Plataforma de Sumarização Automática de Artigos Científicos

Gabriel Simões Rocha Reis
Departamento de Computação Instituto Federal da Bahia
Salvador, Brasil
gabrielsrreis@gmail.com

Jowaner Araújo
Departamento de Computação Instituto Federal da Bahia
Salvador, Brasil
jowaneraraujo@gmail.com

RESUMO

Em meio a uma quantidade infindável de dados e a incapacidade humana de manipulá-las, surgiu a necessidade do desenvolvimento de métodos automatizados para auxiliar as pessoas a manipular esses dados. Tendo em vista essa necessidade crescente da sociedade, a sumarização automática de texto surgiu como forma de atender essa demanda tendo como principal objetivo a criação de resumos, contendo as informações principais de um texto. Nos dias atuais, a sumarização automática de texto já pode ser considerada uma área de pesquisa importante e com grande utilidade no meio acadêmica, pois esta está, constantemente, manipulando uma grande quantidade de informações, como por exemplo nos trabalhos científicos. Com base nessa necessidade, esse estudo foi elaborado visando auxiliar os pesquisadores no trabalho de pesquisa e seleção de artigos científicos. Para isso, foi desenvolvido um sistema Web capaz de criar resumos de artigo científicos e armazená-los em banco de dados. Foram realizados experimentos nesse sistema afim de provar a qualidade dos resumos e sua relação de similaridade com o artigo que o gerou. O sistema é limitado a artigos escritos em português do Brasil e é capaz de gerar resumos utilizando dois algoritmos de sumarização, o algoritmo de Luhn e o outro desenvolvido com base no cálculo do cosseno. Além disso, o sistema calcula o tempo de execução dos algoritmos e as métricas de similaridade do cosseno e *containment* entre o artigo e o resumo gerado. Os resultados demonstram que o sistema foi capaz de gerar os resumos e armazená-los em banco de dados. Os resumos criados com 10% das frases do artigo original apresentaram similaridades superiores a 70%, calculada com base no cálculo do cosseno. Também, foram feitas comparações entre o clássico algoritmo de Luhn e o algoritmo implementado, no qual o algoritmo de Luhn demonstrou-se superior em relação as similaridades do cosseno e *containment*.

PALAVRAS-CHAVE

Sumarização Automática, Processamento de Linguagem Natural, Luhn, distancia do cosseno, artigo cientifico, Instituto Federal da Bahia.

1. INTRODUÇÃO

Nos últimos anos, houve um grande avanço da tecnologia. Esse evento possibilitou que as pessoas tivessem acesso a uma grande quantidade de dados. O principal fato que corroborou para isso foi a popularização da internet e dos *smartphones*. Hoje, qualquer pessoa em quase todos os lugares do mundo tem acesso a uma infinidade de informações.

No campo acadêmico, isso trouxe a possibilidade de acessar estudos e pesquisas de todas as partes do mundo com poucos cliques. Graças a essa facilidade, é mais prático buscar trabalhos mais recentes.

A praticidade de ter acesso as informações, trouxe a dificuldade de filtrar e selecionar o que relevante. Principalmente, no campo dos estudos científicos a dificuldade de selecionar aquilo que é importante para determinado pesquisador se tornou um desafio.

Uma forma de condensar informação em texto muito utilizada é a criação de resumos. Devido, a quantidade tão grande informações a criação manual de resumos é inviável. Para isso foi criada a sumarização automática de texto que permite a criação de um resumo contendo as informações mais relevantes de um texto.

Os sistemas de sumarização automática podem ser extrativo ou abstrativo. O extrativo seleciona as frases mais relevantes do texto e as utiliza para criar o resumo. Enquanto, o abstrativo simula a escrita humana para escrever os resumo. Devido a complexidade no desenvolvimento de sistemas de sumarização automática abstrativos, muitas pesquisas são feitas com sumarização extrativa. Com base nisso, foi levantada a questão: é possível criar um sistema de sumarização automática de textos acadêmicos?

O objetivo da pesquisa é a utilização de algoritmos de sumarização para a criação de resumos a partir de artigos científicos, buscando analisar a qualidade desses resumos. Nesse artigo foi desenvolvido um estudo de um sistema *web* de sumarização automática extrativo para artigos científicos. Dessa forma, foi implementado dois algoritmos de sumarização, um deles é o clássico algoritmo de Luhn e o outro foi um algoritmo baseado no cálculo do cosseno. Os dois algoritmos podem ser utilizados para criar resumos. Além disso, foi implementada as métricas de similaridade do cosseno e *containment* para avaliação do resumo criado.

Para isso, esse trabalho foi dividido em mais 4 capítulos. O capítulo 2 contem todo o embasamento teórico que foi utilizado para o desenvolvimento da pesquisa; O capítulo 3 possui a metodologia seguida; O capitulo 4 é apresenta a solução proposta e as discussões sobre os testes; O capitulo 5 é feitas as conclusões sobre a pesquisa e seu resultado.

2. REFERENCIAL TEÓRICO

Para o desenvolvimento desta pesquisa, foi necessária uma revisão literária com o objetivo de um levantamento da fundamentação teórica do tema proposto. Desta forma, são apresentadas as seguintes bases teóricas sobre as quais o produto foi projetado e implementado. A seção 2.1 apresenta a importância e os fundamentos da sumarização automática; A seção 2.2 Conceitos, fundamentos, e processos utilizados de mineração de textos; A seção 2.3 apresenta conceitos relacionados a Processamento de Linguagem Natural(PLN); A seção 2.4 apresenta conceitos básicos do desenvolvimento Web; A seção 2.5 apresenta alguns conceitos sobre *Unified Modeling Language* (UML); A seção 2.4.1 apresenta o *framework* Django e a linguagem Python a qual ele é escrito. A seção 2.6 apresenta a estrutura dos artigos científicos.

2.1 Sumarização Automática

Esse tópico possui um aprofundamento sobre a sumarização automática e seu processo. O sumário é um instrumento auxiliar que serve para atualização ou comunicação. Os sumários de textos, que aqui são utilizados como resumos, são também textos. Por essa razão, um sumário tem com o objetivo de transmitir ou comunicar somente o que é importante de uma fonte textual de informação.

2.1.1 Histórico

Os estudos da sumarização automática não são novos, as primeiras pesquisas são do fim da década de 50, onde se utilizava técnicas de estatística para realizar a extração de informações dos textos para serem criados sumários. O trabalho desenvolvido por Luhn[17], em 1958, utilizou informações estatísticas da frequência das palavras e de sua distribuição no texto para calcular a sua importância no texto.

Uma década depois, a seleção computacional de sentenças possuía um maior potencial de transmitir ao leitor a substância do documento. Desenvolvido por Edmundson[9], ele utilizava as distribuições sugeridas por Luhn[17] além de considerar as "palavras pragmáticas" (*cue words*), que hoje são chamadas de palavras sinalizadoras. Ele já fazia uso de dicionários eletrônicos, para reconhecer os segmentos textuais relevantes para compor os extratos. Também, comparava os extratos automáticos com extratos manuais para avaliar o desempenho em relação àquele baseado na distribuição de frequência[9].

Em 1975, Pollock e Zamora[28] apontaram a necessidade de restringir o domínio para melhorar os resultados. Em seu trabalho, eles adicionaram aos trabalhos anteriores o cruzamento das sentenças com o título da obra, para determinar as sentenças significativas para um extrato. Após a época em que foram criados esses trabalhos, a exploração de métodos nessa linha ficou estagnada. Isso ocorreu devido às impossibilidades técnicas de implementação, limitações de*hardware*e *software* e indisponibilidade de recursos eletrônicos, como dicionários ou repositórios linguísticos de grande porte[28].

Na década de 90, houve o ressurgimento do interesse pela sumarização automática de textos. Alguns dos motivos dessa retomada foram: os computadores de uso geral, a redução de custo das memorias, os etiquetadores morfossintáticos e os *stemmers*[1]. Esses se tornaram disponíveis e o surgimento da metodologia baseada em corpus[2] graças o conhecimento sobre manipulações estatísticas mais elaboradas.

2.1.2 Classificação

Os sumários podem ter formas de classificação diversas, entre as quais, por função, audiência, formação, abordagem e o número de textos-fonte como mostra a figura 1.

Figure 1: Classificações de sumários

Critério	Classificação
Função	Indicativo, informativo ou crítico
Audiência	Genérico ou focado nos interesses do leitor
Formação	Extrato ou *abstract*
Abordagem	Superficial, profunda ou híbrida
Número de textos-fonte	Monodocumento ou multidocumento

Quanto a função os sumários podem ser classificados como descrito abaixo[22]:

- Sumários informativos ou autocontidos: Esses sumários são textos que contêm todas as informações principais do texto-fonte. As informações são organizadas de forma coerente e coesa para que esses sumários possam apresentar uma boa progressão temática, serem gramaticais e legíveis. Pode-se dizer que esses sumários podem dispensar a leitura do texto-fonte;
- Sumários indicativos: Esse sumários não podem substituir o texto-fonte, mas servem para apresentar sobre o que o texto-fonte trata. Por exemplo, os índices são considerados sumários e são classificados como indicativos;
- Sumários críticos ou avaliativos: Esses são sumários que apresentam opiniões sobre o texto além de possuir o conteúdo do texto. Um bom exemplo de um sumario crítico é a resenhas de livros.

Quanto à audiência, os sumários podem ser classificados como[19]:

- Sumários genéricos: São sumários focados no texto e nas informações contidas nele trazendo as informações mais importantes dos textos-fonte, sem nem uma preocupação com quem irão lê-los;
- Sumários focados nos interesses dos leitores: São sumários que focam em trazer aquilo que é interessante

[1]Método de padronização de palavras para encurtar os termos em menor unidade semântica.
[2]Corpus é um conjunto de textos, que caracterizam um estado ou variedade de linguagem.

para um leitor específico. Por exemplo, um sumário direcionado a um leitor leigo no assunto do texto-fonte, terá informações contextuais e mais detalhadas. Enquanto que para um leitor especialista no assunto, deve conter somente a informações essenciais ao texto.

Quanto a formação, os sumários podem ser classificados como[16]:

- Extratos são sumários criados a partir de segmentos inalterados do texto-fonte. Esses são construídos a partir da junção de segmentos classificados como relevantes dos textos-fonte.
- *Abstracts* são textos reescritos com base no texto fonte. Sua construção é baseada nas partes consideradas mais importantes, assim como nos extratos. A diferença é que essas partes são reescritas.

Quanto ao número de textos processados, o processo de sumarização pode ser classificado como[19]:

- Monodocumento: Esse é o sumário tradicional que é produzido de um único texto-fonte
- Multidocumento: Esse sumário é produzido a partir de uma coleção de textos-fonte.

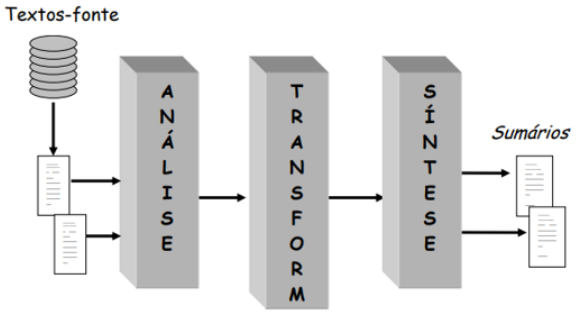
Sumários podem ser construídos basicamente por duas abordagens, a abordagem superficial e a profunda. Essa classificação depende da quantidade e do nível de conhecimento linguístico que será utilizado. A junção das duas abordagens cria a classificação hibrida[27].

2.1.3 Abordagem Superficial ou Empírica

A Sumarização automática que utiliza uma abordagem chamada "superficial" ou empírica, faz uso principalmente de dados estatísticos e empíricos. Em alguns casos, pode-se utilizar um conhecimento linguístico restrito, focando apenas na sintática e morfossintática. Um exemplo clássico da abordagem superficial é a construção de extratos baseada na seleção e justaposição das sentenças do texto-fonte que contêm as palavras mais frequentes do texto. A utilização desse conhecimento serve para decisão de omissão ou não das palavras e dos componentes sintáticos. Sumarizadores da abordagem superficial costumam produzir extratos[27]. A abordagem "superficial" é baseada na arquitetura geral representado na figura 2.

A fase de análise dos textos-fonte presente em na arquitetura geral da sumarização automática, consiste em seu pré-processamento, onde se utiliza técnicas de mineração de dados. A seleção das palavras mais importantes e segmentação do texto-fonte em sentenças e a construção das cadeias lexicais. A construção dos extratos é identificado das sentenças que contêm as cadeias lexicais fortes que: focalizam a primeira ocorrência das sentenças no texto-fonte; que identifica as sentenças que possuem os membros mais representativos; e se concentra na significância do tópico indicado pelas sentenças[17][9].

Figure 2: Arquitetura geral de um sumarizador automático[31]



Problema dessa abordagem são as palavras polissêmicas[3] da língua natural, pois não há repositório eletrônico capaz de definir as ambiguidades das palavras. Para fazer essa detecção é necessário o entendimento do contexto, além da dificuldade de identificar as anáforas ou de controlar o nível de detalhe dos extratos resultantes. Não é feito o tratamento interpretativo do material indicado pelas cadeias lexicais. Isso é resolvido pela abordagem baseadas em conhecimento profundo[31].

Os métodos empíricos identificam os segmentos relevantes e os extraem do texto integralmente. Esses segmentos são incorporados ao extrato na ordem em que eles se apresentam no texto-fonte. Dessa forma, a construção de um sumário se resume na justaposição dos segmentos considerados mais importantes[17][9].

2.1.4 Abordagem Profunda

A abordagem "profunda" é referente ao conhecimento linguístico, utiliza teorias e modelos formais da língua para a criação do sumário. Faz uso de analisadores sintático-semânticos e discursivos, léxicos, *wordnets*[4] e gramáticas. É baseado na teoria discursiva *Rhetorical Structure Theory* (RST)[20] e os analisadores automáticos correspondentes[26][21]. Sumarizadores da abordagem profunda podem gerar *abstracts*. Apesar do desenvolvimento relativamente simples de sumarizadores pela abordagem superficial e de seu baixo custo, é consenso na área que os métodos superficiais produzem sumários de qualidade inferior aos sumários produzidos por métodos profundos[19][34].

2.2 Mineração de Texto

Neste tópico, descreveremos, detalhadamente, as etapas de pré-processamento e classificação de documentos de texto, pois estão relacionados ao conteúdo deste trabalho ao longo de toda a etapa de processamento de documentos de texto que são artigos científicos, até a fase de criação de um resumo do texto.

Para realizar o processo de mineração de texto, o texto é primeiramente decomposto em um vetor de palavras, conhecido como vetor de termos. Em sequência um peso é associado a cada termo. A representação deste vetor pode

[3]Palavra que reúne vários significados

[4]É um banco de dados lexical de relações semânticas entre palavras.

ser vista na Equação (1), onde D1 representa o documento, os W um conjunto de palavras extraídos do texto e os P1 os seus pesos associados.

$$D1 = (W1P1, B2P2, ...Vnpn) \quad (1)$$

Este vetor passa por uma série de transformações no processo de modo que, eventualmente, começa a conter uma série de informações, permitindo-o usar uma grande variedade de métodos de extração de informações. Torres (2005)[36] definiu o processo de pré-processamento de um documento de texto nas sete ações consecutivas conforme figura 3.

Figure 3: Estágios de pré-processamento de palavras [36]

2.2.1 *Análise Léxica*

A Análise léxica é a primeira etapa do processo. Aqui, o conceito do "termo"que será adotado. Normalmente, as próprias palavras presentes no texto são definidas como os termos serem utilizadas no vetor, mas existe autores que acreditam na utilização das frases como indicadores de termos afim de melhorar a semântica para a análise.

2.2.2 *Remover as Stop Words*

As *Stop Words* são os termos indesejados que estão presentes no vetor original.Os *Stop Words* são termos que são considerados sem relevância. Eles aparecem com frequência no texto sem contribuírem para a diferenciação entre categorias. São termos irrelevantes para o processo de classificação e cria um ônus adicional de processamento.

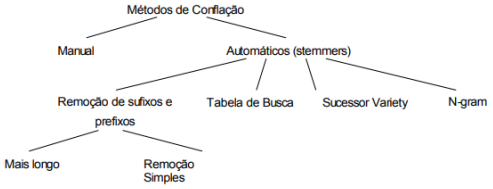
2.2.3 *Redução Morfológica*

A redução morfológica pode ocorrer através de duas técnicas distintas, *"Stemming"* e Lematização". As duas técnicas tem como objetivo redução dos efeitos da variações das palavras em numero, gênero e tempo verbal. Essa variação pode trazer prejuízo na análise que buscas os pontos importantes do texto[6].

A Lematização consiste na transformação das palavras em sua respectivas formas canônicas. Isso torna possível a analise do agrupamento das diferentes desinências das palavras. Dessa forma o processo de lematização permite a desambiguação das palavras[1]. Um dos primeiros trabalhos de lematização foi o projeto WordNet desenvolvido por Miller em 1995 com a lematização de palavra em inglês[24].

O *"Stemming"* é um método amplamente utilizado de padronização de palavras projetado para encurtar os termos em menor unidade semântica possível [25]. A definição é que uma palavra típica tem uma palavra radical (*"Stemming"*) que se refere a alguma ideia central, e que certos prefixos e sufixos são usados para mudar seu significado ou adaptar a palavra a uma determinada regra sintática. *"Stemming"* promove a remoção desses prefixos e sufixos para reduzir uma palavra à sua "essência". Frakes e Baeza-Yates (1992)[10] propuseram uma taxonomia para classificar algoritmos de *Stemming* como mostra a Figura 4. Dentro das categorias propostas, algoritmos de remoção de sufixos e prefixos são os mais comuns na literatura. Entre eles, o mais famoso e usado é o Snowball4, desenvolvido por de Porter[29].

Figure 4: Taxonomia para algoritmos de derivação [10]



O *Stemming* de Porter[29] é baseado na remoção de sufixos de palavras. Alguns dos destaques são o tamanho pequeno, velocidade de processamento e raciocínio simples. Neste algoritmo, os sufixos são removidos simplesmente para melhorar o desempenho dos sistemas de recuperação de informações e mineração de dados, não como um exercício linguístico. As circunstâncias em que o sufixo deve ser removido e a estimativa de valor de remoção do sufixo nem sempre são óbvias.

O método funciona a partir de duas listas, uma com sufixos e outra com condições de exclusão. Quando a condição é atendida, uma operação é realizada na palavra. Quando uma palavra tem poucas letras, o sufixo não é seguido, mesmo se há uma condição que indica a operação. O tamanho de uma palavra é dado pelo número de letras, e não há base linguística para esta observação. Sufixos complexos são removidos em diferentes estágios.

2.2.4 *Normalização de Sinônimos*

As palavras podem ser resumidas em abreviaturas, que possuem o mesmo significado. Por exemplo, a abreviação de Organização das Nações Unidas é ONU e as duas formas devem ser tratadas como a mesma palavra numa classificação. Para isso essas palavras são vistas na lista de termos como o termo de mesmo significado,ou seja, o mesmo termo.

2.2.5 *Atribuição de Pesos*

Para aplicação de varias técnicas, é preciso determinar que alguns termos são mais importantes para a definição do texto do que outros. Por isso, é necessário atribuir pesos que permita a percepção da diferenciação da importância dos termos.

2.2.6 *Redução de Dimensionalidade*

A redução da dimensionalidade é uma operação fundamental para viabilizar a analise do texto e baseia-se na aplicação de transformações e remoções das palavras sobre o

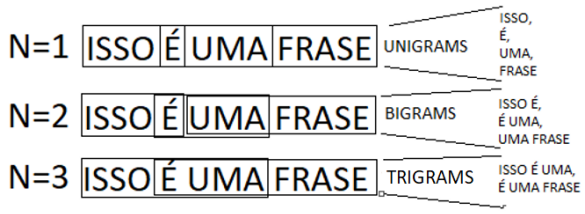
texto criando uma variedade menor de palavras. Técnicas de redução de dimensionalidade objetivam reduzir a variedade de palavras preservando ao máximo o significado do texto.

As técnicas de extração de termos citadas tem como consciência a própria redução da dimensionalidade dos textos. Por exemplo o texto "Eu gosto de estudar.Por isso, eu e minhas amigas estudamos todas as tardes.". Com a remoção das *stopwords*, retirando palavras como "de"e "as", e a lematização do texto, que transformaria a palavra "estudamos"em "estudar", reduziria a dimensionalidade to texto.

2.2.7 *Similaridade de Containment*

A similaridade de *containment* é baseada no modelo n-grams. Um n-grama é uma sequência adjacente de n itens de uma determinada amostra de texto como mostra a figura 5. Esse consiste em uma contagem de ocorrências de palavras. O grande diferencial desse modelo é a vetorização. Pois a contagem não é feita por palavras individuais, e sim por uma sequência contínua de n de palavras. Assim, o modelo é mais adequado para identificar nuancias do texto [18]. A similaridade do *containment* divide o primeiro texto em sequências de n itens e busca no texto de comparação, dessa forma a comparação do resumo com o artigo é diferente da comparação do artigo como o resumo.

Figure 5: Modelo de N-Grams



2.2.8 *Similaridade do Cosseno*

A similaridade do cosseno é uma métrica sofisticada, proposta por Salton e McGill (1987)[11]. Para seu cálculo, é necessário montar uma representação vetorial dos textos comparados e encontrar o valor do cosseno do ângulo formado pelos dois vetores. A vetorização ocorre com a criação do vocabulário como representado na figura 6. Nela está a representação da criação do vocabulário e dos vetores.

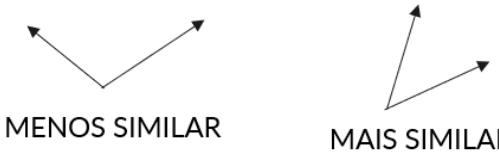
Figure 6: Vetorização das frases

A : EU GOSTO DE BOLO
B : EU GOSTO DE CORRER

VOCABULÁRIO:	BOLO	EU	DE	CORRER	GOSTO
VETOR A :	1	1	1	0	1
VETOR B :	0	1	1	1	1

Quanto mais alto o resultado, mais similares são os textos assim como a figura 7. O vetor de cada texto é formado pelas palavras de cada texto, sendo a posição o número de ocorrências da palavra em questão [30].

Figure 7: Similaridade dos vetores



2.3 Processamento de Linguagem Natural

Com o início da computação, surgiu a necessidade de fazer com que as máquinas entendam as instruções das pessoas. Por isso, surgiram as linguagens de programação, que nada mais é, uma forma de comunicação entre homens com as máquinas. À medida que essas linguagens surgiam estas foram se aproximando cada vez mais da linguagem humana[12]. Apesar de todos os avanços da interpretação de linguagens de programação, as instruções ainda precisão ser exatamente como manda seus padrões caso contrário o mínimo erro pode não ser entendido pelo compilador e causar um erro de sintaxe. Outro objetivo do desenvolvimento das linguagens das máquinas e a comunicação máquina-homem é a capacidade de criar programas que sejam capazes de se comunicar com a língua natural dos humanos. Daí surge o estudo do Processamento de Linguagem Natural (PLN)[12].

O PLN tem inúmeras aplicações, tratando diversos aspectos da comunicação humana como os sons, palavras, sentenças e discursos, considerando formatos e referências, estruturas e significados, contextos e usos. De modo geral, pode-se dizer que o PLN visa fazer o computador se comunicar em linguagem humana. Essa comunicação é feita em diferentes níveis tais como[12]:

- fonético e fonológico: tem como objetivo investigar e estudar os sons das palavras. A fonética é focada em estudar os sons das palavras em sua realização concreta, já fonologia estuda os fonemas das palavras;
- morfológico: é focado no estudo a respeito da estrutura, formação e classificação das palavras a partir unidades de significado primitivas;
- sintático: é a análise do conjunto das regras que determinam as diferentes possibilidades de associação das palavras da língua para a formação de enunciados. É o estudo feito no qual são estudadas as disposições das palavras nas orações, nos períodos, bem como a relação lógica estabelecida entre elas;
- semântico: Esta relacionado aos significados das palavras e de como elas se combinam para dar sentido as sentenças;
- pragmático: é a utilização em diferentes contextos, afetando o significado.

Para se obter a estrutura sintática de uma sentença é preciso realizar o processamento morfossintático obtida por leis gramaticais. As outas informações necessárias, como as categorias morfológicas das palavras, são obtidas a partir de um léxico.

O termo "léxico" significa um conjunto de palavras com suas categorias gramaticais e seus significados. Um léxico é o universo de todos os seus itens lexicais de uma determinada língua, que já utilizou ou poderá vir a utilizar[33]. Utiliza-se o termo "léxico" pois o termo "dicionário" carrega o significado de vocabulário para os leitores[15]. Também, podemos utilizar o termo "léxico" para identificar o componente de um sistema de PLN com informações semânticas e gramaticais sobre itens lexicais. Pode-se utilizar a expressão "base de dados lexical" como sendo uma coleção de informações lexicais.

A etapa inicial da análise textual é o pré-processamento. Nela, é feito a análises léxica e morfológica, sintática e semântica para realizar o reconhecimento e a classificação das entidades. É possível fazer uma relação entre a eficiência da extração das entidades que compõem um texto em linguagem natural e a complexidade do algoritmo. Essa eficiência se comporta de forma exponencial com a complexidade algorítmica. Com a utilização de heurísticas simples, conseguimos obter um acerto de 80% no extrato. Existe uma dificuldade em atingir valores acima de 90%.[2]

A análise léxica de um texto é feito com a extração e classificação dos *tokens*. Um *token* é uma cadeia de caracteres que representa uma parte de uma estrutura. Essa cadeia representa uma palavra ou símbolo em uma linguagem. Na língua portuguesa temos palavras que possuem o mesmo significado, ou até mesmo palavras iguais com significados diferentes, dependendo do contexto. Essas situações são apresentadas na análise morfológica da palavra a depender do contexto. [5]

No PLN, a análise morfológica é realizada no *token*, pois esses correspondem às palavras, no caso da língua portuguesa. Esta classificação também é chamada como etiquetagem. Ela consiste na identificação das classes morfológicas e criação de uma etiqueta. As análises léxica e morfológica dependem uma da outra, por isso podem ser tratadas como uma única etapa.[5]

2.3.1 NLTK

A NLTK é uma biblioteca Python[5] para tratar situações que utilizam PLN. Essa biblioteca é a um projeto gratuito e de código aberto e fornece recursos léxicos, como WordNet, processamento de texto para classificação, tokenização, lematização, marcação, análise e raciocínio semântico. A NLTK está disponível para Windows, Mac OS X e Linux.[3]

Essa biblioteca é bastante utilizada para propósitos acadêmicos e possui, como característica, a facilidade de uso [3]. A NLTK possui uma documentação completa e clara sobre cada ferramenta disponível no *toolkit*. Essa biblioteca também tem uma grande comunidade de usuários disponível na internet através de fóruns e artigos sobre a biblioteca, suas aplicações e resoluções de possíveis problemas.

2.3.2 spaCy

A spaCy é uma biblioteca para processamento avançado de linguagem natural em Python e Cython[6], que tem suporte a mais de 60 línguas. Ela possui modelos de velocidade e rede neural treinados para tarefas de reconhecimento de entidades nomeadas, etiquetação morfossintática, análise de dependência, lematização. Um uso comum da biblioteca spaCy é o reconhecimento de entidades nomeadas. Dessa forma, é possível identificar entidades nomeadas dentro e um conjunto de categorias pré-definidas. Por exemplo, podemos identificar pessoas em que o texto se refere ou até mesmo localizações e Organizações.[35]

A spaCy possui três modelos treinados para a língua portuguesa Esse modelos possuem tamanhos e precisões diferentes que são diretamente proporcionais, quanto maior for o modelo, maior será a precisão. Esse modelos são eles:

1. pt_core_news_sm : Menor modelo com a menor acurácia
2. pt_core_news_md : Modelo intermediário no tamanho e na acurácia
3. pt_core_news_lg : Maior modelo com a maior acurácia

2.4 Sistemas Web

O desenvolvimento de sistemas Web consiste na construção, criação e manutenção de sistema nas redes de internet e intranet. É normalmente desenvolvido em um padrão MVC[7] o que implica em uma divisão entre desenvolvimento de um *front-end* e o desenvolvimento de um *back-end*. Tanto o *front-end* como no *back-end* possui uma pilha de tecnologias que tem como finalidade facilitar e auxiliar no desenvolvimento de um sistema. Uma pilha é um conjunto de *software*, aplicativos, linguagens de programação e ferramentas que são construídas umas sobre as outras para criar e trazer funcionalidades e recursos ao sistema como apresentada na figura 8[32].

O *front-end* representa toda parte vista no "navegador", pelo usuário. São escritos programas para estruturar seus elementos visuais e adicionar interatividade ao sistema. Seus programas são executados através de um *browser* e desenvolvidos em linguagens como HTML[8], CSS[9] e JavaScript[10].

[5]O Python é um linguagem de programação de alto nível, interpretada de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte

[6]Cython é uma linguagem de programação que visa ser um superconjunto da linguagem de programação Python, projetada para fornecer desempenho semelhante ao C com código escrito principalmente em Python com sintaxe adicional opcional inspirada em C.

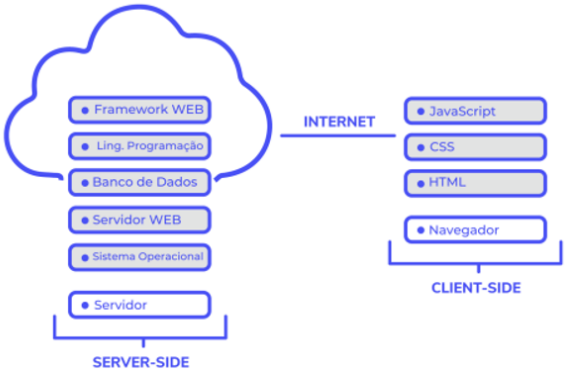
[7]O MVC é o padrão de projeto de software, ou padrão de arquitetura de software focado no reuso de código e a separação de conceitos em três camadas interconectadas.

[8]HTML é uma linguagem de marcação utilizada na construção de páginas na Web

[9]É um mecanismo para adicionar estilo a um documento Web

[10]JavaScript é uma linguagem de programação interpretada estruturada, de script em alto nível com tipagem dinâmica fraca e multiparadigma.

Figure 8: Pilha de tecnologias



A maior parte dos *framework*s para *front-end* são caixas de ferramentas. Alguns dos *framework*s e bibliotecas mais utilizados no *front-end* são Ember[11], React[12], Backbone[13], Vue[14], Angular[15] e Bootstrap[16].[32]

O *back-end* é a parte do programa Web hospedado no servidor. Ele possui os programas que garantam que o servidor, a aplicação Web e o banco de dados possam funcionar bem juntos. Para isso, são utilizadas linguagens do tipo "server-side", como PHP[17], Ruby[18], Java[19] ou Python. Nele, também onde estão os *framework*s *back-end* que auxiliam o desenvolvedor em muitos destes aspectos. Algumas das principais características que os *framework*s *back-end* oferecem são abstração do banco de dados, segurança da aplicação, mapeamento de URLs. Existem diversos *framework*s para *back-end* disponíveis que implementam estas características, alguns dos mais populares são: Spring[20], Express[21], Laravel[22], ASP.NET[23], Ruby on Rails[24] e o Django.[32]

2.4.1 *Django*

O Django é um *framework back-end*, utilizado nesse projeto é escrito em Python. O Python é uma linguagem interpretada de alto nível e de tipagem fraca e dinâmica Ele foi desenvolvida na década de 80 por Guido van Rossum no Instituto de Pesquisa Nacional para Matemática e Ciência da Computação (CWI), na Holanda. Hoje é uma linguagem simples e muito poderosa, muito utilizada em Data Science, Data Analytics, Inteligência Artificial, Deep Learning, Desenvolvimento Web e aplicações de forma geral [8].

O Python é uma linguagem multiplataforma, o que garante portabilidade e compatibilidade em diversos sistemas operacionais e possui código aberto, o que significa que seu código está disponível para todos utilizarem e fazerem suas próprias bibliotecas. É bem integrado com outras linguagens uma vez que ele suporta interação com outras linguagens. Isso permite aos desenvolvedores a possibilidade de utilização de funcionalidades não suportadas em Python, por exemplo algumas escritas em C/C++, Java, C#, .NET, PHP[32].

A proposta do Python é a trazer comodidade para o programador, por isso utiliza poucas palavras, o que implica em um código sucinto. Por estes diferenciais, algumas empresas optam por utilizar-lo.

Graças a popularidade desta linguagem, surgiu o Django, um *framework* para criação de aplicações Web escrito em Python, criado em 2005 por um grupo de programadores do Lawrence Journal-World. A intenção desse *framework* é de tornar mais rápido o desenvolvimento de aplicações Web. O Django tornou-se conhecido por trazer soluções praticas para grande parte dos problemas tradicionais em desenvolvimentos Web.[8]

Diferente de outros *framework*s Web que utilizam o MVC, o Django utiliza a estrutura MTV (Model-Template-View). Essa estrutura o *framework* gerencia a maior parte da comunicação entre requisições HTTP. Um *framework* que utiliza a estrutura MTV, possui as camadas:

- *Model*: No Django, vem com uma solução pronta para mapeamento objeto-relacional. Nele, o esquema do banco de dados é descrito em Python e pode ser utilizada para criar consultas complexas [8].
- *Template*: Esta é a camada de apresentação, onde partes estáticas do arquivo HTML de saída e de partes com uma sintaxe especial que descrevem como o conteúdo dinâmico será apresentado. O Django permite minimizar a redundância entre os *templates* [8].
- *View*: As *views* é a camada que recebe as informações da requisição do lado do cliente e, em seguida, proces-

[11] É um *framework* Web JavaScript de código aberto, baseado na arquitetura Model–view–viewmodel.
[12] React é uma biblioteca JavaScript de código aberto com foco em criar *interfaces* de usuário em páginas Web.
[13] Backbone é uma biblioteca JavaScript inspirada em uma *interface* JSON RESTful que oferece uma arquitetura MV* ou MVC.
[14] Vue é um *framework* JavaScript de código-aberto, focado no desenvolvimento de *interfaces* de usuário e aplicativos de página única.
[15] Angular é uma plataforma de aplicações Web de código-fonte aberto e front-end baseado em TypeScript.
[16] Bootstrap é um *framework* Web com código-fonte aberto para desenvolvimento de componentes de *interface* e front-end para sites e aplicações Web usando HTML, CSS e JavaScript.
[17] PHP é uma linguagem de script de uso geral popular que é especialmente adequada para desenvolvimento Web.
[18] Ruby é uma linguagem de programação interpretada multi-paradigma, de tipagem dinâmica e forte, com gerenciamento de memória automático.
[19] Java é uma linguagem de programação e plataforma computacional.
[20] Spring é um *framework* de código aberto para a plataforma Java.
[21] Express é um *framework* para Node.js que fornece recursos mínimos para construção de servidores Web.
[22] Laravel é um *framework* PHP de código aberto para o desenvolvimento de sistemas Web que utilizam o padrão MVC.
[23] ASP.NET é a plataforma da Microsoft para o desenvolvimento de aplicações Web.
[24] Ruby on Rails é um *framework* escrito em Ruby para construir aplicações Web de forma muito rápida

sam os dados para que sejam armazenados no banco através dos *models* da camada *Model*. A *view* é responsável devolver um objeto HTTPResponse[25] contendo o conteúdo para a página requisitada ou levantar uma exceção como Http404 [8].

Django possui outros recursos que são essenciais que aceleram o desenvolvimento de projetos Web. Alguns deles são:

1. formulários (ou forms), coletam os dados de formulários HTML e os convertem, automaticamente, para objetos Python;
2. mecanismos seguros e robustos de autenticação e permissão, o que permite que seja criado contas e efetuem login ou logout com segurança;
3. armazenamento em cache (caching), o que permite que a renderização[26] das páginas aconteça apenas quando necessário, evitando lentidão na aplicação;
4. Uma *interface* padrão para administração do site, onde é possível criar, exibir e editar quaisquer modelos de dados, entre outros [8].

2.5 UML

A *Unified Modeling Language* (UML) é uma linguagem de modelagem desenvolvida para especificar, elaborar, construir, visualizar e documentar *softwares*. Tem como objetivo auxiliar desenvolvedores a definirem características do *software* antes do *software* começar a ser realmente desenvolvido. Essa modelagem é feita através de diagramas[13].

Os diagramas trazem múltiplas visões do sistema a ser modelado de forma possibilitando analisar e modelar diversos aspectos, procurando atingir a completitude da modelagem. Dessa forma, cada diagrama complementa os outros. Os diagramas provenientes do UML são[13]:

- Diagrama de casos de uso
- Diagrama de classes e diagrama de objetos
- Diagrama de sequência
- Diagrama de estado
- Diagrama de atividade
- Diagrama de interação
- Diagrama de arquitetura
- Diagrama de componentes
- Diagrama de Implantação

Em seguida será descrito sobre cada um dos diagramas oferecidos pela UML, destacando suas principais características.

[25]HTTPResponse é a resposta que o servidor envia ao cliente.
[26]Renderização é o processamento para combinação de um material bruto digitalizado como imagens, vídeos ou áudio e os recursos incorporados ao *software* como transições, legendas e efeitos.

2.5.1 Diagrama de Casos de Uso

É o diagrama utilizado normalmente nas fases de levantamento e análise de requisitos. Ele tem utilidade durante todo o processo de modelagem e desenvolvimento,podendo servir como base para outros diagramas. Precisa ser escrito em uma linguagem simples e de fácil entendimento para que os usuários possam ter noção como o sistema irá se comportar. Esse diagrama busca identificar os atores que utilizarão de alguma forma o *software*. As funcionalidades do sistema, disponíveis aos atores, são conhecidas nesse diagrama como casos de uso.[14]

2.5.2 Diagrama de Classes e Diagrama de Objetos

Esse diagrama é provavelmente o mais utilizado e um dos mais importantes. Ele define as classes utilizadas pelo sistema, assim como sua estrutura. O diagrama de classes apresenta os atributos e métodos de cada classe, além do relacionamento entre as classes.

O diagrama de objetos é associado ao diagrama de classes e é bastante dependente deste. O diagrama de objetos fornece uma apresentação dos objetos e os valores armazenados pelos objetos em um determinado momento da execução de do *software*.[14]

2.5.3 Diagrama de Sequência

O diagrama de sequência é construído de acordo com a ordem temporal em que o fluxo de dados entre os objetos envolvidos ocorre. Ele é baseado em um caso de uso e nos objetos das classes envolvidas em um processo. Um diagrama de sequência, normalmente, identifica o evento que gera o processo modelado, assim como o ator responsável por esse evento. Com isso, ele determina como esse processo deve ser desenrolado e concluído. Para isso, ele apresenta a chamada de métodos disparados entre os objetos.[14]

2.5.4 Diagrama de Estados

O diagrama de estado mostra o objeto e seus possíveis estados e as transações responsáveis pela mudança de estados. Pode ser utilizado para expressar o comportamento de uma parte do sistema e/ou ser utilizado para apresentar o protocolo de uso de parte de um sistema.[14]

2.5.5 Diagrama de Atividade

O diagrama de atividade descreve os passos necessários para finalizar uma atividade específica. Essa atividade pode ser um método com elevada complexidade, um algoritmo, ou um processo completo. Esse apresenta o fluxo de controle de uma atividade.[14]

2.5.6 Diagrama de Interação

O diagrama de interação fornece uma visão geral dentro de um sistema ou processo de negócio[14]. Busca mostrar como os objetos se relacionam entre si para atender aos requisitos. Esse diagrama auxilia na transformação do modelo de domínio em diagrama de classes.

2.5.7 Diagrama de Arquitetura

Esse diagrama pode ajudar *designers* e desenvolvedores de sistemas a visualizar a estrutura de alto nível de um

sistema. Também, é possível utilizar os diagramas de arquitetura para descrever os padrões de projeto. Os diagramas de arquitetura são necessários para desenvolvedores de sistemas esclarecerem e comunicarem ideias acerca da estrutura e requisitos de um sistema.[14]

2.5.8 Diagrama de Componentes

O diagrama de componentes está associado à linguagem de programação que será utilizada. Ele representa os componentes do sistema que será implementado como módulos de código-fonte, bibliotecas, formulários, arquivos de ajuda, módulos executáveis etc. Esse diagrama determina como tais componentes irão interagir dentro do sistema.[14]

2.5.9 Diagrama de Implantação

O diagrama de implantação apresenta os requisitos de *hardware* do sistema, as características físicas como servidores, estações, topologias e protocolos de comunicação. De modo geral, ele apresenta todo o aparato físico necessário para que o sistema seja executado. É um diagrama que permite apresentar como será a distribuição dos módulos do sistema.[14]

2.6 Estrutura de Artigos

Os artigos científicos são uma unidade de informação do periódico científico. Através deles, as pesquisas cientificas são transformadas em conhecimento científico de domínio público. Os artigos científicos, geralmente, apresentam resultados de pesquisa, discute ideias, métodos, técnicas relatos de experiência, estudos de caso etc. De modo, geral um artigo científico se caracteriza por ser um trabalho sucinto, e tem como objetivo transmitir ideias e informações de maneira clara e concisa.[4]

Os artigos científicos possuem uma estrutura que é constituída por elementos pré-textuais, textuais e pós-textuais. Os elementos pré-textuais são compostos por : título, que é o termo ou expressão indicativa do conteúdo do artigo; subtítulo que visa complementar o conteúdo do título; autores que são todas as pessoas responsáveis pela execução e redação da pesquisa; resumo, este sintetiza os objetivos do estudo ou pesquisa, procedimentos básicos e as principais conclusões inferida; palavras-chave que são palavras representativas do conteúdo do documento.[4]

Os elementos textuais são compostos por três partes introdução, desenvolvimento e conclusão.

- Introdução: é a parte dos elementos textuais que se apresenta o tema do artigo, o problema discutido, a definição do assunto abordado, a justificativa e a relevância do trabalho. Na introdução deve-se conter fatos históricos e trabalhos clássicos. Também é nela que se apresenta os principais objetivos e a finalidades da pesquisa, contendo especificação dos aspectos abordados, o tema, o problema a ser resolvidos e a metodologia utilizada;
- Desenvolvimento: é onde está o conteúdo principal do artigo. Este contém o referencial teórico e os procedimentos metodológicos. É nessa parte que é feita a exposição do assunto tratado,discussão do tema da pesquisa e a afirmação das ideias/fundamentos;
- Conclusão: é a parte que apresenta as conclusões correspondentes aos objetivos estabelecidos e às hipóteses lançadas no início. Para isso, é retomada as principais ideias do desenvolvimento e apresentada um desfecho do trabalho a partir dos resultados obtidos. É nessa parte que o autor se posiciona, fazendo sugestões para novas pesquisas e apresentando seu ponto de vista.

Por fim os artigos científicos possuem os elemento pós-textuais como: título em língua estrangeira, subtítulo em língua estrangeira, resumo em língua estrangeira, Palavras-chave em língua estrangeira, notas explicativas, referências, glossário, apêndices e anexos. De modo geral os elementos pós-textuais possuem dados complementares. Também possui itens obrigatórios como as referencias utilizadas.[4]

3. METODOLOGIA

Objetivo do artigo é descrever a construção de uma plataforma de sumarização de artigos científicos utilizando algoritmos de similaridade do cosseno e Luhn.

O método utilizado para o desenvolvimento dessa pesquisa está dividida nas seguintes fases:

1. Levantamento das premissas;
2. Desenvolvimento do programa;
3. Testes;

3.1 Levantamento das Premissas

O levantamento de premissas consiste em buscar os objetivos do projeto e delimitar o escopo. A fase de levantamento das premissas consiste em:

- Buscar a viabilidade do projeto;
- Levantamento dos requisitos do sistema;

Verificar a viabilidade do projeto consiste em buscar na literatura há existência de pesquisas com o mesmo escopo, a viabilidade do projeto com base na tecnologia existente e na validação dos ganhos que a pesquisa pode trazer.

O levantamento de requisitos tem como finalidade descobrir os requisitos do sistemas,todas as necessidades técnicas e todas as funções que o sistema precisa possuir. Nessa parte do trabalho, serão vistas as funções do sistema.

3.2 Desenvolvimento do Programa

O desenvolvimento do programa consiste na criação do sistema de sumarização focado na estrutura de artigos. Dessa forma, será criado um sistema que gere um resumo a partir do elementos textuais de um artigo, ou seja, introdução, desenvolvimento e conclusão. Nessa fase será implementada tanto a lógica de sumarização automática quanto uma interface amigável e intuitiva. Logo, o desenvolvimento do programa será construído, segundo as seguintes etapas:

- Prototipação da interface;

- Diagrama de sequência;
- Desenvolvimento arquitetura do programa;
- Diagrama de classe;
- Implementação do programa;

O desenvolvimento dos casos de uso mostrará todas as necessidades do programa e como se espera para utilizá-lo. Em seguida, será feita a prototipação da interface onde será decidida como as funcionalidades serão apresentadas ao usuário e como ele irá interagir com o sistema. A prototipação da interface será feita em conjunto com o diagrama de sequência, onde serão documentados os fluxos do usuário e dos dados dentro do sistema enquanto o utiliza.

Também, será feito o diagrama de classes com as classes necessárias para desenvolver o sistema. A partir disso, se tem o que será preciso para se desenvolver a arquitetura do programa. Para isso, será utilizado a linguagem de programação Python. Utilizando o padrão de projeto *model-template-view* (MTV), este modelo é utilizado dentro do Django que é o *framework* Web que será utilizado para o projeto.

Na implementação, o sistema será codificado, construindo os requisitos do sistema em forma de funcionalidade.Nessa etapa, serão elaborados nesta ordem:

- Desenvolvimento da interface com usuário;
- Desenvolvimento da persistência em banco;
- Desenvolvimento dos algoritmos de sumarização ;
- Desenvolvimento dos algoritmos de similaridade entre textos;

O desenvolvimento começa pela interface com usuário, que é formado pelas telas e do fluxo de tela, para isso será utilizado Html, Css e Javascript. Logo após, o desenvolvimento da persistência terá 2 etapas. A etapa inicial, onde se utilizará o banco Sqlite para a implantação e testes de desenvolvimento e a etapa final o banco Postgres para a finalização e implantação do sistema.

O desenvolvimento dos algoritmos de sumarização iniciará com a implementação do processamento de textos para a sumarização que consiste em, nesta ordem:

- *Paragraph Splitter* (divisão dos parágrafos);
- *Sentence Splitter* (divisão das sentenças);
- *Case Folding*;
- *Tokenization*;
- Remoção das *Stop Words*;
- Remoção da pontuação;
- Remoção de dígitos;
- *Lemmatization*;

Para o processamento dos textos será utilizada a biblioteca spaCy de processamento de linguagem natural (PLN), a biblioteca NLTK no cálculo da distância do cosseno e a biblioteca de algoritmos de sumarização automáticos Sumy para o algoritmo de Luhn.

O desenvolvimento do cálculo das similaridades será feito pelo algoritmo da similaridade do cosseno, utilizando a mesma biblioteca NLTK para calcular a distância do cosseno e a similaridade de *containment* para indicar também a similaridade entre o texto e o resumo criado.

3.3 Testes

Por fim, serão feitos os testes para validar os resultados do programa e da própria pesquisa, que serão divididos em duas partes:

1. Primeira parte: Utilizará os algoritmos de sumarização Luhn e o Cosseno. Esses gerarão resumos, com diferentes tamanhos, do mesmo artigo científico. O tamanho do resumo é dito pelo percentual da quantidade de frases em relação ao texto original.

   A partir dai, será analisada as similaridades já citadas. Elas representam o quão similar está o artigo do resumo. Além disso será feita a análise do tempo de processamento gasto de geração dos resumos.

2. A segunda parte: Será feita uma bateria de testes utilizando os resumos da massa de dados do Seminário de Difusão e Integração de Conhecimentos (SeDICon)[27]. Cada artigo originará dois resumos com a mesma porcentagem de frases do texto original empregando os algoritmos de sumarização Luhn e o Cosseno. Nesse teste, a qualidade dos dois algoritmos de sumarização Luhn e o Cosseno serão avaliadas, através das similaridade do cosseno e do *containment*.

4. SOLUÇÃO PRPOSTA

4.1 Levantamento das Premissas

A primeira parte do trabalho foi o levantamento das premissas, iniciou-se com a primeira premissa "É possível fazer um algoritmo de sumarização automática". Partindo disso seguiu-se uma etapa de investigação de artigos e trabalhos na área, com a a leitura de vários artigos para buscar a viabilidade do algoritmo de sumarização automática. Artigo lidos nessa etapa foram:

- A sumarização automática de textos: principais características e metodologias;
- Uma plataforma para sumarização automática de textos independente de idioma;
- Sumarização automática;
- Protótipo para sumarização automática de textos escritos em língua portuguesa;

[27] shorturl.at/dmAY9

- Um método de sumarização automática de textos através de dados estatísticos e processamento de linguagem natural;
- Avaliação do desempenho de um software de sumarização automática de textos;
- Uma comparação sistemática de diferentes abordagens para a sumarização automática extrativa de textos em português;
- Avaliando algoritmos de regressão para sumarização automática de textos em português do Brasil;
- Comparativo entre o algoritmo de Luhn e o algoritmo do Cosseno para sumarização de documentos;

Com as leituras ficou claro que estudos sobre algoritmos de sumarização automática já existem. Pois foram encontrados vários artigos criados com diferentes tipos de algoritmos de sumarização analisando sobre seus aspectos e suas vantagens. Dessa forma, a premissa foi adaptada para: " É importante a construção de uma plataforma de sumarização de artigos científicos".

O método de sumarização escolhido foi a sumarização por extração. Essa escolha foi baseada no fato de ser o método mais estudado no meio científico e do alto custo de desenvolver *abstracts* como citado na seção 2.1.

Os algoritmos de extração escolhidos na sumarização foram o algoritmo de Luhn e o algoritmo implementado baseado no cálculo do cosseno. O cosseno é um algoritmo baseado em grafos, seguindo as ideias originalmente apresentadas por Mihalcea[23]. Já o Algoritmo de Lunh analisa a frequência e distribuição das palavras com objetivo de calcular a importância das sentenças visando criar os resumos. Após as definições dos algoritmos de sumarização, foi feito o levantamento de requisitos da plataforma, baseada nas seguintes premissas:

- "É possível criar um sistema de sumarização automática de textos acadêmicos."

Com base nessa premissas, foram levantados os seguintes requisitos do sistema:

1. Cadastro do usuário figura 9: Qualquer pessoa que acesse o sistema, pode efetuar o cadastro de novo usuário para o sistema. Após efetuar o cadastro de um usuário, a pessoa terá acesso ao sistema com seu e-mail e senha cadastrada;
2. Acesso do sistema figura 10: Para acessar o sistema e ter acesso a suas funcionalidades, o usuário deve fornecer o *login* e a senha que foram previamente cadastrados. O sistema deve possuir um usuário administrador com *login* "admin@admin.com" e senha "1234";
3. Sair do sistema : O usuário pode realizar o *logout* do sistema. O *logout* deve ser de fácil acesso e está disponível em qualquer momento e em todas as telas do sistema;

Figure 9: Casos de Uso

Figure 10: Casos de Uso

4. Adição de um novo artigo figura 11: O usuário pode adicionar um novo artigo no sistema para poder criar resumos. O sistema deve salvar os elementos pré-textuais de uma artigo como: o título do artigo, as palavras-chave e os autores. Além disso, deve-se informar se esse artigo é público para outros usuários visualizarem ou não;

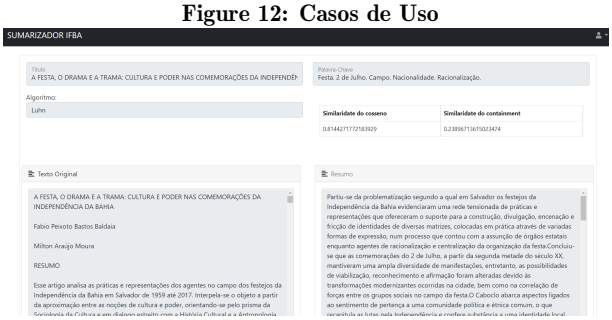
Figure 11: Casos de Uso

5. Edição do artigo figura 11: O usuário pode editar um artigo no sistema e seus resumos. O sistema deve permitir editar o título do artigo, as palavras-chave e os autores. Além disso, deve poder editar se esse artigo é público para outros usuários visualizarem ou não.

Apenas usuários que tenham criado o artigo, podem editá-lo;

6. Exclusão do artigo: O usuário pode excluir um artigo no sistema. Ao excluir um artigo todos seus resumos serão excluídos assim como as informações do artigo. Apenas usuários que tenham criado o artigo, podem excluí-los;
7. Criação de resumo figura 12: O usuário pode adicionar um novo resumo no sistema que ficará vinculado ao artigo. Para criar um resumo deve-se escolher entre os algoritmos de sumarização do cosseno ou o de Luhn. O sistema deve também calcular as medidas de similaridade baseadas na similaridade do cosseno ou *containment*. Cada artigo pode ter mais de um resumo. Apenas usuários que tenham criado o artigo, podem criar resumos dele;

Figure 12: Casos de Uso



8. Edição de resumo: O usuário pode editar o resumo no sistema. Apenas usuários que tenham criado o artigo podem editar os seus resumos. Ao editar o resumo, as medidas de similaridade devem ser recalculadas.
9. *Download* do resumo: O usuário pode fazer download de um resumo no sistema;
10. Busca de artigos: O usuário pode buscar artigos já criados no sistema. Só podem ser listados artigos públicos ou que tenham sido criados pelo usuário. Para a busca, deve ter os filtros de título, autores, palavras-chave, período de criação e se deve retornar artigos públicos ou apenas os seus próprios;
11. Listar artigos figura 13: O usuário pode visualizar a lista de artigos que tem acesso. Cada item da listagem deve apresentar o título do artigo, autores, palavras-chave, usuário que criou e a data de criação.

A funcionalidade de criar usuários permite ao próprio sistema criar novos usuários, sem a necessidade de um operador e tornando-o mais independente. A funcionalidade de controle de usuário permite o acesso e a restrição de acesso aos resumos possíveis e seguras. Uma das principais funções do sistema é criar novos resumos, editá-los e a possibilidade de excluir um resumo criado por quem o criou, além de buscar os resumos já criados. A busca de resumos pode ser feita pelos usuários que o criaram ou por resumos declarados como públicos.

Após a o levantamento das premissas, continuou para o desenvolvimento do programa.

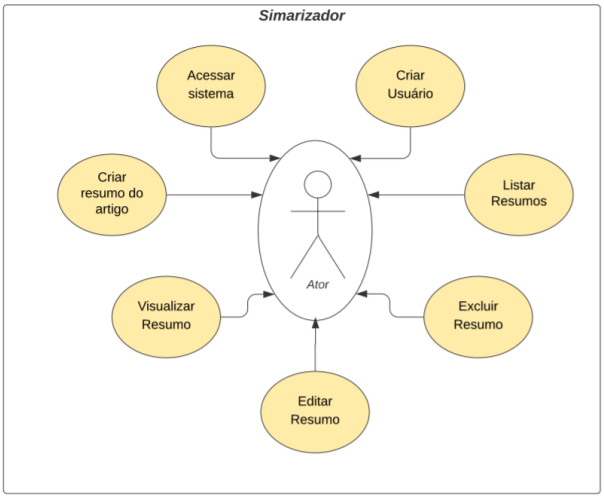
Figure 13: Casos de Uso

4.2 Desenvolvimento do Programa

O início do desenvolvimento do programa foi pela criação do diagrama de casos de uso da figura 14. Os casos de uso foram baseados nos requisitos do sistemas que já tinham sido levantados. Com base nos seguintes casos de uso o sistema foi construído:

1. Acessar no sistema;
2. Criar Usuário;
3. Criar resumo do artigo;
4. Listar Resumos;
5. Editar Resumo;
6. Visualizar Resumo;
7. Excluir Resumo;

Figure 14: Casos de Uso

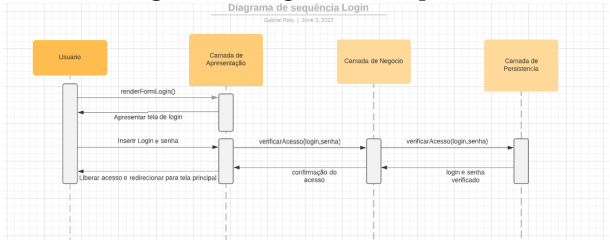


Definindo os casos de uso, pode-se dar início ao desenvolvimento dos protótipos das telas do sistema. É necessário pensar na usabilidade do sistema e como os casos de uso serão apresentados ao usuário. Nesse momento, começou-se a pensar no fluxo de tela juntamente com o fluxo de sistema.

Após a definição dos fluxos, foi construído o diagrama de sequência. O diagrama de sequência foi feito em UML,

representando a sequência de processos no sistema. O diagrama foi desenhado para determinar a sequência global do comportamento. Esse diagrama, foi feito para representar a informação de uma forma simples e lógica como mostra a figura 15.

Figure 15: Diagrama de Sequência



Com o diagrama de sequência elaborado, iniciou-se a definição da arquitetura do sistema. O primeiro passo foi a definição da linguagem. A decisão foi pela utilização do Python, que é uma linguagem extremamente utilizada nos trabalhos e pesquisas de sumarização automática. Ao decidir a linguagem, as bibliotecas spaCy, NLTK e Sumy foram escolhidas por serem bibliotecas de processamento de linguagem natural eficazes. Também foi escolhida a biblioteca Sumy que possui alguns algoritmos de sumarização já implementados. Para as telas(*frontend*) foi decidido a utilização do HTML, Javascript e Css. Essas são linguagens comumente utilizadas no desenvolvimento de telas.

Foi empregado o *framework* Django para o desenvolvimento Web. O Django é um *framework* para desenvolvimento *backend*, desenvolvido em Python, que utiliza o padrão *model-template-view* (MTV) e foi essa a arquitetura adotada para o sistema. Para o *frontend* não foi necessário a utilização de nem um *framework* pela simplicidade das telas do sistema.

Decidindo-se a arquitetura, é possível partir para a criação do diagrama de classe. Nesse momento, foram desenhadas todas as classes necessárias no sistema para desenvolver uma plataforma de sumarização automática. Por fim, desenvolveu-se o sistema efetivamente.

O sistema começou a implementação pela interface gráfica, desenvolvendo todas as telas juntamente com seu fluxo funcional. Por fim, foi criado um protótipo das telas em HTML, Css e Javascript, que foi avaliado para verificar se a usabilidade , buscando permitir que os usuários completem as tarefas desejadas no menor tempo possível.

Quando concluído, o protótipo da interface foi apresentado ao grupo de estudos GEPIO que é constituído de possíveis utilizadores. Nessa apresentação, foi passada a ideia do trabalho e apresentada as telas e os fluxos. Os professores apresentaram seus pontos de vista e indicaram melhorias, como um campo de autores dos artigos para ser apresentado e a possibilidade de busca de artigos por autores.

Com base nas mudanças, foi implementado o desenvolvimento do sistema em Django, utilizando as telas do protótipo. Quando concluída a etapa das tela foi iniciado o desenvolvimento das tabelas no banco de dados. O banco escolhido para o desenvolvimento foi o SqLite por sua praticidade e versatilidade durante o desenvolvimento. Para o produto final foi decidido utilizar o Mysql.Foi desenhado as tabelas:

- Usuário
- Texto
- Resumo

A tabela de Usuário é utilizada tanto para o controle de acesso quanto para definir a restrição de acesso aos textos apenas aqueles que o criaram. Isso é feito através de uma relação entre as tabelas de Texto e Usuário. Enquanto a tabela de Resumo tem uma relação com a tabela de Texto.

Criado o bando e suas conexões, foi iniciado o desenvolvimento dos algoritmos de sumarização automática. O primeiro passo para desenvolver o algoritmo é a implementação do processamento dos textos que serão sumarizados. Para isso foi utilizada a biblioteca de processamento de linguagem natural spaCy.

O processamento começa pela divisão de parágrafos, pois foi adotada a estratégia de sumarização por parágrafo, que consiste em dividir os parágrafos e criar resumos isolados de cada um dos parágrafos.

Com os parágrafos separados, é divida as sentenças do texto. Essas sentenças é feito o *case folding* que é a transformação das letras em letras minúsculas. O próximo passo é a *Tokenization* da sentença, criando uma lista de *tokens*. Dessa lista, é retirada as *stopwords*, que são as palavras como pouca importância para o significado do texto, e a remoção das pontuações e dos dígitos. O último passo do processamento do texto é a *lemmatization* que é o agrupamento das formas flexionadas das palavras para que possam ser analisadas como um único item, esse é chamado como lema da palavra.

Em seguida o algoritmo de sumarização calcula a distância do cosseno para encontrar a importância de cada sentença em comparação com o texto. Para o encontrar o cosseno, foi utilizado a biblioteca NLTK para calcular o cosseno do ângulo formado entre os vetores que representem pares de frases.

Também, foi desenvolvido também a sumarização pelo algoritmo de Luhn para servir de contraponto ao cosseno. Para sua criação, foi utilizada a biblioteca Sumy que possui vários algoritmos de sumarização já desenvolvidos, dentre esse algoritmos foi utilizado o algoritmo de Luhn.

Por fim foi desenvolvido dois algoritmos de similaridade entre textos, um foi o de similaridade do cosseno, baseado no cálculo do cosseno, assim como o algoritmo de sumarização implementado. E o outro foi a similaridade *containment*, que é baseado

4.3 Teste

A fases do testes de software consistem na utilização dos artigos da massa de dados SeDICon, para criar resumos e avaliá-los pelos algoritmos de similaridade que obtém métricas de avaliação entre os resumos e os artigos.

Todos resumos gerados pelo sistema de sumarização automático são classificado como: informativos quanto a função; genéricos quanto a audiência; extratos quanto a formação; monodocumento quanto ao numero de textos.

Na primeira bateria de teste foi realizado com o artigo "Análise de Agrupamentos como Ferramenta de Inovação em Gestão da Saúde: Consumo de Álcool dos Discentes do IFBA/Campus Salvador"[7]. O artigo em questão possui 18 paginas.

Inicialmente foram feitos 18 resumos, sendo eles divididos em 9 para cada algoritmo, dessa forma, foram utilizados o algoritmo de Luhn e o algoritmo desenvolvido que utiliza a similaridade do cosseno. Para cada algoritmo foram gerados os resumos, contendo porcentagens de frases do artigo diferentes, como mostra a tabela 1.

Table 1: Dados dos resumos gerados

Algoritmo	Sim. Cosseno	Sim. Containment	Formato	Tempo Proc.	Percentual(%)
Luhn	83,2%	33,4%	COMPLETO	00:00:08	10%
Luhn	86,4%	53,6%	COMPLETO	00:00:06	20%
Luhn	87,0%	69,5%	COMPLETO	00:00:10	30%
Luhn	87,5%	80,6%	COMPLETO	00:00:10	40%
Luhn	87,7%	89,9%	COMPLETO	00:00:11	50%
Luhn	86,8%	95,0%	COMPLETO	00:00:11	60%
Luhn	85,9%	96,6%	COMPLETO	00:00:11	70%
Luhn	85,9%	96,6%	COMPLETO	00:00:11	80%
Luhn	85,9%	96,6%	COMPLETO	00:00:15	90%
Cosseno	77,7%	17,8%	COMPLETO	00:15:11	10%
Cosseno	80,4%	29,5%	COMPLETO	00:15:46	20%
Cosseno	80,9%	38,4%	COMPLETO	00:16:35	30%
Cosseno	81,7%	49,1%	COMPLETO	00:15:55	40%
Cosseno	81,3%	57,4%	COMPLETO	00:16:06	50%
Cosseno	81,3%	68,5%	COMPLETO	00:16:19	60%
Cosseno	81,1%	80,1%	COMPLETO	00:15:49	70%
Cosseno	80,2%	89,1%	COMPLETO	00:15:52	80%
Cosseno	79,0%	94,7%	COMPLETO	00:16:10	90%

A tabela 1 é formada pela coluna "Algoritmo"que apresenta qual algoritmo de sumarização foi utilizado; A coluna "Sim. Cosseno"apresenta um valor de similaridade entre o texto original e o resumo baseado no cálculo do cosseno; A coluna "Sim. *Containment*"apresenta um valor de similaridade entre o texto original e o resumo criado, baseado no algoritmo do *Containment*; A coluna "Formato"que apresenta se o resumo é do feito do texto todo ou de cada paragrafo; A coluna "Tempo Proc."que é o tempo de processamento; E a coluna "Percentual"que representa a porcentagem de quantidade de frases do resumo em relação ao texto original.

Observamos que o algoritmo de Luhn em um resumo com 10% do artigo obteve uma similaridade superior ao algoritmo do cosseno e essa superioridade se manteve nas outras porcentagens de resumo.

É possível observar as evoluções das similaridades nos gráficos 16 e 17. Neles estão representados os gráficos criados a partir dos dados da tabela 1. O gráficos possuem o eixo X que representa a porcentagem de quantidade de frases do resumo em relação ao texto original em percentual. E no eixo Y está as similaridades, onde a verde é do algoritmo de Lunh e a vermelha é do algoritmo do cosseno.

Figure 16: Gráfico de evolução das similaridade do cosseno

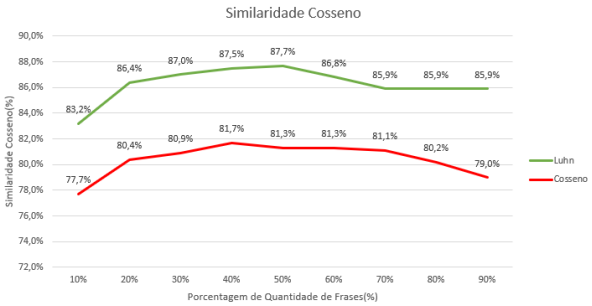
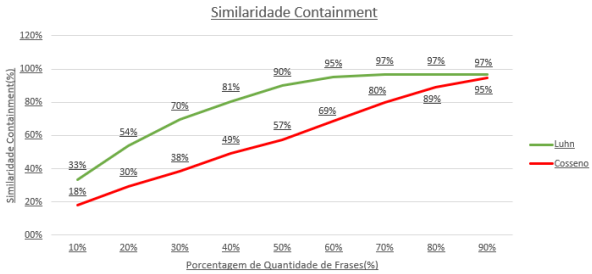


Figure 17: Gráfico de evolução das similaridade do *containment*



Nesses gráficos observamos que um movimento semelhante entre ambas as similaridades, com a similaridade do cosseno mantendo um certa frequência e a similaridade *containment* inciando baixa e crescendo a medida que cresce o Porcentagem de quantidade de frases do resumo. Uma análise importante no gráfico de Luhn está no ponto que o resumo atinge 70% do artigo, pois nesse momento os valores da similaridades se mantém constante.

Outro ponto importante está quando se observa que o algoritmo de Luhn atingiu o ápice da similaridade do cosseno em resumos com 50% da porcentagem de quantidade de frases do artigo. Enquanto no algoritmo do cosseno esse ápice foi atingido no resumo com 40% da Porcentagem de quantidade de frases do artigo.

Uma outra diferença marcante apresentada nos algoritmos, foi o tempo de execução. Com base na tabela 1 foi calculado a média de tempo de processamento do algoritmo de Luhn e o algoritmo do Cosseno. Sendo o tempo de execução do algoritmo do cosseno expressivamente maior do que o tempo do algoritmo de Lunh. Em media, o algoritmo de Luhn demorou 15 segundos para criar um resumo enquanto o algoritmo do Cosseno demorou 15 minutos e 58 segundos.

Ainda em relação ao tempo foi observado uma diferença expressiva na excussão do algoritmo do cosseno desenvolvido em relação ao hardware. O teste com o resumo possuindo 10% das frases do artigo, teve um tempo de 15 minutos e 11 segundos e foi executado observada em uma máquina com o processador Intel Core i5-10210U com 8 GB de memoria RAM. Enquanto outra máquina foi feito o mesmo teste com o mesmo artigo em uma maquina de Intel Pentium CPU B940 com 4 GB de RAM e obteve um tempo de 48 minutos

e 58 segundos.

A segunda bateria de testes foi realizado com 5 artigos da massa de dados SeDICon. Em todos os resumos adotou-se 10% da quantidade de frases do artigo e foram gerados 2 resumos de cada texto, cada um utilizando um algoritmo diferente. A partir dos dados da execuções dos resumos foram gerados os gráficos 19 e 18.

Figure 18: Gráfico com resumos e os valores da Similaridade do Cosseno

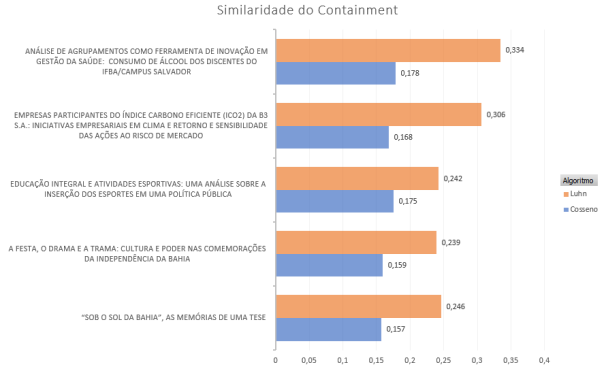
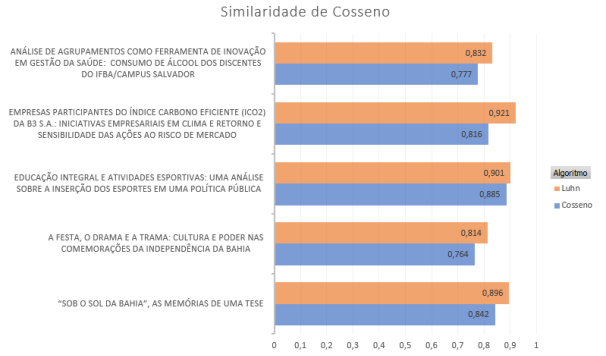


Figure 19: Gráfico com resumos e os valores da Similaridade do *Containment*



Os gráficos do lado esquerdo possuem os nomes dos artigos que foram utilizados para criar os resumos e do lado direito as barras com os valores calculado das similaridades entre os artigos e o resumo criado. Neles estão os valores das similaridades do cosseno representada no gráfico 18 e a similaridade do *containment* representada no gráfico 19.

Como vemos nos gráficos, todos os resumos criados a partir do algoritmo de Luhn obteve uma similaridade do cosseno percentualmente maior que os obtidos a partir do algoritmo do Cosseno. O mesmo ocorreu para a similaridade de *containment*. De modo geral o algoritmo de Luhn obteve melhores resultados que o algoritmo do Cosseno em relação as similaridades do cosseno e *containment*.

5. CONCLUSÃO

Esse estudo inicial possibilitou a criação de uma plataforma de sumarização automática voltada para artigos científicos. Os testes na massa de dados do SeDICon apresentam que a plataforma se mostrou capaz de criar resumos de artigos científicos e armazená-los em base de dados. Tanto os resumos quanto os artigos ficam disponíveis para serem pesquisados futuramente. Além disso, a plataforma é capaz de criar resumos com porcentagens diferentes dos artigos e com diferentes algoritmos de sumarização automática.

Em relação aos algoritmos de sumarização utilizados e com base nos dados coletados pelos experimentos realizados, é possível inferir que o algoritmo de Luhn obteve melhores resultados para os testes de similaridade que o algoritmo desenvolvido baseado no cálculo do cosseno. Além disso, foi possível analisar que o tempo de execução do algoritmo de Luhn é expressivamente menor que o do algoritmo do cosseno.

Para trabalhos futuros é possível introduzir diferentes formas de calcular a similaridade entre o resumo e o artigo. Uma métrica de clareza dos resumos também traria um avanço significativo a pesquisa. Isso possibilitaria uma melhor avaliação dos algoritmos de sumarização, uma boa forma de avaliar os resumos poderia ser os testes com os autores dos artigos, onde eles avaliariam os resumos com base em um barema.

Uma outra proposta para trabalhos futuro seria a adição de uma base de dados inteligente de artigos científicos, acoplada a plataforma de sumarização, para facilitar o acesso aos artigos científicos. Também pode ser utilizado métodos abstrativos para a criação de resumos e avaliar a diferença com os métodos extrativos implementados.

6. REFERENCES

[1] L. Antiqueira. *Desenvolvimento de técnicas baseadas em redes complexas para sumarização extrativa de textos*. PhD thesis, Universidade de São Paulo, 2007.

[2] C. N. Aranha and M. Vellasco. Uma abordagem de pré-processamento automático para mineração de textos em português: sob o enfoque da inteligência computacional. *Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ*, pages 33–34, 2007.

[3] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. "O'Reilly Media, Inc.", 2009.

[4] M. G. Curty and V. R. C. Boccato. O artigo científico como forma de comunicação do conhecimento na área de ciência da informação. *Perspectivas em ciência da informação*, 10(1), 2005.

[5] P. P. O. T. DE CONCLUSÃO. *TÍTULO: PROTÓTIPO PARA SUMARIZAÇÃO AUTOMÁTICA DE TEXTOS ESCRITOS EM LÍNGUA PORTUGUESA ÁREA: Linguística Computacional. Palavras-chave: Sumarização automática. Processamento de linguagem natural.* PhD thesis, UNIVERSIDADE REGIONAL DE BLUMENAU, 2013.

[6] J. De Lucca and M. d. G. V. Nunes. Lematização versus stemming. *USP, UFSCar, UNESP, São Carlos, São Paulo*, 2002.

[7] J. de Oliveira Araújo, J. L. d. A. L. Neto, M. A. da Silva Modesto, A. C. A. Neto, M. T. T. Andrade, and R. S. N. de Argollo. Análise de agrupamentos como ferramenta de inovação em gestão da saúde:

Consumo de álcool dos discentes do ifba/campus salvador.

[8] Django Software Foundation. Django.

[9] H. Edmundson. New methods in automatic extracting. journal of the acm (jacm). *Journal of the ACM*, 16(2):264–285, 1969.

[10] W. B. Frakes and R. Baeza-Yates. *Information retrieval: data structures and algorithms*. Prentice-Hall, Inc., 1992.

[11] M. J. M. Gerard Salton. *Introduction to modern information retrieval*. McGraw Hill Computer Science Series, 1987.

[12] M. Gonzalez and V. L. S. Lima. Recuperação de informação e processamento da linguagem natural. In *XXIII Congresso da Sociedade Brasileira de Computaçao*, volume 3, pages 347–395, 2003.

[13] R. R. Gudwin. Introdução à linguagem uml. *São Paulo: Unicamp. Disponível em:< http://www. dca. fee. unicamp. br/˜ gudwin/ftp/ea976/Estruturais2010. pdf>. Acesso em*, 10, 2019.

[14] G. T. Guedes. *UML 2-Uma abordagem prática*. Novatec Editora, 2018.

[15] L. Guthrie, J. Pustejovsky, Y. Wilks, and B. M. Slator. The role of lexicons in natural language processing. *Communications of the ACM*, 39(1):63–72, 1996.

[16] K. S. Jones. What might be in a summary? *Information retrieval*, 93:9–26, 1993.

[17] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.

[18] L. H. A. Malta and M. A. R. L. Kuroiva. Aprendizado de máquina e processamento de linguagem natural aplicados à identificação de discurso de ódio. 2019.

[19] I. Mani. *Automatic summarization*, volume 3. John Benjamins Publishing, 2001.

[20] W. C. Mann and S. A. Thompson. *Rhetorical structure theory: A theory of text organization*. University of Southern California, Information Sciences Institute Los Angeles, 1987.

[21] D. Marcu. *The theory and practice of discourse parsing and summarization*. MIT press, 2000.

[22] M. Maybury. *Advances in automatic text summarization*. MIT press, 1999.

[23] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

[24] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[25] C. Paice. Another stemmer. *SIGIR Forum*, 24:56–61, 11 1990.

[26] T. A. S. Pardo, L. Antiqueira, M. d. G. V. Nunes, O. N. Oliveira, and L. D. F. Costa. Using complex networks for language processing: The case of summary evaluation. In *2006 International Conference on Communications, Circuits and Systems*, volume 4, pages 2678–2682. IEEE, 2006.

[27] T. A. S. Pardo et al. Sumarização automática: principais conceitos e sistemas para o português brasileiro. 2008.

[28] J. J. Pollock and A. Zamora. Automatic abstracting research at chemical abstracts service. *J. Chem. Inf. Comput. Sci.*, 15:226–232, 1975.

[29] M. Porter. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14, 03 1980.

[30] B. H. Rasteiro, R. A. Monteiro, and T. A. S. Pardo. Processamento de linguagem natural na educaç ao superior: Comparando automaticamente currıculos de cursos de computaç ao.

[31] L. H. M. Rino and T. A. S. Pardo. A sumarização automática de textos: principais características e metodologias. In *Anais do XXIII Congresso da Sociedade Brasileira de Computação*, volume 8, pages 203–245, 2003.

[32] C. P. Santiago, N. L. Veras, A. P. de Aragão, D. A. Carvalho, and L. A. Amaral. Desenvolvimento de sistemas web orientado a reuso com python, django e bootstrap. *Sociedade Brasileira de Computação*, 2020.

[33] I. Scapini. Relações entre itens lexicais. *Poersch, JM; Wertheimer, AMC; Ouro, MEP; Ludwig, EM*, pages 393–429, 1995.

[34] K. Spärck Jones. Automatic summarising: a review and discussion of the state of the art. 2007.

[35] B. Srinivasa-Desikan. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[36] J. A. S. Torres et al. Sumarização automática de artigos científicos de engenharia de software como suporte ao processo de revisão sistemática. 2011.