura 4.5

ITERAÇÃO 1	TESTE	TREINAMENTO	TREINAMENTO	TREINAMENTO	TREINAMENTO
ITERAÇÃO 2	TREINAMENTO	TESTE	TREINAMENTO	TREINAMENTO	TREINAMENTO
ITERAÇÃO 3	TREINAMENTO	TREINAMENTO	TESTE	TREINAMENTO	TREINAMENTO
ITERAÇÃO 4	TREINAMENTO	TREINAMENTO	TREINAMENTO	TESTE	TREINAMENTO
ITERAÇÃO 5	TREINAMENTO	TREINAMENTO	TREINAMENTO	TREINAMENTO	TESTE

Figura 4.5: Exemplo de amostragem usando a técnico de *k-fold*

O conjunto de dados utilizado para este teste contém cerca de 1200 documentos,

previamente categorizados em 27 classes diferentes, e são usados como forma de treinamento e classificador dos algoritmos testados.

Com os dados a serem testados em mãos, foi utilizado a técnica de K-fold para poder subdividir os documentos a fim de gerar uma validação mais clara dos algoritmos utilizados. Assim os dados foram divididos utilizando um valor de k igual a 5, realizando assim 5 iterações, dividindo os documentos em 20% deles para teste e os 80% restantes como base de treinamento, como apresentado na Figura 4.5.

Inicialmente o conjunto de teste visou validar a capacidade dos algoritmos avaliarem o conjunto de dados levando em conta as 27 classes criadas para os dados selecionados.

Após o teste de acurácia realizado, construiu-se o gráfico da Figura 4.6 evidenciando a baixa acurácia quando os algoritmos utilizados visam classificar os documentos em muitas classes diferentes. Neste caso de teste as acurácias para os algoritmos de Naive Bayes, S.V.M e Decision Tree foram, respectivamente, 57,6%, 48% e 47%.

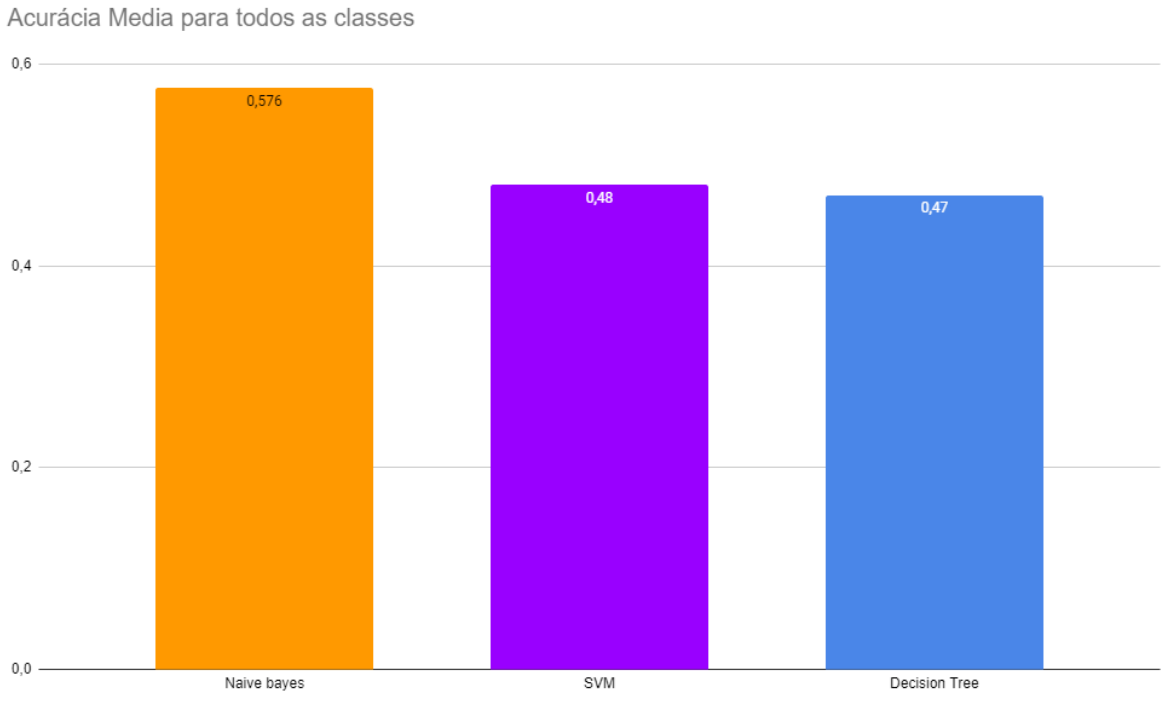


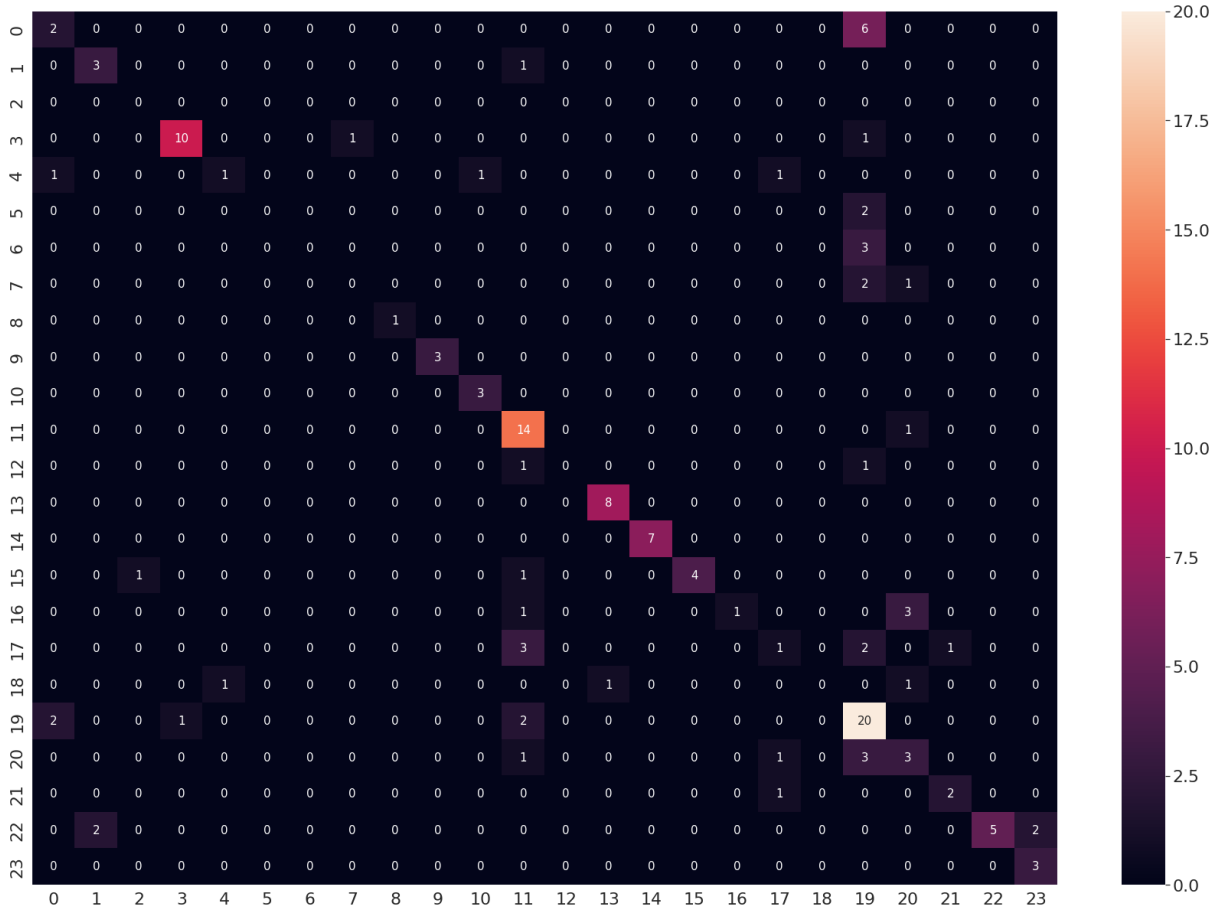
Figura 4.6: Acurácia dos métodos de classificação para todas as classes

Após a realização dos testes foi possível também gerar uma matriz de confusão (Figura 4.7) que apresenta a distribuição dos dados classificados durante o teste e as categorias criadas.

A partir dos dados apresentados na matriz, foi possível notar que existem classes

com poucos documentos gerando um desbalanceamento da base de dados utilizada no treinamento dos métodos, podendo enviesar o resultado obtido. Este problema pode ser notado em uma das classes existentes na matriz, onde ela possui a maioria dos erros de classificação.

Pode se notar também que algumas classes possuem apenas um documento associado a ela, onde o documento seria avaliado sem estar no conjunto de treinamento ou constaria no treinamento, porem não haveria possibilidade de entrar no conjunto de teste, acarretando em um numero menor de classes no conjunto de testes



Eixo x: Representa o predito. Eixo Y: Representa o conjunto verdade

Figura 4.7: Matriz de confusão gerada após os testes.

Tendo em vista os resultados obtidos, quando se visa classificar o conteúdo de forma mais generalista, uma nova abordagem para a execução dos testes e fluxo de classificação dos documentos foi tomada.

Desta forma, para a obtenção de uma melhor acurácia durante o fluxo de categorização dos dados os mesmos foram divididos em sub domínios mais específicos, com o

intuito de diminuir a gama de classes a serem usadas durante a etapa de classificação dos documentos, melhorando assim a acurácia obtida pelos algoritmos.

Como exemplo desta divisão, o corpus usado para o treinamento e para os testes foram divididos em algumas subclasses como: [**Política**], [**História do Brasil**] e [**Períodos Históricos**] e o resultado para cada teste de acurácia para esses subdomínios pode ser visto no gráfico apresentado na Figura 4.8.

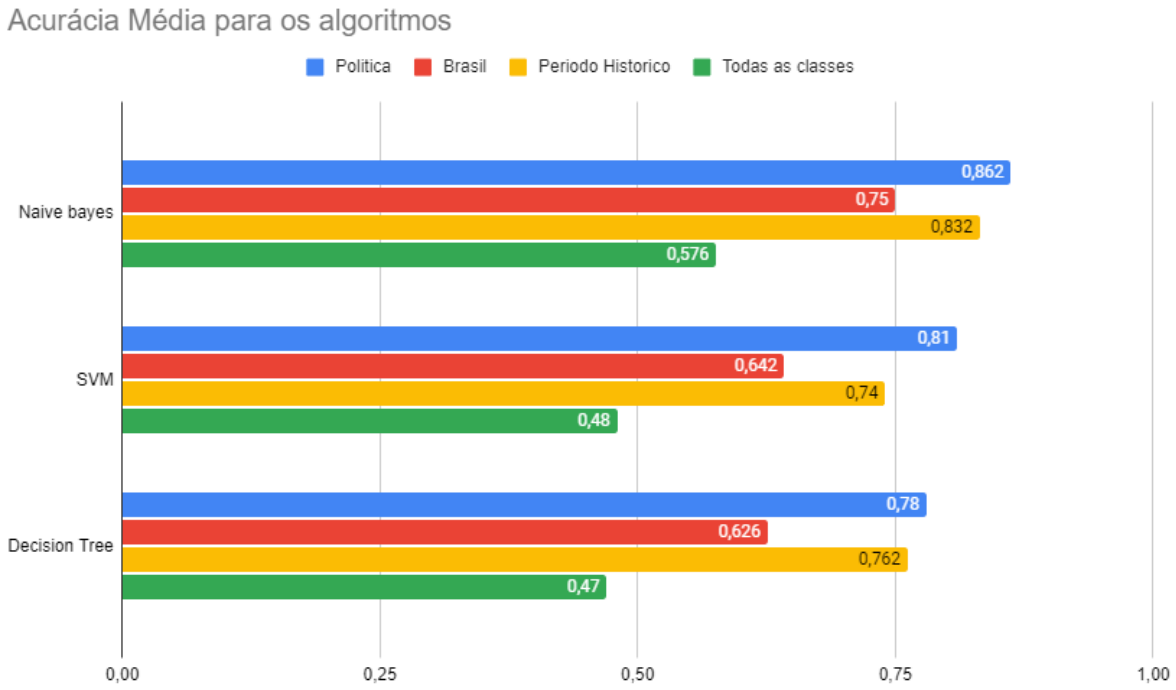


Figura 4.8: Acurácia dos métodos de classificação divididos em subdomínios

Para o subdomínio de Períodos Históricos os documentos também foram divididos em 5 classes diferentes, sendo elas: Civilizações Antigas, Grécia Antiga, Idade Média, Império Romano e Povos Antigos.

Utilizando essas 5 categorias como base para se treinar os métodos de classificação foi realizado o mesmo fluxo de teste feito para os outros subdomínios, gerando os seguintes resultados: *Naive Bayes* com 83,2%, *SVM* com 74% e *Decision Tree* com 76,2%.

Após a realização dos testes dos algoritmos, foi possível visualizar que em ambos os testes o algoritmo de Naive Bayes se saiu melhor, independente do subdomínio utilizado durante os testes.

Um outro aspecto que pode ser observado é a diferença de acurácia entre o subdomínio de Períodos Históricos e Brasil, onde a classificação entre as classes contidas

dentro do domínio de Períodos Histórico se apresentou melhor do que a categorização para as classes do domínio brasileiro.

Após uma análise, para entender qual o real motivo para esta diferença de acurácia, entre estes dois subdomínios da disciplina de História, foi notado que o domínio de Períodos Históricos possui, nos textos usados como base de treinamento e teste, termos muito específicos para cada classe existente neste domínio, diferentemente do contexto Brasil, que possui muitos termos comuns entre as classes usadas no treinamento para dos classificadores, gerando assim essa diferença de acurácia, na qual Períodos Históricos obteve uma melhor acurácia nos três algoritmos utilizados como métodos de treinamento e testes. Isto pode ser visto na Tabela 4.1 e no Gráfico da Figura 4.7.

	Política	Brasil	Período Histórico	Todas as classes
Naive bayes	86,2%	75%	83,2%	57,6%
SVM	81%	64,2%	74%	48%
Decision Tree	78%	62,6%	76,2%	47%

Tabela 4.1: Tabela comparativa com as acurácias dos métodos de *Machine Learning*

Como forma de exemplificar podemos realizar um comparativo entre as classes existentes nestes dois subdomínios. Em Períodos Históricos foi notado, por exemplo, que em documentos referentes a Idade Média, o próprio termo "idade média" era algo que só aparecia em documentos referentes a essa categoria de textos, a mesma coisa acontece quando se analisa a classe de Império Romano, onde termos como *"Império"*, *"Roma"*, *"Romano"*, *"Imperador"*, são comuns em textos voltados para esta categoria dentro de Períodos Históricos.

Diferentemente de Períodos Históricos, o contexto Brasil apresenta muitos termos repetidos nos documentos usados para o treinamento dos algoritmos testados, como por exemplo os termos *"Brasil"*, *"Brasileira"*, *"Rio de Janeiro"*, que acabam por aparecer em diversos documentos de classes diferentes, prejudicando a forma como os algoritmos são treinados, podendo gerar uma falsa classificação de documentos.

5 Conclusões

O volume de dados disponibilizados na web e a facilidade de publicação dos mesmos têm levado a um crescimento acelerado e desestruturado de materiais educacionais no contexto web. As informações estão espalhadas na web, muitas vezes, de forma não organizada, causando um certo desconforto ao usuário durante suas pesquisas, consumindo um grande tempo de busca e afetando de alguma forma o processo de aprendizagem.

Diante desse cenário, este trabalho de conclusão de curso propôs uma ferramenta de busca e recuperação de informação a materiais educacionais utilizando métodos e algoritmos de aprendizagem de máquina, os quais foram testados na recuperação de conteúdos na área de História.

O trabalho realizou uma analise do funcionamento dos métodos de *Machine Learning* e entender o comportamento de cada método utilizado no trabalho tendo em vista os conjuntos de dados usados durante a testes dos algoritmos.

Foi possível entender como os documentos educacionais estão dispostos na internet e ter uma visão da pesquisas mais recentes que visão a classificação e recomendação de documentos voltados para a área da educação. Foi possível entender o quão amplo são os cenários de utilização de métodos de classificação e as formas que são utilizados.

Como contribuição do trabalho, entende-se que a ferramenta desenvolvida possibilita uma nova forma de realizar consultas por materiais educacionais, trazendo ao usuário uma nova possibilidade de acessar o conteúdo que deseja, trazendo uma interface mais simples e com poucas interações para a realização das consultas

Outro ponto que pôde ser observado no trabalho foram as técnicas usadas para a categorização dos documentos, onde foi possível entender o comportamento de cada método utilizado durante o desenvolvimento, levando a compreender como a base de dados usada para o treinamento dos algoritmos pode afetar diretamente na acurácia e na forma como os documentos são classificados.

Outra contribuição da pesquisa é a forma como a arquitetura foi idealizada, utilizando módulos, onde cada um tem sua responsabilidade, auxiliando na forma como testes

foram executados, proporcionando uma também uma forma fácil para se integrar novas funcionalidades.

Desta forma podemos desenvolver, futuramente, algumas melhorias relacionadas à arquitetura proposta inicialmente, adicionando alguns novos módulos que serão capazes de adicionar a plataforma um maior dinamismo no que se trata dos conjuntos de dados usados para o treinamento dos algoritmos de classificação e também uma maior abrangência das disciplinas apresentadas na plataforma. Essa arquitetura otimizada é apresentada na figura 5.1.

Como trabalho futuro é possível a criação de novos conjuntos de dados para o treinamento dos algoritmos, criando classes mais especificas, para que o problemas encontrados durante a analise da matriz de confusão não ocorra.

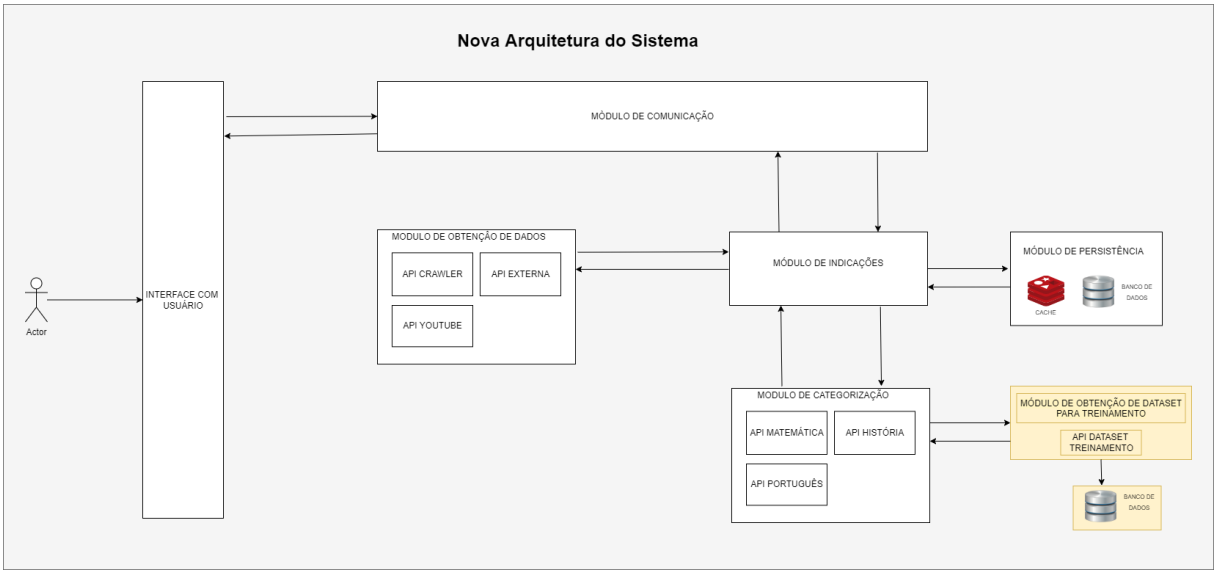


Figura 5.1: Nova arquitetura proposta pelo aluno

Nesta nova arquitetura é proposto um novo módulo, que terá como função integrar novos conjuntos de dados para serem utilizados no fluxo de classificação das páginas, além de poder contar com novas *features* de classificação adicionando novas disciplinas para o sistema. Este novo módulo poderá atuar como uma aplicação que poderá disponibilizar para a ferramenta criada uma forma de aumentar a gama de informações contidas e usadas durante o fluxo de categorização.

Este novo módulo também poderá agregar novos *crawlers*, métodos de classi-

ficação e até mesmo bases de treinamento para estes algoritmos, vindos de comunidades externas que queiram contribuir para o crescimento da ferramenta proposta. A possibilidade da criação deste novo módulo acontece pelo fato da arquitetura desenvolvida utilizar um conceito modular, na qual novas *features* possam ser plugadas na arquitetura, sem que os outros módulos sejam afetados. Assim será possível criar uma arquitetura mais abrangente a novas páginas de *dataset* e outros conteúdos educacionais, além da área de História, usada como testes nesta pesquisa.

De maneira geral, espera-se contribuir com professores, alunos e outros profissionais da educação em sua tarefa de recuperar materiais educacionais relevantes e adequados ao propósito inicial das informações requeridas, de forma mais eficiente, funcionando assim como um instrumento de apoio à recuperação de informação inserida no processo de ensino e aprendizagem.

Bibliografia

ABDI, H.; WILLIAMS, L. J. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, Wiley Online Library, v. 2, n. 4, p. 433–459, 2010.

APUK, V.; NUÇI, K. P. Classification of pedagogical content using conventional machine learning and deep learning model. *arXiv preprint arXiv:2101.07321*, 2021.

BALOGLU, O.; LATIFI, S. Q.; NAZHA, A. What is machine learning? *Archives of disease in childhood. Education and practice edition*, England, p. edpract–2020–319415, 2021. ISSN 1743-0585.

BERRAR, D. *Cross-Validation.* 2019.

CAMBRONERO, C. G.; MORENO, I. G. Algoritmos de aprendizaje: knn & kmeans. *Intelgencia en Redes de Comunicación, Universidad Carlos III de Madrid*, v. 23, 2006.

CORDEIRO, K. M. d. A. O impacto da pandemia na educação: A utilização da tecnologia como ferramenta de ensino. Faculdades IDAAM, 2020.

CRESWELL, J. W.; CRESWELL, J. *Research design.* [S.l.]: Sage publications Thousand Oaks, CA, 2003.

DHENAKARAN, S.; SAMBANTHAN, K. T. Web crawler-an overview. *International Journal of Computer Science and Communication*, v. 2, n. 1, p. 265–267, 2011.

EDGCOMB, J. B.; ZIMA, B. Machine learning, natural language processing, and the electronic health record: innovations in mental health services research. *Psychiatric Services*, Am Psychiatric Assoc, v. 70, n. 4, p. 346–349, 2019.

FELDMAN, R.; SANGER, J. et al. *The text mining handbook: advanced approaches in analyzing unstructured data.* [S.l.]: Cambridge university press, 2007.

GONZALEZ, M.; LIMA, V. L. Recuperação de informação e processamento da linguagem natural. In: *XXIII Congresso da Sociedade Brasileira de Computaçao.* [S.l.: s.n.], 2003. v. 3, p. 347–395.

GUPTA, S.; AGARWAL, M.; JAIN, S. Automated genre classification of books using machine learning and natural language processing. In: IEEE. *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence).* [S.l.], 2019. p. 269–272.

JÚNIOR, C. P.; ARAÚJO, R. D.; DORÇA, F. A. Recomendação personalizada de conteúdo instrucional complementar usando repositório de objetos de aprendizagem e recursos da web. In: SBC. *Anais do XXXI Simpósio Brasileiro de Informática na Educação.* [S.l.], 2020. p. 1293–1302.

KAUSAR, M. A.; DHAKA, V.; SINGH, S. K. Web crawler: a review. *International Journal of Computer Applications*, Foundation of Computer Science, v. 63, n. 2, 2013.

LIDDY, E. D. Natural language processing. 2001.

MAHESH, B. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, v. 9, p. 381–386, 2020.

MARANTE, Y. et al. Evaluating educational recommendation systems: a systematic mapping. In: SBC. *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. [S.l.], 2020. p. 912–921.

MEDEIROS, R. et al. Uma análise comparativa entre repositórios de recursos educacionais abertos para a educação básica. In: SBC. *Anais do XXXII Simpósio Brasileiro de Informática na Educação*. [S.l.], 2021. p. 213–224.

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

NASTASE, V.; STRUBE, M. Decoding wikipedia categories for knowledge acquisition. In: *AAAI*. [S.l.: s.n.], 2008. v. 8, p. 1219–1224.

PATIL, A. S.; PAWAR, B. Automated classification of web sites using naive bayesian algorithm. In: CITESEER. *Proceedings of the international multiconference of engineers and computer scientists*. [S.l.], 2012. v. 1, p. 519–523.

PEREIRA, A. S. et al. Metodologia da pesquisa científica. Brasil, 2018.

QAISER, S.; ALI, R. Text mining: use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, Foundation of Computer Science, v. 181, n. 1, p. 25–29, 2018.

RAMOS, J. et al. Using tf-idf to determine word relevance in document queries. In: PISCATAWAY, NJ. *Proceedings of the first instructional conference on machine learning*. [S.l.], 2003. v. 242, p. 133–142.

RESHAMWALA, A.; MISHRA, D.; PAWAR, P. Review on natural language processing. *IRACST Engineering Science and Technology: An International Journal (ESTIJ)*, v. 3, n. 1, p. 113–116, 2013.

RUDIN, C.; WAGSTAFF, K. L. *Machine learning for science and society*. [S.l.]: Springer, 2014.

SANTANA, V. V. de et al. A importância do uso da internet sob o viés da promoção interativa na educação em tempos de pandemia. *Brazilian Journal of Development*, v. 6, n. 10, p. 78866–78876, 2020.

SANTOS, E. Educação online para além da ead: um fenômeno da cibercultura. *Educação Online: cenário, formação e questões didático-metodológicas. Rio de Janeiro: Wak Editora*, 2010.

SOARES, V. H. A. S. Combinações de similaridade semântica e frequência de termos para agrupamento de textos. Universidade Federal de Viçosa, 2017.

WEBER, C. et al. Construção de um corpus anotado para classificação de entidades nomeadas utilizando a wikipedia e a dbpedia. Pontifícia Universidade Católica do Rio Grande do Sul, 2015.

WOOGUE, P. D. P.; PINEDA, G. A. A.; MADERAZO, C. V. Automatic web page categorization using machine learning and educational-based corpus. *Int. J. Comput. Theory Eng*, v. 9, n. 6, p. 427–432, 2017.

ZERBINATTI, L. *Extração de conhecimento de laudos de radiologia torácica utilizando técnicas de processamento estatístico de linguagem natural.* Tese (Doutorado) — Universidade de São Paulo, 2010.