CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

André Fernandes Bispo

AUTOMAÇÃO DO PREENCHIMENTO DE UM GRAFO DE CONHECIMENTO APLICADO A ARTIGOS DE UM PORTAL DE NOTÍCIAS ESPORTIVAS

Palmas – TO

2021

André Fernandes Bispo

AUTOMAÇÃO DO PREENCHIMENTO DE UM GRAFO DE CONHECIMENTO APLICADO A ARTIGOS DE UM PORTAL DE NOTÍCIAS ESPORTIVAS

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Engenharia de Software pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Jackson Gomes de Souza.

Palmas – TO

2021

André Fernandes Bispo

AUTOMAÇÃO DO PREENCHIMENTO DE UM GRAFO DE CONHECIMENTO APLICADO A ARTIGOS DE UM PORTAL DE NOTÍCIAS

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Engenharia de Software pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Jackson Gomes de Souza.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e Jackson Gomes de Souza

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fabiano Fagundes

Centro Universitário Luterano de Palmas – CEULP

_____

Profª. Dra. Parcilene Fernandes Brito

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2021

Dedico este trabalho, primeiramente, aos meus pais Antonia Maria Fernandes de Souza e Antenor Bispo, que são as matrizes da minha vida e os meus principais pontos de equilíbrio e suporte para a realização desta, que é a maior obra da minha vida até o presente momento. Dedico também, aos meus caros amigos que sempre me apoiaram nos momentos mais difíceis durante esta incrível jornada percorrida até aqui.

AGRADECIMENTOS

Agradeço aos meus pais, em especial à minha mãe, Antonia, por todo o apoio e conselhos entregues a mim, em forma de carinho e compreensão durante esta jornada. Minha mãe, sem dúvidas, é o meu maior pilar, minha maior apoiadora e minha melhor amiga, sem ela, eu jamais chegaria a este ponto da minha vida, o mais alto que uma vez estive. Também agradeço profundamente aos meus demais familiares, que também foram fundamentais para que eu pudesse chegar neste momento tão gratificante em minha vida.

Agradeço aos meus queridos e estimados amigos João Vitor, Samuel Germano, Samuel Jácome, Paulo Sérgio, Lucas Pedro e André Costa. Aquela panelinha querida e unida que sempre podíamos contar quando as coisas apertavam e também sendo a galera da resenha.

Não poderia deixar de agradecer de maneira especial ao meu querido amigo João Vitor Soares Egidio, que me persegue desde o Ensino Médio do saudoso curso de Mecatrônica no IFTO Campus Palmas. João foi uma das maiores graças da minha vida, o amigo que eu precisava ter – o amigo que todos deveriam ter. Juntos entramos na faculdade e, a partir daí formou-se uma grande e longínqua amizade que perdura dentro e fora dos limites do campus da universidade.

Igualmente agradeço ao meu orientador M.e Jackson Gomes de Souza pelas considerações, orientações, ponderações e, claro, pela paciência em indicar as direções para a elaboração e desenvolvimento deste trabalho, trilhando o caminho para a minha formação. Muito obrigado Jackson!

Gostaria de agradecer à minha banca, formada pelos professores M.e Fabiano Fagundes e Dra. Parcilene Fernandes Brito. É importante salientar que suas ponderações na elaboração do projeto deste trabalho foram essenciais para a evolução e amadurecimento do mesmo, e também para mim, que tomei as sugestões inferidas e as adotei em minha vida.

Assim como agradeço ao meu orientador e à banca, gostaria de agradecer também aos professores e excelentes profissionais Madianita Bogo Marioti, Cristina D'Ornellas Filipakis Souza, Fernando Luiz de Oliveira e, principalmente, ao professor Fábio Castro Araújo, um companheiro de jogatinas e também, um amigo que me ajudou e me fortaleceu no decorrer do processo de desenvolvimento deste trabalho.

Por fim, a cada uma das pessoas que me ajudaram e me apoiaram, direta ou indiretamente, no decorrer do curso, sendo este um dos momentos mais marcantes e queridos da minha vida e que isto seja o início de muitas aventuras.

Muito obrigado a todos!

RESUMO

BISPO, André Fernandes. **Automação do Preenchimento de um Grafo de Conhecimento Aplicado a Artigos de um Portal de Notícias Esportivas.** 2020. 91 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Engenharia de Software, Centro Universitário Luterano de Palmas, Palmas/TO, 2020.

O presente trabalho tem por objetivo realizar a automação do preenchimento de um Grafo de Conhecimento. Para tanto, serão apresentados conceitos sobre extração de dados, Grafo de Conhecimento e Processamento de Linguagem Natural. Um Grafo de Conhecimento é uma base de conhecimento que utiliza nós e arestas ou triplas para organizar as informações. Para o preenchimento deste grafo foi escolhido um portal de notícias esportivas como fonte de alimentação, os dados das notícias publicadas neste portal foram extraídos, analisados pelo Processamento de Linguagem Natural que teve por objetivo analisar a notícia, tokenizando e etiquetando cada um dos elementos do texto de acordo com suas classes gramaticais, permitindo a identificação das entidades ou elementos mais importantes, então estes dados processados por um algoritmo desenvolvido na linguagem Python com o objetivo de encontrar as entidades da notícia e informações pertinentes ao trabalho. Por fim, armazenou-se os dados processados no Grafo de Conhecimento de maneira automatizada.

Palavras-chave: Grafo de Conhecimento; Processamento de Linguagem Natural; Automação.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

KG – *Knowledge Graph* (Grafo de Conhecimento)

NLP – *Natural Language Processing* (Processamento de Linguagem Natural)

NER – *Named Entity Recognition* (Reconhecimento de Entidade Nomeada)

EE – *Entity Extraction* (Extração de Entidade)

NE – *Named Entities* (Entidades Nomeadas)

EL – *Entity Linking* (Vinculação de Entidade)

RDF - *Resource Description Framework* (Estrutura de Descrição de Recursos)

DOM - *Document Object Model* (Modelo de Objeto de Documento)

JSON - *JavaScript Object Notation* (Notação de Objeto JavaScript)

TXT - Texto

GE - Globo Esporte

ESPN - *Entertainment and Sports Programming Network* (Rede de Programação de Entretenimento e Esportes)

CQL - *Cypher Query Language* (Linguagem de Consulta em Cifra)

SUMÁRIO

1 INTRODUÇÃO

Em 1968, no filme *2001: Odisseia no espaço,* um dos personagens mais marcantes foi *Hal*, um computador capaz de se comunicar com as pessoas utilizando linguagem natural. Esta ideia de máquinas entenderem o funcionamento da linguagem humana gerou pesquisas e publicações científicas ao longo do tempo. Em 2018 foi desenvolvido o Google Duplex[1] – uma assistente que se comunica tão bem que humanos que se interagiram com ela, não perceberam que se tratava de uma máquina.

A leitura e interpretação de texto pelas máquinas é chamada de Processamento de Linguagem Natural (*Natural Language Processing*, em inglês, ou NLP). Segundo Silva (2008), o NLP, de forma geral, constitui a matriz das tecnologias linguísticas e um dos novos paradigmas da Língua e da Linguística. Assim, o NLP identifica as palavras presentes numa frase e realiza processos de análise que permitem extrair informações e identificar o contexto, dando um significado a cada palavra conforme o domínio. O NLP é responsável por analisar as estruturas de uma linguagem natural, como a semântica, léxica e sintaxe, sendo ela a responsável por analisar e identificar o contexto que as palavras têm em um texto, resolvendo problemas de ambiguidades, por exemplo.

Alinhado ao NLP, está o Grafo de Conhecimento (*Knowledge Graph*, em inglês, ou KG) que segundo Paulheim (2016), (i) descreve as entidades do mundo real e suas relações, organizadas em um grafo, (ii) define também possíveis classes e relações de entidades em um esquema. Ainda sobre o Grafo de Conhecimento, Pan et. al (2017) descrevem que suas características primárias estão no arranjo estrutural da representação de conhecimento, nos processos de coordenação das informações e nos algoritmos de busca.

Para o preenchimento do KG, é necessário que se faça a extração dos dados que podem ser frases, textos ou conceitos, de uma determinada fonte de dados, sejam eles sites, artigos, entre outros. Destes dados, deverá ser feita a extração de entidades que são as principais palavras ou expressões de um texto.

A extração dos dados do trabalho foi obtida por meio de um portal de notícias esportivas que é organizado nas seguintes áreas de conteúdo: página inicial; página ou *blog* de cada área de conteúdo, que mostra as notícias da área em questão; página para cada notícia ou artigo, que apresenta título, data da publicação, conteúdo e notícias relacionadas. Além disso, também foram

[1] Para saber mais sobre o Duplex, acesse: https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html

utilizadas informações extraídas de portais auxiliares que contribuíram para que a taxa de assertividade do trabalho atingisse níveis satisfatórios.

Em vista disso, este trabalho originou-se do problema de automatizar o preenchimento de um Grafo de Conhecimento através do Processamento de Linguagem Natural em um determinado artigo de notícia esportiva. Para realizar este preenchimento, foi necessário extrair um artigo de um portal de notícias, aplicar o Processamento de Linguagem Natural, identificando os pontos desejados e armazená-los no Grafo de Conhecimento. Diante disto, chega-se à hipótese de que é possível automatizar o preenchimento de um grafo de conhecimento no contexto de um portal de notícias por meio da aplicação de técnicas de Processamento de Linguagem Natural e reconhecimento de entidades, extraindo entidades dos artigos de notícias (como pessoa, local e data) e criando relacionamentos (ligações) entre eles para, com isso, gerar o grafo de conhecimento.

Para alcançar o objetivo deste trabalho foi necessário selecionar os portais onde os dados seriam extraídos; foi então, realizado os mapeamentos destes portais, a fim de extrair somente os dados de interesse, eliminando assim, a extração de dados indesejados, como anúncios ou textos que não compreendem a notícia. Após o mapeamento, foi configurado o algoritmo de extração Scrapy com a linguagem de programação Python; foi configurado o algoritmo Linguakit que realizou a análise do artigo extraído e identificou as entidades presentes no mesmo; foi implementado o algoritmo, responsável por aplicar as regras que visam buscar as entidades, padrões e/ou informações pertinentes para o trabalho no artigo analisado, sendo este algoritmo desenvolvido em Python. E, por fim, inseriu-se as entidades e outras informações identificadas no artigo no grafo de conhecimento.

O uso de Grafo de Conhecimento pode fornecer meios mais práticos e competitivos para a tomada de decisões em diversas áreas, sejam acadêmicos ou mercadológicos, visto que esta ferramenta, associada ao Processamento de Linguagem Natural é capaz de identificar o que está sendo dito em uma seção de comentários de um determinado site ou realizar a leitura de um livro, identificar os principais pontos e criar resumos, facilitando a pesquisa de alunos, professores e pesquisadores. Isso já pode ser visto no campo linguístico, onde traduções simultâneas permitem que haja uma comunicação entre povos de diferentes idiomas.

A aplicabilidade dos Grafos de Conhecimento é infindável, sendo utilizados em diversos setores da indústria. Um exemplo disto é o Google, que o utiliza nos resultados de pesquisas feitas pelos usuários, podendo ser também utilizado em bibliotecas, enciclopédias, portais, artigos, revistas, sistemas hospitalares, tribunais de justiça, etc.

O conteúdo do presente trabalho está estruturado da seguinte maneira: na segunda seção foi discutido sobre o Referencial Teórico, onde foi disponibilizado informações sobre os conceitos e trabalhos acadêmicos que embasam este trabalho. Na terceira seção foi abordada a metodologia aplicada no desenvolvimento do trabalho. Na quarta seção foi apresentado os resultados obtidos e discussões em torno destes resultados. Por fim, na quinta seção foram expostos as considerações finais e trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 EXTRAÇÃO

Para realizar a extração dos dados do portal de notícias esportivas (e dos portais auxiliares) que serão processados e armazenados posteriormente, foi utilizado o *framework* Scrapy do Python.

Python é uma linguagem orientada a objetos criada em 1992, por Guido van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda, foi criado originalmente com o foco no trabalho de físicos e engenheiros, mas hoje em dia é utilizado por várias empresas de tecnologia em diversas áreas. Tem uma linguagem com sintaxe clara e sucinta, tornando sua codificação facilmente legível por outros desenvolvedores. De acordo com Borges (2016, p. 14), o Python integra diversas estruturas de alto nível, como listas, dicionários, tuplas, entre outras, bem como um grande número de módulos prontos, além disso, *frameworks* de terceiros podem ser integrados na linguagem.

O Scrapy é um *framework web* utilizado para a raspagem de dados da *web* (*web scraping*). No Scrapy são criados os *spiders* que são programas que percorrem as páginas na *web* e realizam a extração dos dados desejados pelo usuário, auxiliando na extração de dados complexos de forma simples e fácil. De acordo com Kouzis-Loukas (2016, p. 1) o Scrapy permite realizar operações que limpam, formam e enriquecem os dados, sendo possível armazená-los em locais desejados pelo usuário, como banco de dados e com uma baixa exigência de desempenho da máquina que está realizando a tarefa.

2.2 PROCESSAMENTO DE LINGUAGEM NATURAL

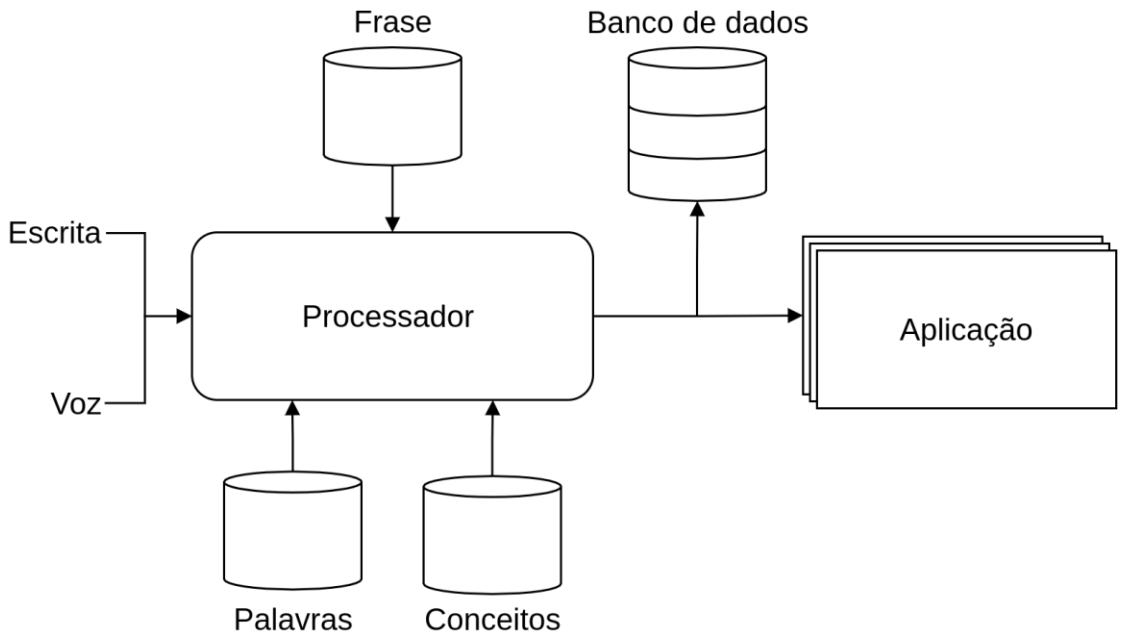
Em virtude das necessidades biológicas humanas, as linguagens naturais foram e são adaptadas à natureza psicológica do homem de forma estrutural e “se não fossem assim adaptadas, não poderiam ser adquiridas" (LYONS, 1991, p. 2). A linguagem natural, após milhares de anos de evolução, entre extinções e surgimentos de novas formas de comunicação envolvendo a fala, tornou-se a maneira mais usual e prática para seres humanos se comunicarem entre si.

Nos dias atuais, existem milhares de maneiras diferentes de comunicação utilizando uma linguagem natural, entre idiomas e dialetos, como o português, inglês, espanhol, entre outros. Após todo esse período de adaptações e reformulações, as línguas continuam evoluindo para adaptar-se ao homem contemporâneo, isso pode ser visto no português que absorveu algumas palavras existentes na língua inglesa e a própria língua inglesa que sofreu influências do francês.

O Processamento de Linguagem Natural é uma disciplina que se utiliza de conhecimentos sobre a linguagem natural humana, bem como sua comunicação, valendo-se destes preceitos para que haja uma comunicação com sistemas operacionais, bem como, uma maneira de facilitar a comunicação entre seres humanos (SANTOS, 2001, p. 229). De acordo com Vieira e Lopes (2010, p. 184), o "Processamento de Linguagem Natural é uma área da Ciência da Computação que estuda o desenvolvimento de programas de computador que analisam, reconhecem e/ou geram textos em linguagens humanas, ou linguagens naturais". Referindo-se à linguística computacional, Vieira e Lima (2001, p. 1) proferem que é "a área de conhecimento que explora as relações entre linguística e informática, tornando possível a construção de sistemas com capacidade de reconhecer e produzir informação apresentada em linguagem natural".

Um exemplo de uso do NLP pode ser visto no *Cloud Speech-to-Text* que, segundo Sanches (2017) é uma ferramenta desenvolvida pela Google para converter voz em texto em tempo real ou de arquivos armazenados. Ainda de acordo com Sanches (2017), utilizar o *Cloud Speech-to-Text* apresenta várias vantagens, entre as quais está o acesso a toda a base de dados presente na ferramenta, permitindo que o trabalho se concentre no desenvolvimento das técnicas pós transcrição do áudio. A Figura 1, mostra uma ideia geral do processo de funcionamento do NLP.

Figura 1: Etapas do Processamento de Linguagem Natural.



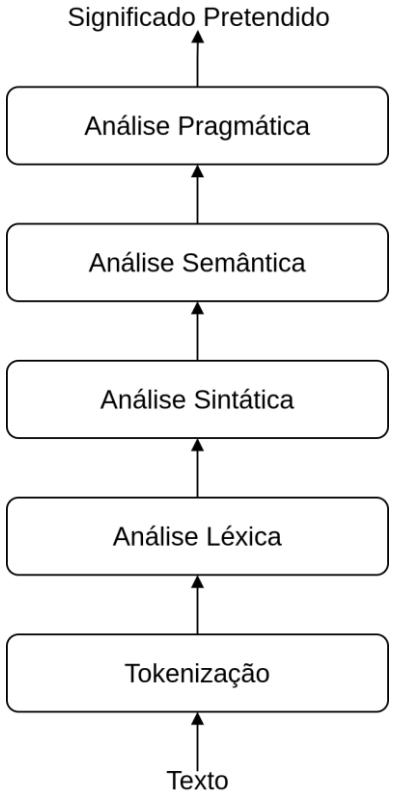
Fonte: https://lbsitbytes2010.wordpress.com/2013/03/27/scientist1-makoto-nagao-roll-no6/ (tradução nossa).

Na figura acima, é observável a entrada de dados por meio da voz ou escrita; o processamento destes dados, que podem ser frases, palavras ou conceitos; o armazenamento da informação processada em um banco de dados; e a sua inserção em uma aplicação.

2.2.1 Etapas do NLP

Para se comunicar de forma eficaz, os falantes dos idiomas seguem certos protocolos que irão organizar as palavras e frases de forma coerente, criando um padrão que permitirá que os ouvintes possam entender o que está sendo dito, além de permitir o aprendizado do idioma por outros povos. A Figura 2, mostra algumas etapas destes protocolos que permitem a coerência da fala e da escrita.

Figura 2: Estágios de análise no NLP.



Fonte: Adaptada de Dale (2010, tradução nossa)

De acordo com Dale (2010), o processo de análise no NLP tende a ser decomposto em cinco estágios principais (Figura 2), onde as sentenças de um texto são primeiro analisadas sintaticamente, fornecendo uma ordem e estrutura que são mais acessíveis a uma análise semântica, seguido por um estágio de análise pragmática. Nos tópicos a seguir, será feita uma descrição de cada uma destas etapas.

2.2.1.1 Tokenização

De forma sucinta, a tokenização é responsável por separar as sentenças em unidades menores, ou seja, fragmentar as frases em palavras. Por exemplo, na frase “Antonia é mãe do

André", a tokenização dividiria a frase em 5 partes, onde cada parte corresponde a uma palavra, ou seja, originaria o seguinte vetor: [Antonia, é, mãe, do, André].

A tokenização pode apresentar dificuldades dependendo da linguagem a ser processada, visto que, segundo Palmer (2010) há uma diferença fundamental entre as linguagens delimitadas por espaço e as não segmentadas.

Nas linguagens delimitadas por espaços, os limites das palavras são indicados pela inserção de espaços em branco entre elas, onde estes espaços determinam o início e fim destas palavras (PALMER, 2010). É possível observar essa regra na maioria dos idiomas europeus, no entanto, apesar de haver uma delimitação natural que facilita a tokenização, ainda há problemas a serem resolvidos. Nos idiomas provenientes do latim, há ambiguidades no que tange ao uso das pontuações (pontos, vírgulas, aspas, hífens, entre outros), visto que um mesmo sinal pode apresentar funções diferentes (PALMER, 2010, p.16). Isso pode ser visto no Exemplo 1.

Exemplo 1: "Prof. Antonia, mãe do André, gastou R$ 39,99 em um restaurante.".

Nesta frase, é possível observar que há dois pontos de interesses. Primeiro, nos dois modos de uso do ponto final: na palavra "Prof.", o ponto final é utilizado para abreviar a palavra "Professora" e no segundo uso, é utilizado para concluir a frase. E segundo, nas vírgulas: nos dois primeiros usos da vírgula, ela é utilizada para determinar o aposto da frase e no terceiro uso é utilizada para separar o número inteiro "39" de sua parte fracionária "99".

Nas linguagens não-segmentadas, como o japonês e o tailandês, por exemplo, as palavras são escritas de forma continuada, sem uma indicação de limites (PALMER, 2010), em outras palavras, diferentemente das linguagens europeias, nestes idiomas as palavras não possuem um espaçamento em branco delimitando cada uma. Palmer (2010, tradução nossa) conclui: "a tokenização de línguas não-segmentadas, portanto, requer informações lexicais e morfológicas adicionais".

2.2.1.2 Análise Léxica

Segundo Coppin (2013), a análise léxica "examina os modos pelos quais palavras se desmembram em componentes e como isso afeta o *status* gramatical delas", em outras palavras, a análise léxica tem por objetivo identificar qual classe a palavra pertence, seja ela um substantivo, artigo, adjetivo, pronome, advérbio, entre outras.

De acordo com o Hippisley (2010), uma palavra pode ser pensada de duas maneiras: como uma *string* em um texto, por exemplo, o verbo ENTREGAR; ou como um objeto mais

abstrato que é o termo principal para um vetor de *strings*, assim, o verbo ENTREGAR compõe o seguinte vetor: [entrega, entregador, entregando, entregue].

As palavras podem ser consideradas como os principais elementos encontrados na composição de um texto, sendo também os elementos mais complexos de serem analisados devido às várias formas que um mesmo verbo pode ser escrito, sendo assim, a análise léxica é o mecanismo responsável por realizar a análise no nível da palavra. Por exemplo, a palavra "infelizmente" pode ser fragmentada em 3 partes: primeiro pelo prefixo de negação "in", depois pelo radical "feliz" e por último pelo sufixo "mente". O prefixo "in" não realiza alteração na classe gramatical do adjetivo "feliz". No entanto, com o sufixo "mente", isso ocorre, pois com a adesão deste complemento à palavra "infeliz", a classe gramatical é alterada de um adjetivo, para advérbio de modo.

2.2.1.3 Análise Sintática

A compreensão do significado de uma frase é um ponto fundamental para entender a mensagem que está sendo passada e, de acordo Dale (2010), uma frase expressa uma ideia ou um pensamento e diz algo sobre algum mundo concreto ou abstrato. Já Culicover (2009, p. 1), expressa que a sintaxe é "o sistema que governa a relação entre forma e significado em uma linguagem". A sintaxe é responsável por combinar as palavras em sentenças por meio de regras gramaticais para determinar como estas palavras serão combinadas (ROSA, 2011), no entanto, as frases não são somente uma sequência de palavras, sendo necessário a análise destas frases, a fim de encontrar uma junção satisfatória.

Segundo Ljunglöf e Wirén (2010, p. 61), a maioria dos formalismos gramaticais[2] são derivados ou podem estar relacionados à gramática livre de contexto. A forma padrão para representar uma gramática livre de contexto, segue o seguinte formalismo: $G = \{\Sigma, N, S, R\}$. Onde Σ e N são conjuntos de símbolos terminais (Ex.: {Antonia, mãe, André}) e não terminais (Ex.: {verbos, adjetivos, substantivos}), respectivamente. $S \in N$ é o símbolo inicial. R é um conjunto finito de regras de produção, sendo $A \rightarrow \alpha$, onde $A \in N$ é um símbolo não terminal e α pode representar tanto os símbolos terminais, quantos os não terminais.

Para qualquer idioma, a maneira como as palavras serão sequenciadas determinará sua ordenação, que influenciará os morfemas e a fonética, isto é, o sequenciamento dos sons no tempo (CULICOVER, 2009, p. 1). Na língua portuguesa, a sintaxe é dividida entre os termos essenciais

[2] Para saber mais, acesse: http://uece.br/eventos/siel2015/anais/trabalhos_completos/150-31743-18012016-203429.pdf

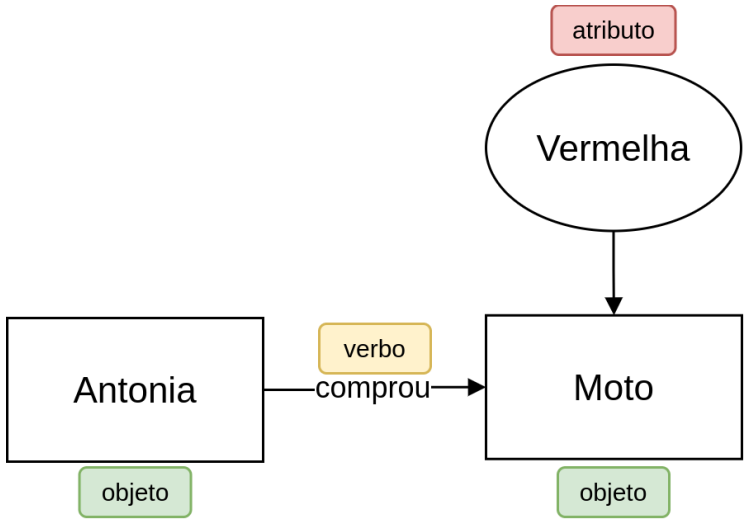
da oração (sujeito e predicado), termos integrantes e os termos acessórios. O conhecimento sobre a sintaxe torna-se essencial para que haja uma comunicação coerente entre os interlocutores, seja na escrita ou na fala.

2.2.1.4 Análise Semântica

A análise semântica refere-se à análise dos significados das palavras, expressões e frases (GODDARD e SCHALLEY, 2010), em outros termos, a análise semântica tem como foco identificar o significado e a interpretação de uma palavra inserida numa frase ou de uma frase inserida num texto, bem como, analisar as mudanças de sentido destas palavras e frases de acordo com o tempo.

De acordo com Coppin (2013, pág. 511), a análise semântica "envolve a elaboração de uma representação dos objetos e ações que uma sentença esteja descrevendo, incluindo detalhes fornecidos por adjetivos, advérbios e preposições". Uma rede semântica pode ser utilizada para a representação dos objetos e isso pode ser observado na frase "Antonia comprou uma moto vermelha", com esta frase a seguinte rede semântica da Figura 3 é criada:

Figura 3: Rede Semântica



Na Figura 3, os objetos representados por retângulos são compostos por "Antonia" e "moto", onde os objetos são interligados por uma seta atribuída com o verbo "comprou". Por fim, o atributo "vermelha" está sendo representado por uma elipse.

2.2.1.5 Análise Pragmática

A compreensão do significado das frases é feita tanto pela semântica, como foi explicitado anteriormente, quanto pela pragmática. Segundo Levison (2007), a "pragmática é o estudo das relações entre língua e contexto que são 'gramaticalizadas' ou codificadas na estrutura de uma língua", isto é, a pragmática busca compreender o significado de uma frase dentro de um contexto dando sentido à sentença. Para exemplificarmos isso, podemos observar as seguintes frases:

1. Antonia aparenta estar com coriza.
2. Antonia está tendo uma crise alérgica.

Ao analisar a primeira frase de maneira individual, não é possível deduzir a causa da coriza que Antonia está tendo, uma vez que, esta informação não foi expressa. Na segunda frase, já é possível ao locutor inferir que a possível causa da coriza possa ser proveniente de uma crise alérgica.

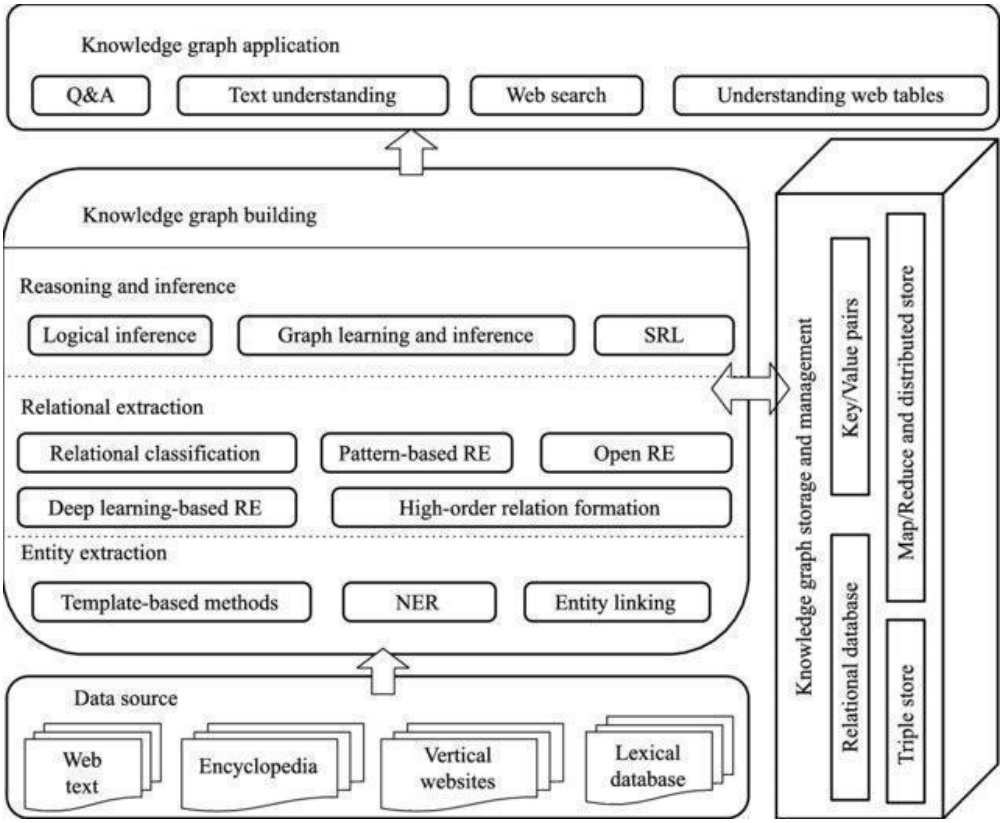
2.3 GRAFO DE CONHECIMENTO

Após a extração e processamento do artigo, é necessário que ele seja armazenado adequadamente de maneira que atenda à demanda exigida. Para este trabalho, foi utilizada uma base de dados dinâmica que permite uma fácil e rápida pesquisa utilizando o Grafo de Conhecimento como base estrutural de como os dados serão organizados. O KG armazena as informações presentes num texto, por meio de entidades e retorna resultados mais precisos, de forma organizada e coerente, para o usuário ou para uma máquina que o esteja acessando.

Segundo Singhal (2012), ex-vice-presidente da Google, o Grafo de Conhecimento seria como um "gráfico" que relaciona entidades do mundo real umas com as outras. Uma outra definição foi expressa por Pan et. al (2017), ao informar que as características primárias dos Grafos de Conhecimento estão no arranjo estrutural da representação de conhecimento, nos processos de coordenação das informações e nos algoritmos de busca.

Um exemplo da estrutura de um Grafo de Conhecimento pode ser visto na Figura 4:

Figura 4: Estrutura de um Grafo de Conhecimento.



Fonte: Adaptado de Yan et al. (2016)

Na Figura 4, é visível todo processo de um Grafo de conhecimento, que se inicia pela fonte de dados (*Data source*), em seguida se direciona para o processamento desses dados (*Knowledge graph building*), nesta etapa é realizada a extração de entidade (*Entity extraction*), a extração de relação (*Relacional extraction*) e o raciocínio e inferência (*Reasoning and inference*). À direita é mostrado os tipos de banco de dados (*Knowledge graph storage and management*) que poderão ser utilizados para a armazenagem dos dados processados, e acima, há a aplicação em si do KG (*Knowledge graph application*).

Os dados que servirão de alimentação para o KG podem vir de várias fontes, dentre as quais, destaca-se os textos da *web*, enciclopédias, sites verticais[3] e banco de dados léxicos[4]. Para que estes dados sejam inseridos no KG, é necessário que antes sejam extraídos de seus locais de hospedagem, esta extração pode ser feita de diversas maneiras, sendo o uso de linguagens de programação a escolhida para este trabalho.

[3] Um site vertical é um site especializado em determinado setor, por exemplo: saúde, aviação, segurança, etc.
[4] Para saber mais, acesse: https://www.teses.usp.br/teses/disponiveis/55/55134/tde-17062015-113227/publico/JulianaGalvaniGreghi_ME.pdf

Após a extração dos dados, deve ser feita a extração das entidades do texto, estas entidades podem ser entendidas como sendo as palavras mais importantes presentes num texto, por exemplo: “André viajou para Fernando de Noronha no verão”. Nesta frase, pode-se identificar que “André”, “Fernando de Noronha” e “verão” são os pontos chaves que garantem a contextualização da frase, sendo estes pontos, as entidades.

Seguindo o caminho proposto pela Figura 3, após a extração de entidades é realizada a extração de relações, nesta etapa, o objetivo é reunir fatos sobre entidades com alta precisão e memória, focando no problema de extração de relações binárias (YAN et al., 2016). Sendo essa relação binária uma tripla (sujeito, predicado, objeto), onde denota-se uma relação semântica entre o sujeito e o objeto por meio do predicado. Aproveitando o exemplo anterior, a tripla seria composta por “André”, “viajou” e “Fernando de Noronha”.

Como os dados extraídos da *web* contêm ruídos, as entidades e relações tendem a ser incompletas e sujeitas a erros. Como resultado, torna-se necessário a redução destes conflitos construídos automaticamente no KG (YAN el al., 2016). Para isto, deve ser realizado o raciocínio e inferência no KG, que é responsável por melhorar a convergência gerando novas instâncias de relação no Grafo de Conhecimento.

Além disso, é estritamente necessário que haja um repositório adequado para o armazenamento dos dados do KG. Segundo Date (2013), “um sistema de banco de dados é basicamente um sistema computadorizado de manutenção de registros”. Ao invés de ter um modelo unificado para todos os bancos de dados gráficos, cada banco de dados gráfico ou cada tipo de banco de dados gráfico é projetado para tipos específicos de tarefas (YAN et al., 2016). Nas seções seguintes será feito um melhor detalhamento sobre o tipo de banco de dados que foi utilizado para o presente trabalho.

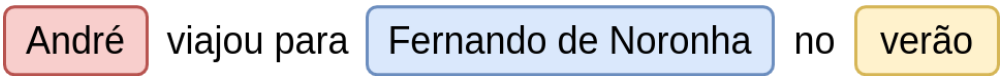
Por fim, de acordo com Pan et al. (2017), conforme citado por Lopes (2020), os Grafos de Conhecimento difundiram-se na academia e na indústria por serem uma das ferramentas mais eficazes em abordagens de integração. Ademais, o Grafo de Conhecimento concentra entidades e conceitos fragmentados, que podem ser conectados para a formação de um conjunto completo e estruturado para um repositório de conhecimento, facilitando seu gerenciamento, recuperação, uso e compreensão das informações contidas (YAN et al., 2016).

2.3.1 Extração de Entidade (EE)

O *Named Entity Recognition* (NER, Reconhecimento de Entidade Nomeada, em português) pode ser visto como a principal tarefa da Extração de Entidade, pois tem como objetivo localizar e classificar as entidades presentes em um texto não estruturado em

determinadas categorias, como pessoas, organizações, locais e, incluindo também expressões temporais, como moedas, data, hora, porcentagem, etc. Para Yan et al. (2016, p. 3), utilizando o Processamento de Linguagem Natural, o NER tem o papel de identificar e classificar determinados tipos de elementos de informação no texto, chamados de entidades nomeadas (NE). A Figura 5 mostra uma representação de seu funcionamento:

Figura 5: Funcionamento do NER.



Nas palavras destacadas estão as principais entidades que o NER classificaria. Em "**André**", a provável classificação seria Pessoa, visto que esta palavra é recorrentemente relacionada a este tipo de entidade. Em "**Fernando de Noronha**", por sua vez, a entidade seria Lugar, visto que este nome está mais relacionado ao nome de uma cidade do que de uma pessoa. Por fim, a classificação de "**verão**" seria Clima.

O NER é uma tarefa complexa, visto que possui muitos desafios a serem resolvidos. Um destes desafios está no processo de identificação das entidades nomeadas. Por exemplo, "Ilha de Fernando de Noronha", neste exemplo, pode ser identificada uma única NE: "Ilha de Fernando de Noronha" ou duas NEs: "Ilha de Fernando de Noronha" e "Fernando de Noronha". De acordo com Amaral (2017, p. 20) "A etapa de classificação é ainda mais complexa que a etapa de identificação, devido à ambiguidade das palavras, ou seja, à mesma NE pode ser atribuída a mais de uma classe, dependendo do contexto" (OLIVEIRA, 2013). No exemplo anterior, "Fernando de Noronha" é classificado como local, enquanto na frase "(...) o português Fernando de Noronha[5] veio ao Brasil explorar nossas terras", "Fernando de Noronha" é classificado como Pessoa.

Um dos sistemas NER atuais mais populares é baseado em técnicas de *Machine Learning* (ML, Aprendizado de Máquina, em português). Segundo Yan et al. (2016, p. 3), em relação às palavras em um texto, o NER tem por objetivo treinar modelos preditivos para classificá-los em categorias predefinidas que podem ser classes ou *tags* mencionadas anteriormente.

Uma palavra pode conter diferentes significados dependendo do contexto que está inserida, por exemplo, a palavra "maçã" pode se referir tanto à fruta, quanto à empresa, isto torna a identificação do contexto em que está inserida, um elemento crucial para a classificação da

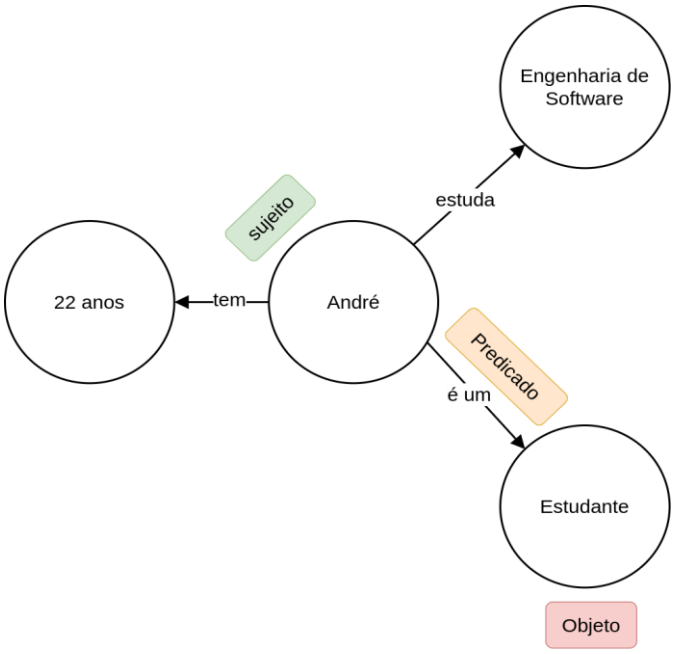
[5] Para saber mais sobre Fernando de Noronha ou, Fernão de Noronha ou, mais precisamente, Fernão de Loronha, acesse: https://pt.wikipedia.org/wiki/Fern%C3%A3o_de_Noronha

palavra na entidade adequada no KG. Para esta tarefa, há o *Entity Linking* (EL, Vinculação de Entidade em português) que tem por objetivo vincular as entidades de um texto à sua representação no KG, sendo uma tarefa essencial para vincular (*linking*) as informações textuais e estruturais (YAN et al., 2016). Para a vinculação das entidades, é estritamente necessária a aplicação do NER no texto, uma vez que as entidades devem ser identificadas, para então, vinculá-las no KG.

2.3.2 Armazenamento e Gerenciamento de Grafos de Conhecimento

O *Resource Description Framework*[6] (RDF) é um padrão criado pela *World Wide Web Consortium*[7] (W3C), que permite a representação de informação ou recursos *web* por meio de triplas, que podem ser organizadas como grafos direcionados (TAVARES et al., 2015). Um padrão RDF consiste em um conjunto de triplas (sujeito, predicado e objeto) e pode ser representado como um grafo, onde os vértices representam os sujeitos e objetos e as arestas correspondem aos predicados que são os rótulos que descrevem a relação do vértice do sujeito com o vértice do objeto. Um exemplo de repositório RDF é o DBpedia[8]. A Figura 6, demonstra um exemplo de três triplas RDF, compostas por um sujeito, três objetos e três predicados.

Figura 6: Tripla RDF.



[6] https://www.w3.org/RDF/

[7] Segundo o W3C (2020), "o *World Wide Web Consortium* (W3C) é uma comunidade internacional, onde as organizações, uma equipe em tempo integral e o público trabalham juntos para desenvolver padrões da web". Para saber mais, acesse: https://www.w3.org/Consortium/

[8] https://wiki.dbpedia.org/dbpedia-wiki

Na tripla, o vértice "**André**" representa o sujeito e ao seu redor estão os objetos "**22 anos**", "**Engenharia de Software**" e "**Estudante**" que descrevem ou indicam características deste sujeito. Os objetos e o sujeito estão interligados pelos predicados "**tem**", "**estuda**" e "**é um**" que indicam o tipo de relação entre eles.

3 METODOLOGIA

Nesta seção serão descritos os materiais e métodos que foram utilizados e como foi o passo a passo para realização do trabalho proposto, de automação do preenchimento de um grafo de conhecimento aplicado a artigos de um portal de notícias.

3.1 MATERIAIS

Para o desenvolvimento deste trabalho, foram utilizados os seguintes materiais: Python, Linguakit, Neo4j e Visual Studio Code.

O Python, segundo Guido van Rossum (1996), criador da linguagem, é uma linguagem de programação orientada a objetos desenvolvida em 1992, com enfoque na legibilidade, possuindo apenas uma possibilidade de endentação e reduzindo ao máximo as maneiras de codificação de um código específico.

O Linguakit, de acordo com o site oficial (2020), é capaz de explorar, analisar e obter informações de textos e documentos escritos, contendo entre outras ferramentas linguísticas, módulos de conjugação e tradução linguística; identificador morfossintático e analisador sintático; analisador de sentimentos e extrator de palavras-chaves.

O Neo4j, de acordo com o site oficial (2020), é um banco de dados desenvolvido para analisar os dados e seus relacionamentos, conectando os dados à medida que são armazenados, permitindo diversas formas de consultas de maneira rápida e prática. O banco de dados do Neo4j opera com o conceito de grafo nativo, possuindo uma estrutura flexível definida por relacionamento armazenados entre registros de dados. Cada vértice do grafo armazena os dados das entidades, aos quais estão ligadas através de arestas.

O Visual Studio Code, segundo o site oficial (2021), é um editor de código-fonte leve e poderoso. Sendo uma ferramenta multiplataforma, estando disponível para Windows, macOS e Linux. O Visual Studio Code vem com suporte integrado para as linguagens JavaScript, TypeScript e Node.js e tem um rico ecossistema de extensões para outras linguagens (como C++, C#, Java, Python, PHP, Go) e tempos de execução (como .NET e Unity).

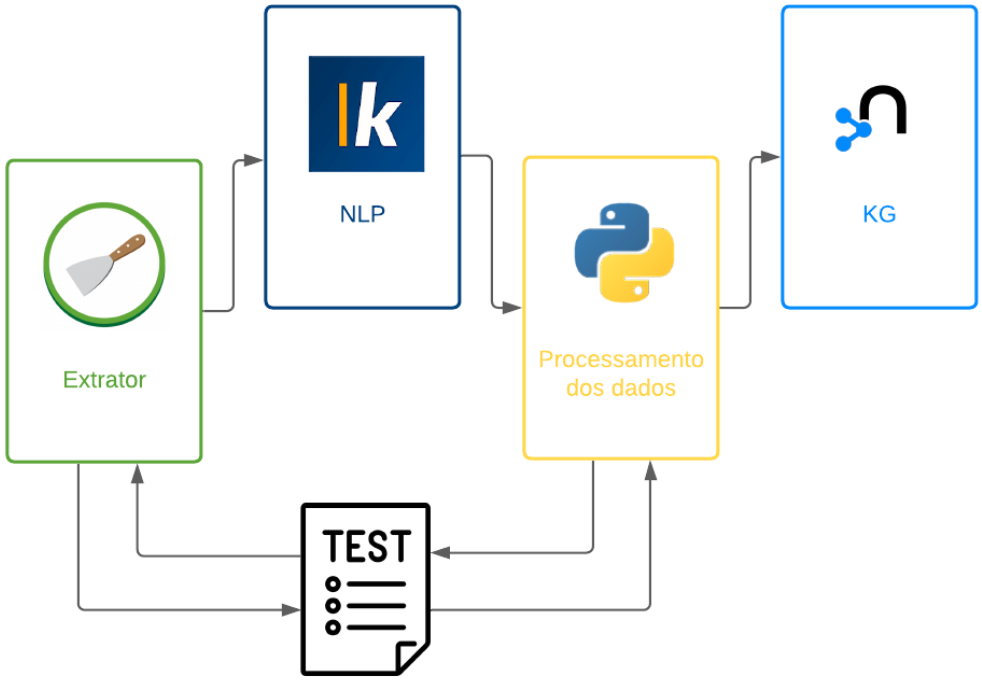
3.2 MÉTODOS

Para alcançar o objetivo deste trabalho, foi necessário organizar todo o fluxo de trabalho, para isto, foi utilizado o Kanban. Segundo Anderson (2010), o Kanban baseia-se nos seguintes pilares: foco na qualidade, redução do trabalho em progresso, permitindo entregas frequentes; redução da variação do fluxo de processos; priorização de demandas. A delineação do trabalho seguirá de acordo com o infográfico da Figura 7:

Figura 7: Infográfico do processo de desenvolvimento do trabalho.

A contextualização é um ponto delicado para o desenvolvimento do trabalho, visto que ela permite um direcionamento e embasamento técnico e teórico para o trabalho como um todo. Para garantir esse embasamento, foi necessária uma fundamentação em artigos e livros que testaram os conceitos e ferramentas que foram utilizados. Livros, dissertações (mestrado ou doutorado) e artigos científicos serviram de base para a elaboração da contextualização do trabalho. Feito a contextualização do trabalho, inicia-se o desenvolvimento das funcionalidades, como apresentado na Figura 8.

Figura 8: Fluxo da Metodologia.



Na fase de implementação, foi realizado o desenvolvimento do trabalho. Primeiramente, foram codificados os extratores dos dados dos portais. Para realizar estas extrações, foi utilizado

o *framework* Scrapy do Python, visto que possui inúmeras funcionalidades que permitem a extração adequada dos dados desejados, armazenando cada uma das extrações em arquivos distintos.

Em seguida, é realizada a configuração do Linguakit. No Linguakit, foi feita a seleção dos módulos que permitiram o desenvolvimento do trabalho, sendo feita a análise do artigo extraído na etapa anterior, onde nesta análise foi feito o Processamento de Linguagem Natural, identificando e nomeando as entidades presentes no texto.

Após a análise, foi realizado o processamento das informações extraídas pelo Linguakit, onde neste processamento, são identificadas as entidades e informações complementares no artigo, sendo organizadas para que possam ser inseridas no Grafo de Conhecimento.

Após o processamento dos dados, foi utilizado um algoritmo disponibilizado pela documentação do Neo4j, para realizar o armazenamento destes dados de maneira remota no Neo4j que utiliza o KG como estrutura para a disposição dos dados.

Nos testes, foi averiguado se o trabalho seguiu de acordo com o objetivo estabelecido, com estes testes sendo aplicados durante e após o desenvolvimento das funcionalidades, permitindo adequações e correções no trabalho que garantiram a qualidade dos resultados esperados.

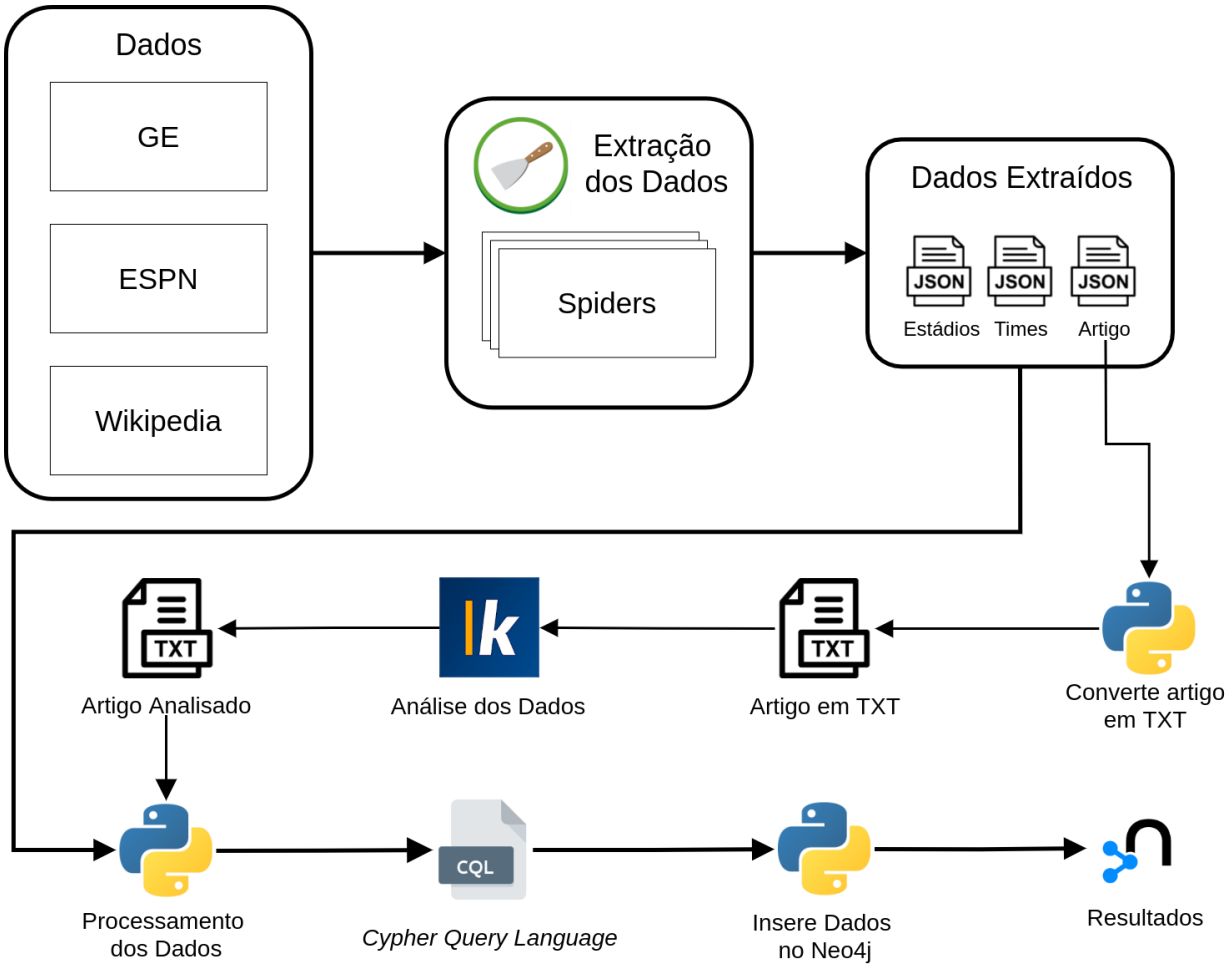
4 RESULTADOS E DISCUSSÃO

Nesta seção serão descritos os resultados e como se procedeu a execução de cada etapa deste trabalho que busca automatizar do preenchimento de um grafo de conhecimento aplicado a artigos de um portal de notícias.

4.1 VISÃO GERAL

Nesta seção é apresentada a visão geral da ferramenta produzida neste trabalho com o detalhamento dos passos seguidos nas próximas seções. A Figura 9 apresenta a arquitetura da ferramenta[9] construída neste trabalho.

Figura 9: Arquitetura da ferramenta.



Os tópicos abaixo descrevem as etapas mostradas na Figura 9:

[9] A ferramenta está disponível em: https://github.com/Andrefer1/tcc

- **Dados:** nesta seção são apresentadas as origens dos dados utilizados no trabalho, sendo o portal do Globo Esporte[10] (GE) a principal fonte destes dados, visto que corresponde ao lugar onde os artigos serão extraídos. Outra fonte corresponde ao portal da *Entertainment and Sports Programming Network*[11] (ESPN), onde são extraídas as listas dos times de futebol. A última fonte corresponde ao portal do Wikipédia[12] onde são extraídas informações sobre os estádios de futebol.
- **Extração dos Dados:** nesta etapa do trabalho é feita a extração dos dados das fontes apresentadas anteriormente, para esta extração é utilizado a biblioteca Scrapy do Python que utiliza o programa *spider* para percorrer a DOM dos portais e realizar a extração dos dados desejados.
- **Dados Extraídos:** ao fim da extração, os dados são armazenados em seus respectivos arquivos *JavaScript Object Notation* (JSON).
- **Converte artigo em TXT:** o Linguakit apresenta limitações quanto às extensões de arquivos suportados para a análise dos dados, por conta disto, os dados do artigo que estão armazenados no arquivo JSON são copiados para um arquivo de texto do tipo TXT.
- **Análise dos Dados:** sendo uma das principais etapas do trabalho, a análise dos dados tem por objetivo identificar e tokenizar os elementos presentes no texto, etiquetando cada um dos *tokens* de acordo com suas classes gramaticais no contexto em que estão inseridos, por fim, armazenando o resultado desta análise em outro arquivo TXT.
- **Processamento dos dados:** sendo o ponto central do trabalho, o processamento dos dados percorre o arquivo com os dados do artigo tokenizados e etiquetados, em busca de entidades e padrões nas etiquetas que permitam a obtenção de informações pertinentes para o preenchimento do grafo de conhecimento. Além disso, o processamento dos dados utiliza-se dos arquivos JSON contendo os dados sobre os times de futebol e estádios com o intuito de aumentar a precisão dos resultados.
- ***Cypher Query Language*:** ao fim do processamento dos dados, é gerado um arquivo contendo as *queries* que serão processadas pelo Neo4j, onde uma *query* pode ser entendida como uma consulta, solicitação ou requisição para o banco de dados (LONGEN, 2021).
- **Insere Dados no Neo4j:** o processamento das *queries* se dará por meio do algoritmo disponibilizado pela documentação do Neo4j. Este algoritmo extrai os dados do arquivo contendo as *queries*, as processa e cria o grafo no Neo4j.

- **Resultados:** nesta etapa são mostrados os resultados obtidos pela execução de todo o processo da ferramenta.

4.2 MAPEAMENTO

O primeiro passo para o desenvolvimento deste trabalho foi o mapeamento estrutural do portal de notícias. Este mapeamento tornou-se necessário devido à presença de conteúdos que não compreendem o artigo presente na página ou aos dados desejados para a extração, como anúncios, *links*, *pop-ups*, entre outros que precisam ser excluídos no momento da extração para evitar a poluição dos dados.

Inicialmente, foi feito o mapeamento do portal de notícias onde estão os artigos a serem extraídos, sendo o GE, o portal escolhido para esta extração, além disso, somente os campeonatos brasileiros masculino série A e B do profissional serão analisados. Neste portal, as notícias são organizadas em áreas e/ou *blogs* dos times, onde são dispostas todas as notícias que envolvem o time.

Dito isto, para a extração dos dados, primeiramente foi feita a identificação dos elementos (*tags*) presentes no *Document Object Model*[13] (DOM), que segue um modelo hierárquico entre os elementos (árvore), para em seguida selecionar estes mesmos elementos cujos valores serão extraídos.

De acordo com Hégaret et al. (2004), organizado no modelo de uma árvore de elementos, o DOM define a estrutura lógica dos documentos HTML e XML e a maneira como um documento é acessado e manipulado. Com o DOM, é possível construir documentos, navegar por sua estrutura, adicionar, modificar ou excluir *tags* e/ou valores. A Figura 10 mostra o exemplo de um DOM.

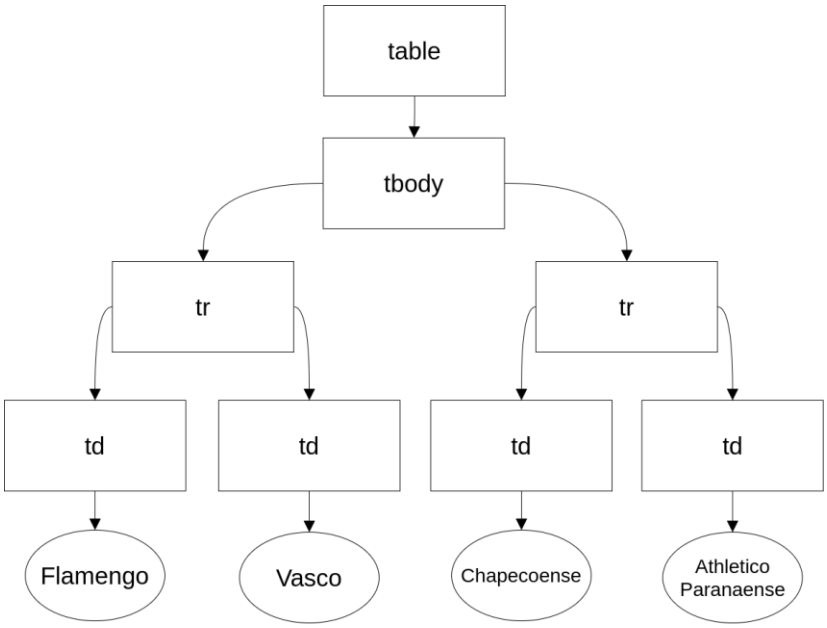
[10] Globo Esporte está disponível em: https://ge.globo.com/
[11] ESPN está disponível em: https://www.espn.com.br/
[12] Wikipédia está disponível em: https://pt.wikipedia.org/wiki/Wikip%C3%A9dia:P%C3%A1gina_principal
[13] Para saber mais, acesse: https://dom.spec.whatwg.org/

Figura 10: DOM de uma tabela.

```
<table>
  <tbody>
    <tr>
      <td> Flamengo </td>
      <td> Vasco </td>
    </tr>
    <tr>
      <td> Chapecoense </td>
      <td> Athletico Paranaense </td>
    </tr>
  </tbody>
</table>
```

Na figura acima, é possível observar uma tabela composta por quatro *tags*, cujos valores destas *tags* são compostos por nomes de times de futebol. Representando o DOM da Figura 10 graficamente, tem-se o seguinte resultado (Figura 11):

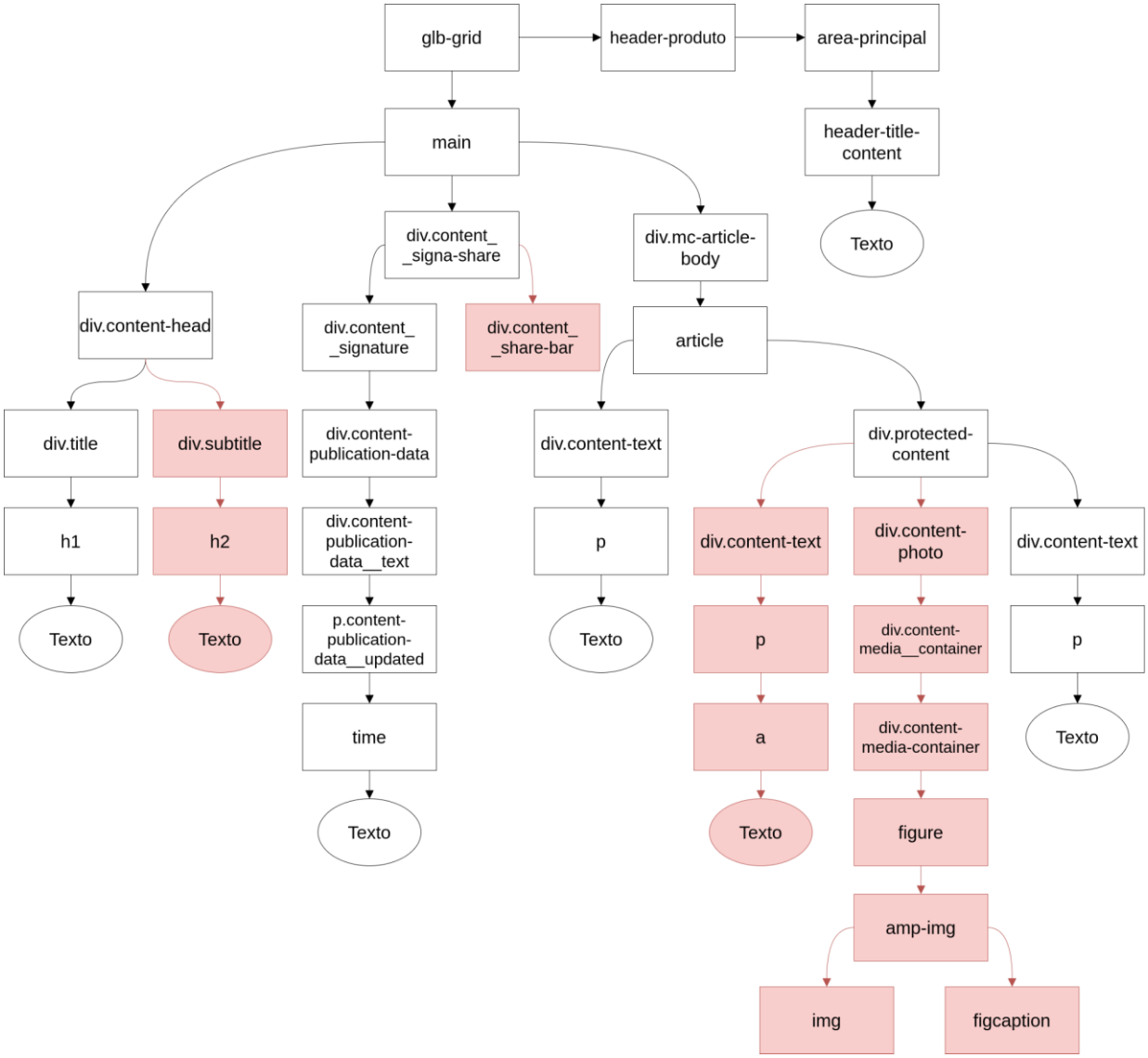
Figura 11: Árvore gráfica de um DOM.



A Figura 11 demonstra a árvore do DOM da tabela da Figura 10. Esta árvore tem como elemento raiz a *tag* <*table/*>, que possui a *tag* <*tbody/*> como filha. Esta *tag* <*tbody/*> possui duas *tags* filhas <*tr/*>, que possuem cada uma duas *tags* filhas <*td/*> que possuem, por fim, os valores contendo os nomes dos times.

No mapeamento do portal foram identificadas as principais *tags* que compõem sua estrutura. É importante salientar que o DOM do portal alterna a cada página, uma vez que uma notícia pode apresentar vídeos que complementam o artigo, enquanto outra apresenta imagens com a mesma finalidade, apesar dessas diferenças, o portal possui *tags* fixas, ou seja, não se alteram conforme a notícia. De modo geral, o DOM do portal segue a estrutura apresentada na Figura 12.

Figura 12: DOM do portal do GE.



Durante o mapeamento do DOM, foram desconsideradas todas as *tags* que não integravam a *tag* <*main/*>, que é onde se encontra os alvos da extração, excetuando-se a *tag* <*header-title-content/*>, onde se encontra o nome do *blog*. Na *tag* <*main/*>, inicialmente foram identificadas as *tags* que compreendiam ao título e subtítulo do texto (<*div.title/*> e <*div.subtitle/*>), assim como suas respectivas *tags* filhas (<*h1/*>, <*h2/*>) que possuem como valor os textos alvos para a extração. Em seguida foi mapeada a *tag* <*div.content__signa-share/*> que corresponde ao autor do artigo, também apresenta informações sobre a publicação, bem como botões de redes sociais que visam facilitar compartilhamento do artigo. Após o reconhecimento das *tags* do cabeçalho do artigo, foi feito o mapeamento dos elementos que compreendem ao texto, estando todos elementos inclusos na *tag* <*article/*>.

O texto é dividido em duas partes: na primeira há o parágrafo introdutório que está contido na *tag* <*div.content-text/*> que contém a *tag* <*p/*>, onde está incluso o valor desejado para a extração. Paralelo ao texto introdutório, na *tag* <*div.protected-content/*>, encontra-se o restante do artigo, porém também estão inclusos *links*, imagens, vídeos, entre outros dados.

Com o mapeamento concluído, foi feita a eliminação dos dados que não seriam úteis para os próximos passos do desenvolvimento deste trabalho (em vermelho), entre estas exclusões, estão o subtítulo, que apresenta informações redundantes, o autor do artigo, botões, *links* e as mídias (imagens e vídeos), restando apenas o nome do *blog*, o título, data da publicação e o texto do artigo.

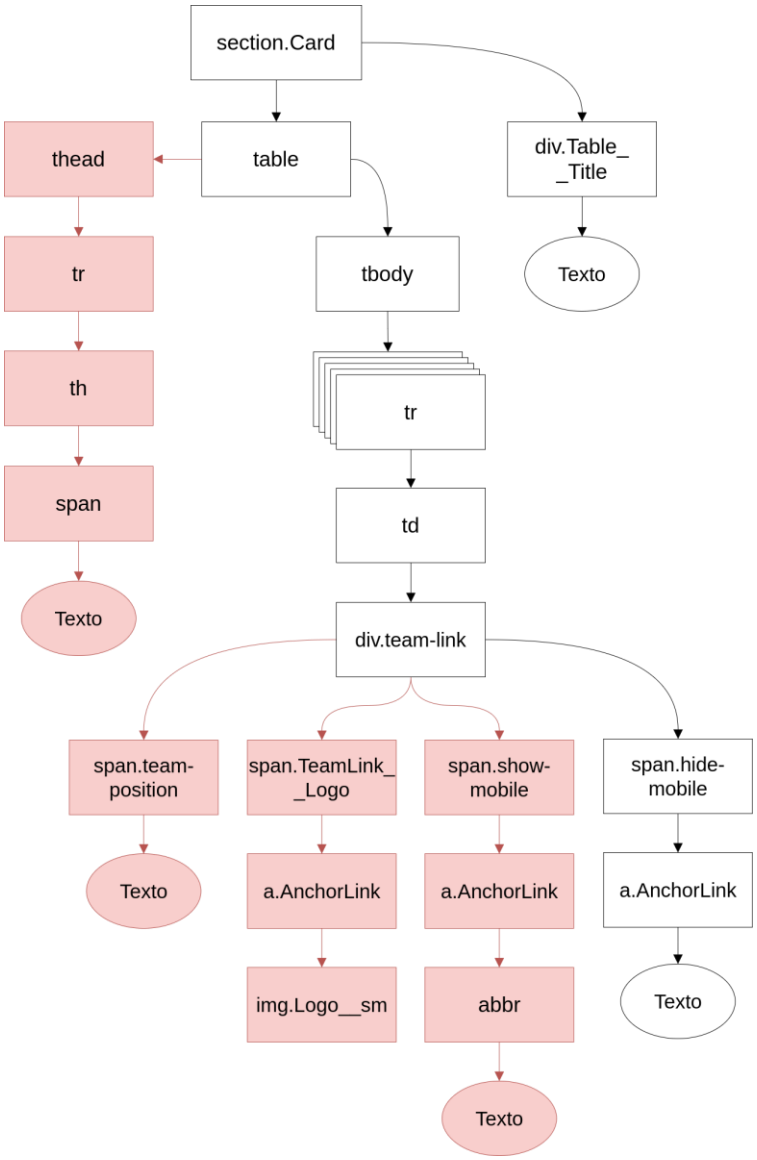
Além da extração do artigo, foi necessário extrair os nomes dos times e dos estádios de futebol. Estas extrações tornaram-se necessárias, devido à problemática que envolve redundâncias nos nomes das entidades que o contexto esportivo dos artigos extraídos pode trazer. Isso pode ser observado no Exemplo 2.

Exemplo 2: "O **Flamengo** enfrentará o **São Paulo**, no **Raulino de Oliveira**, em **Volta Redonda**.".

Como pode ser observado no exemplo acima, é possível identificar quatro entidades que apresentam problemas de redundâncias. A começar pelo **Flamengo**, que se refere tanto ao clube de futebol, quanto ao bairro Flamengo no Rio de Janeiro. **São Paulo** é o mais pertinente entre eles, uma vez que pode representar 4 conhecidos significados: clube de futebol, cidade, estado e pessoa. **Raulino de Oliveira** no contexto apresentado refere-se ao nome de um estádio de futebol, porém, fora deste ambiente refere-se ao nome de uma pessoa. Enquanto **Volta Redonda** refere-se tanto à cidade, quanto ao clube de futebol.

A extração dos nomes dos times de futebol, tiveram como alvo os times das séries A e B do campeonato brasileiro. Esta extração também teve como objetivo, trazer um suporte para uma identificação mais assertiva das entidades presentes no texto. Para satisfazer esta necessidade, foi escolhido o portal de notícias esportivas da ESPN. Assim como no primeiro mapeamento, foi feita a identificação das *tags* cujos valores serão extraídos.

Figura 13: DOM do portal da ESPN.

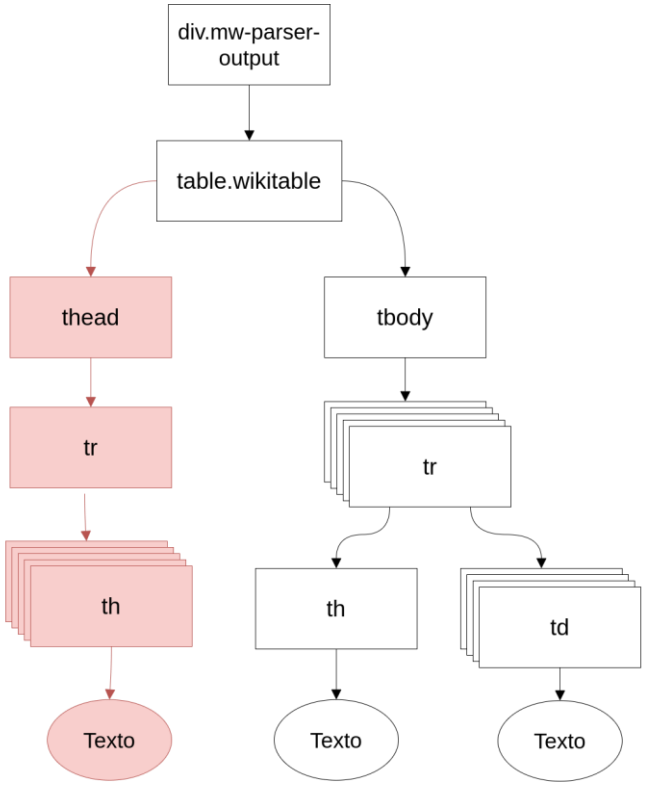


A Figura 13 representa a estrutura geral da página da ESPN (desconsiderando-se os anúncios, *links* e botões), onde os nomes dos times foram extraídos. A *tag* <*section.Card*/> engloba os valores alvos da extração, sendo composta pelas *tags* <*div.Table__Title*/> que corresponde ao nome da série na qual os clubes pertencem e pela *tag* <*table*/> que corresponde

à tabela na qual os nomes dos times estão inseridos. Nesta *tag*, se encontra a *tag* <*thead*/> que corresponde ao título da tabela, seguida da <*tbody*/>, corpo da tabela, que engloba, além dos nomes dos clubes, informações como a posição em que se encontra na tabela (<*span.team-position*/>), a logo (<*img.Logo__sm*/>) e a abreviação do nome dos clubes (<*abbr*/>). Identificadas *tags* desejadas para a extração, descartou-se as excedentes (em vermelho).

Por fim, foi feita a extração dos nomes dos estádios de futebol, seus apelidos, cidades e estados que sediam. Para a extração das informações sobre os estádios, foi escolhido o portal do Wikipédia, por conter uma informação detalhada e de maneira que permitisse a extração. Assim como nos mapeamentos anteriores, foi feita a identificação das *tags* da página que continham os valores desejados para a extração (Figura 14).

Figura 14: DOM do portal do Wikipédia.



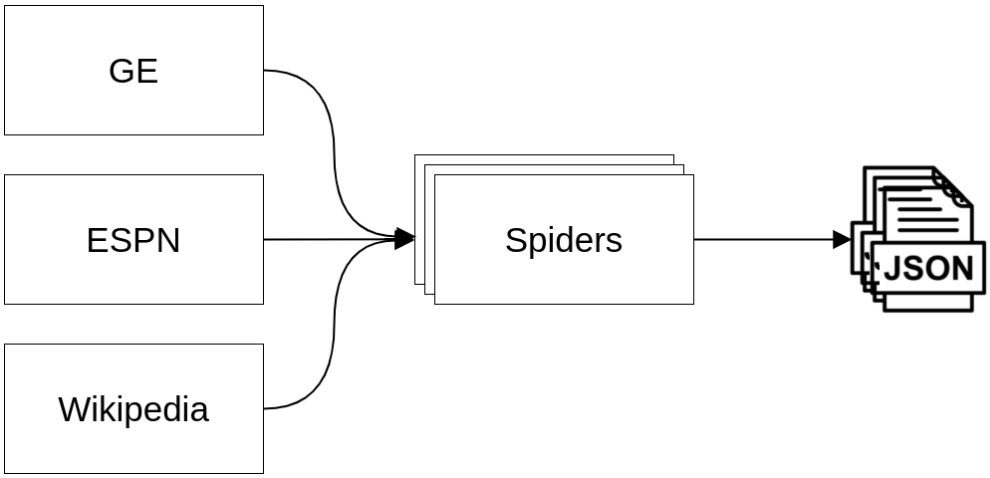
Inicialmente foi identificada a *tag* que correspondia à tabela contendo os dados desejados (<*table.wikitable*/>). Foi feito o mapeamento desta tabela, onde foram identificadas as *tags* que correspondiam ao cabeçalho da mesma: <*thead*/> que contém a linha <*tr*/>, que possui cinco *tags* filhas <*th*/> que correspondem às colunas. Paralelo à *tag* do cabeçalho, está a *tag* correspondente ao corpo da tabela <tbody/>. Nesta *tag* encontram-se as linhas <*tr*/> da tabela, onde cada uma destas contém uma *tag* <*th*/> que possui um valor decrescente da posição do

estádio corrente em relação à sua capacidade de espectadores e, quatro *tags* <*td/*> que possuem, respectivamente, o apelido do estádio, a cidade que sedia, a unidade federativa e a capacidade de espectadores que suporta.

4.3 CONFIGURAÇÃO DO SCRAPY

Após o mapeamento dos sites, iniciou-se o desenvolvimento do extrator dos dados selecionados, apoiado pelos testes que garantiram o retorno dos dados desejados. Para esta extração foi escolhido o Scrapy, um *framework* de extração de dados para *websites* que utiliza a linguagem Python para seu funcionamento. Para esta extração, o Scrapy possui um programa chamado *spider* que tem a função de navegar por meio das páginas na internet e extrair as informações pré-selecionadas. A Figura 15 apresenta a arquitetura da extração dos dados.

Figura 15: Arquitetura da extração.



De acordo com a Figura 15, a arquitetura baseia-se em três fontes de dados distintas, onde cada uma possui um extrator *spider* próprio para realizar a extração. Feita extração dos dados, o *spider* gera os respectivos arquivos JSON de cada raspagem, contendo os dados selecionados no mapeamento das páginas.

Nas seções seguintes será demonstrado o funcionamento do Scrapy nas extrações dos dados.

4.3.1 Extração do artigo

Cabe salientar que o processo utilizado na extração dos dados do artigo é semelhante às extrações que serão citadas posteriormente, diferenciando apenas o *link* para a página, as *tags*

com os valores alvos e as funções para o tratamento dos dados. A Figura 16 apresenta o algoritmo Scrapy (*spider*) que realizou a extração dos dados do portal.

Figura 16: Algoritmo Scrapy para a extração dos dados do portal do GE.

```
import scrapy

class Noticia(scrapy.Spider):
    name = 'noticia'

    start_urls = [
        'https://ge.globo.com/futebol/times/atletico-mg/noticia/atletico-mg-x-chapecoense-com-covid-19-nacho-
         fernandez-nathan-e-outros-tres-jogadores-viram-baixas-de-ultima-hora-no-galo.ghtml'
     ]

    open('./ArquivosJSON/1 - noticiaBruta.json', 'w').close()

    def parse(self, response):
        for noticia in response.xpath('//body'):
            yield {
                'blog': noticia.xpath('//h1[@class="header-title"]//a//text()').extract_first(),

                'data': self.corrigir_data(
                        noticia.xpath('//div[@class="content__signa share"]//time//text()'
                 ).extract_first()),

                'titulo': noticia.xpath('//h1[@class="content-head__title"]//text()').extract_first(),

                'conteudo': self.corrigir_conteudo(
                        noticia.xpath('//p[@class="content-text__container "]//text()'
                 ).extract()),
            }
```

Inicialmente, após criar o arquivo Python, foi importado o Scrapy para que fosse possível utilizar suas propriedades, em seguida foi criada a classe “Noticia” que recebe o módulo *scrapy.Spider* como parâmetro, este módulo determinará como os dados do portal serão extraídos. A variável *starts_url* recebe como valor(es) o(s) *link*(s) da(s) página(s) alvo(s) da extração. Em seguida, é realizada a sobrescrita, com um conteúdo vazio, do arquivo JSON onde os dados serão armazenados, para isto, o arquivo foi aberto em modo de escrita (“w”) e fechado logo em seguida, caso o arquivo JSON ainda não exista, será criado no mesmo diretório inserido no primeiro parâmetro. E por fim, na função *parse*, foram inseridas as *tags* do portal cujos valores foram extraídos, e, após a execução do algoritmo, foi retornado um arquivo no formato JSON (criado anteriormente) com todo o conteúdo selecionado. A Figura 17 mostra o exemplo de uma página na qual os dados do artigo foram extraídos:

Figura 17: Blog do Flamengo no portal do GE.

Como é possível observar, a página possui uma estrutura organizacional padrão: iniciando-se com o cabeçalho que engloba os *links* e/ou botões que redirecionam o usuário para outras áreas do portal, nome do *blog* em questão, título do artigo e seu conteúdo (textos, *links*, imagens e/ou vídeos). O *spider* localiza as *tags* correspondentes ao nome do blog (em verde), título do artigo (em azul), data de publicação (em rosa) e o conteúdo textual (em vermelho) e, após o procedimento de identificação, o *spider* realiza a extração de seus valores, tendo como resultado o arquivo representado pela Figura 18:

Figura 18: Arquivo JSON com um trecho do conteúdo do artigo.

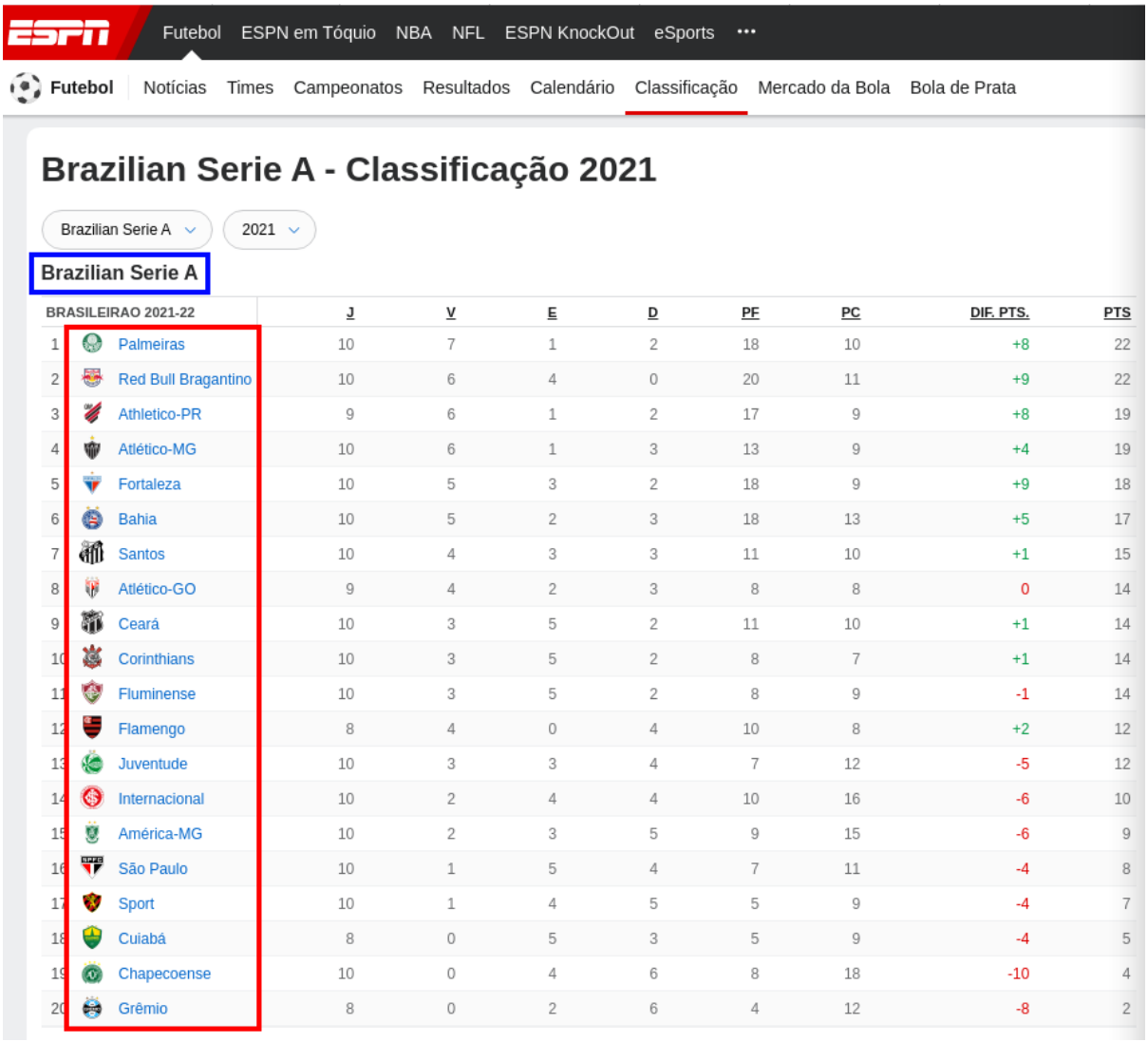

```
[
  {
    "blog": "flamengo",
    "data": "17/06/2021",
    "titulo": " Análise: Flamengo atinge seu grau mais alto de maturidade na
temporada; segundo gol simboliza ",
    "conteudo": [
      " Há 16 jogos invicto, ",
      "o time alcança seu grau mais elevado de maturidade na temporada. Mérito para
Rogério Ceni",
      ". As peças mudam, mas o patamar elevado, tanto na defesa quanto no ataque, é
mantido. O lance do segundo gol contra o Coritiba, marcado por Bruno Henrique, foi
um símbolo de como o Flamengo tem sido superior a seus adversários. ",
      "Ao todo, a jogada durou 53 segundos, com 23 trocas de passes",
      ". Renê (4 toques), Vitinho (2 toques), Gomes (5 toques), Arão (2 toques),
Matheuzinho (2 toques), Gerson (2 toques), Bruno Henrique (2 toques) e Michael (1
toque) participaram do lance. Apenas Diego Alves, Rodrigo Caio e Muniz não
encostaram na bola. ",
    ]
  }
]
```


No arquivo JSON, os dados do artigo foram organizados em uma lista de objetos cujas chaves foram definidas no Scrapy, nele é possível encontrar o nome do blog corrente, a data de publicação, o título e o conteúdo que ainda precisa passar por um tratamento antes de ser feita a identificação de suas entidades.

4.3.2 Extração dos nomes dos times

Assim como no portal do GE, a página da ESPN possui uma estrutura organizacional padrão e, seguindo o mapeamento feito na seção anterior, a página (Figura 19) divide-se em duas partes interessadas, sendo a primeira correspondente ao nome do campeonato e a outra a tabela com os nomes dos times.

Figura 19: Portal da ESPN.



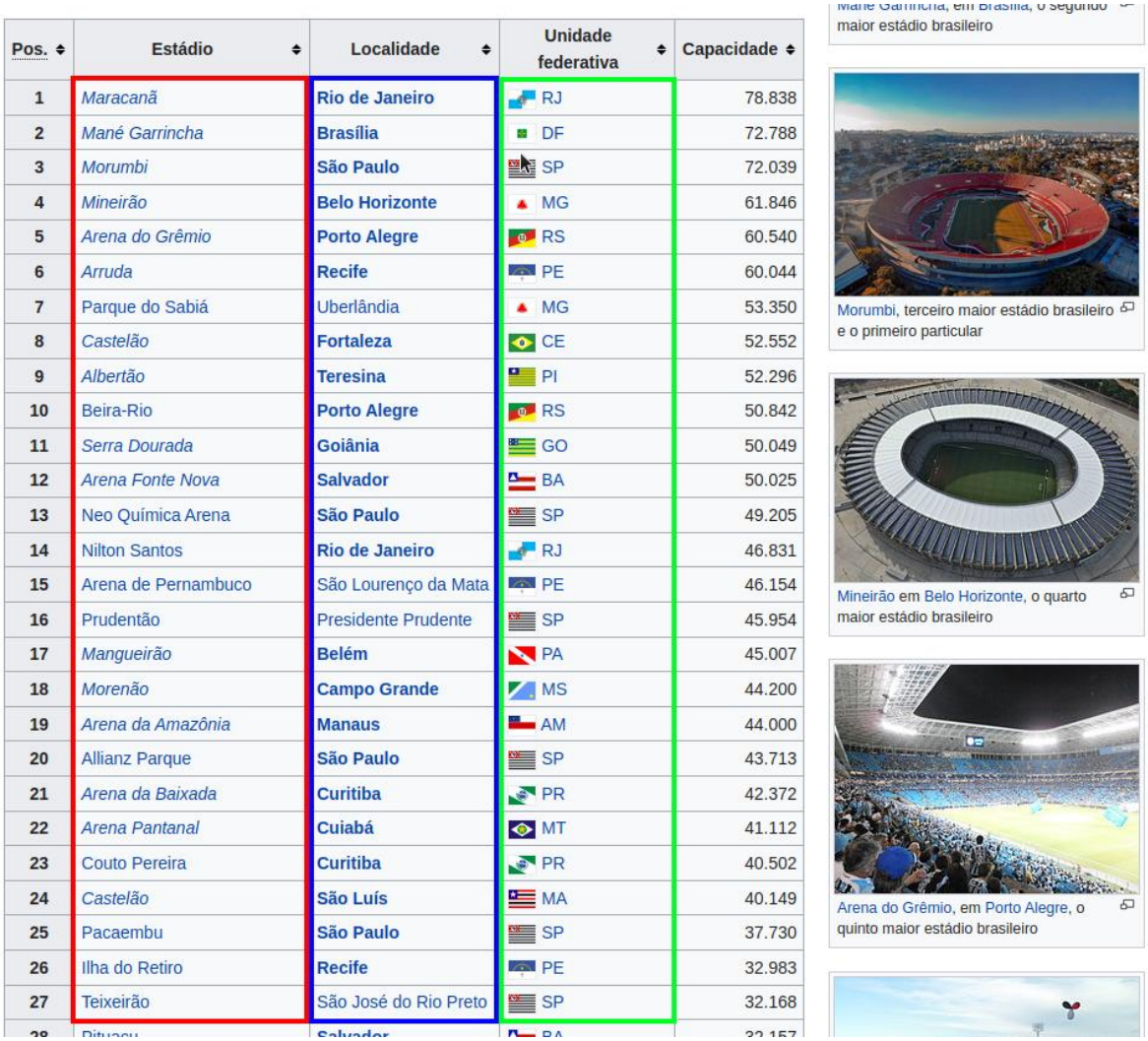
BRASILEIRAO 2021-22	J	V	E	D	PF	PC	DIF. PTS.	PTS
1 Palmeiras	10	7	1	2	18	10	+8	22
2 Red Bull Bragantino	10	6	4	0	20	11	+9	22
3 Athletico-PR	9	6	1	2	17	9	+8	19
4 Atlético-MG	10	6	1	3	13	9	+4	19
5 Fortaleza	10	5	3	2	18	9	+9	18
6 Bahia	10	5	2	3	18	13	+5	17
7 Santos	10	4	3	3	11	10	+1	15
8 Atlético-GO	9	4	2	3	8	8	0	14
9 Ceará	10	3	5	2	11	10	+1	14
10 Corinthians	10	3	5	2	8	7	+1	14
11 Fluminense	10	3	5	2	8	9	-1	14
12 Flamengo	8	4	0	4	10	8	+2	12
13 Juventude	10	3	3	4	7	12	-5	12
14 Internacional	10	2	4	4	10	16	-6	10
15 América-MG	10	2	3	5	9	15	-6	9
16 São Paulo	10	1	5	4	7	11	-4	8
17 Sport	10	1	4	5	5	9	-4	7
18 Cuiabá	8	0	5	3	5	9	-4	5
19 Chapecoense	10	0	4	6	8	18	-10	4
20 Grêmio	8	0	2	6	4	12	-8	2

O *spider* percorreu o DOM da página e fez a extração do nome do campeonato que é um identificador de qual campeonato os times pertencem. Feito isto, o *spider* identificou as *tags* correspondentes à tabela e fez a extração dos nomes dos clubes, armazenando-os em uma lista.

4.3.3 Extração dos nomes dos estádios

Utilizando o mesmo processo das etapas anteriores e informadas ao *spider* as *tags* desejadas para a extração, o programa percorreu o DOM em busca destas *tags*, então foi feita extração de seus valores.

Figura 20: Portal da Wikipédia.



Pos.	Estádio	Localidade	Unidade federativa	Capacidade
1	*Maracanã*	**Rio de Janeiro**	RJ	78.838
2	*Mané Garrincha*	**Brasília**	DF	72.788
3	*Morumbi*	**São Paulo**	SP	72.039
4	*Mineirão*	**Belo Horizonte**	MG	61.846
5	*Arena do Grêmio*	**Porto Alegre**	RS	60.540
6	*Arruda*	**Recife**	PE	60.044
7	Parque do Sabiá	Uberlândia	MG	53.350
8	*Castelão*	**Fortaleza**	CE	52.552
9	*Albertão*	**Teresina**	PI	52.296
10	Beira-Rio	**Porto Alegre**	RS	50.842
11	*Serra Dourada*	**Goiânia**	GO	50.049
12	*Arena Fonte Nova*	**Salvador**	BA	50.025
13	Neo Química Arena	**São Paulo**	SP	49.205
14	Nilton Santos	**Rio de Janeiro**	RJ	46.831
15	Arena de Pernambuco	São Lourenço da Mata	PE	46.154
16	Prudentão	Presidente Prudente	SP	45.954
17	*Mangueirão*	**Belém**	PA	45.007
18	*Morenão*	**Campo Grande**	MS	44.200
19	*Arena da Amazônia*	**Manaus**	AM	44.000
20	Allianz Parque	**São Paulo**	SP	43.713
21	*Arena da Baixada*	**Curitiba**	PR	42.372
22	*Arena Pantanal*	**Cuiabá**	MT	41.112
23	Couto Pereira	**Curitiba**	PR	40.502
24	*Castelão*	**São Luís**	MA	40.149
25	Pacaembu	**São Paulo**	SP	37.730
26	Ilha do Retiro	**Recife**	PE	32.983
27	Teixeirão	São José do Rio Preto	SP	32.168

Na página representada pela Figura 20, os valores estão armazenados em uma tabela que possui ao todo 149 linhas e 5 colunas, onde a primeira (“Pos.”) e a última coluna (“Capacidade”) foram desconsideradas. Na segunda coluna os valores referem-se aos apelidos dos estádios (Ex.: Maracanã), além dos valores, ainda nesta coluna, também foram extraídos os valores dos atributos *title*, onde estão armazenados os nomes reais dos estádios (Ex.: apelido: Maracanã, nome: Estádio Jornalista Mário Filho). Isso pode ser visto na Figura 21.

Figura 21: *Tag* da segunda coluna da tabela do Wikipédia.

```
<td>
    <a href="/wiki/Est%C3%A1dio_Jornalista_M%C3%A1rio_Filho" title="Estádio Jornalista Mário Filho">
        <i>Maracanã</i>
    </a>
</td>
```

Em seguida, na terceira coluna (“Localidade”), é extraído o nome da cidade em que o estádio está sediado (Ex.: Rio de Janeiro).

Por fim, é extraído o nome do estado e, assim, como na segunda coluna, é extraído o valor da coluna que corresponde à Unidade Federativa (UF) (Ex.: RJ) e o valor do atributo *title*, onde o nome do estado está armazenado por extenso (Ex.: Rio de Janeiro (estado)). Nos casos em que o nome da cidade e do estado são idênticos, o nome do estado possui a *string* “(estado)” no final.

4.4 PROCESSAMENTO DOS DADOS PELO LINGUAKIT

O Linguakit foi a ferramenta utilizada para a análise dos dados extraídos do portal e, diferentemente da configuração do Scrapy, para configurar o Linguakit bastou realizar a clonagem dos arquivos necessários para a execução diretamente do GitHub, que é um repositório de versionamento de arquivos.

Desenvolvido com a linguagem de programação Perl, o Linguakit foi utilizado para realizar a identificação de entidades do artigo, que foi extraído pelo Scrapy na etapa anterior. Para isto, o Linguakit possui diversos módulos que ao serem executados, resultam em dados úteis para análises posteriores.

Feita a extração do artigo, bem como suas informações complementares, o Scrapy armazenou os dados em um arquivo JSON, porém, este formato não é suportado pelo Linguakit. Para resolver este problema, foi desenvolvido um algoritmo que busca os dados do arquivo JSON e retorna um arquivo no formato de texto (TXT), suportado pelo Linguakit.

Com o Linguakit pronto para uso e o arquivo TXT devidamente preparado, é feita a execução do *script* que fará a análise: “`./linguakit <módulo> <submódulo> <idioma> [./origem] > [./destino]`”, onde:

- **linguakit**: refere-se ao comando de chamada da ferramenta;
- **módulo**: refere-se ao tipo de análise que será realizada no texto. Por exemplo: o módulo *tok* fará a tokenização de cada um dos elementos presentes no texto;
- **submódulo (opcional)**: é um acompanhamento do módulo que, especifica o tipo de análise que deverá ser feita. Por exemplo: o módulo *tok* acompanhado do submódulo -*sort* fará a tokenização do texto e os ordenará, em ordem decrescente, de acordo com a frequência que cada *token* aparece no texto. Se nenhum submódulo for inserido, o Linguakit utilizará o submódulo padrão.

- **idioma**: refere-se ao idioma do texto, onde somente os idiomas espanhol, galego, galego-português[14], inglês e português são suportados;
- **origem**: refere-se ao diretório em que o arquivo com o texto a ser analisado se encontra;
- **destino (opcional)**: refere-se ao diretório onde o texto analisado será armazenado.

Para o desenvolvimento deste trabalho, foi utilizado o módulo *tagger* acompanhado do submódulo *-nec*, assim, realizando a tokenização do artigo e a etiquetagem gramatical de cada um dos *tokens*. Podemos ver o resultado do *script* com o módulo *tagger -nec* no Exemplo 3.

Exemplo 3: "O ex-jogador Zico, ídolo do Flamengo, participou do revezamento da tocha olímpica, que chegará ao Estádio Olímpico de Tóquio.".

A frase descrita acima, gerou o seguinte resultado representado pela Quadro 1:

Quadro 1: Etiquetagem das classes gramaticais do Exemplo 3.

FORMA	LEMA	ETIQUETA
O	o	DA0MS0
ex-jogador	ex-jogador	NC00000
Zico	zico	NP00SP0
,	,	Fc
ídolo	ídolo	NCMS000
de	de	SPS00
o	o	DA0MS0
Flamengo	flamengo	NP00V00
,	,	Fc
participou	participar	VMIS3S0
de	de	SPS00
o	o	DA0MS0

[14] Saiba mais sobre o idioma galego-português: http://www.usp.br/gmhp/publ/AreA7.pdf

revezamento	revezamento	NCMS000
de	de	SPS00
a	o	DA0FS0
tocha	tocha	NCFS000
olímpica	olímpico	AQ0FS0
,	,	Fc
que	que	PR0CN000
chegará	chegar	VMIF3S0
a	a	SPS00
o	o	DA0MS0
Estádio_Olímpico_de_Tóquio	Estádio_Olímpico_de_Tóquio	NP00V00
.	.	Fp

A execução do Linguakit com o módulo *tagger -nec* retornou 3 informações sobre cada um dos *tokens* que são: a forma, o lema e a etiqueta. No trecho "participou participar VMIS3S0" tem-se que: (i) a forma é o estado em que a palavra se encontra na frase ("participou"), (ii) o lema é o estado natural da palavra ("participar"), ou seja, sua forma canônica, deve ser sempre escrita no infinitivo e (iii) a etiqueta é o rótulo que classifica gramaticalmente a forma da palavra ("VMIS3S0"), onde o primeiro elemento da etiqueta, refere-se à classe gramatical da palavra, sendo "V" o código da classe Verbo e os outros elementos da etiqueta referem-se a atributos que caracterizam este verbo. No decorrer deste tópico, será apresentado mais detalhes sobre a estrutura de representação das etiquetas.

Esta etiqueta é baseada no conjunto de rótulos propostos pelo Grupo Consultivo de Peritos em Normas de Engenharia da Linguagem (EAGLES[15], em inglês) que, segundo Calzolari et al. (1996) tem como propósito desenvolver padrões e diretrizes para anotações morfossintáticas de léxicos e corpus.

As classes gramaticais no Linguakit são organizadas em quadros que demonstram como se deu a etiquetagem dos *tokens* no texto. Os Quadros 2 e 3, demonstram como esta etiquetagem funciona.

[15] Sobre o EAGLES: http://www.ilc.cnr.it/EAGLES96/intro.html

Quadro 2: Organização padrão dos quadros das etiquetas.

ETIQUETAS			
Posição	**Atributo**	**Valor**	**Código**
Coluna 1	Coluna 2	Coluna 3	Coluna 4

Todos os quadros seguem um padrão organizacional, onde a coluna 1 corresponde ao índice em que os atributos aparecem no quadro. A coluna 2 corresponde a estes atributos, onde a quantidade de atributos varia de acordo com a categoria que é composta por substantivo, adjetivo, verbo, entre outras. Na coluna 3, há os valores que cada atributo pode assumir. Por fim, na coluna 4 há a representação em código de cada um dos valores. As etiquetas são exclusivamente compostas por estes códigos, onde a posição de cada um dos códigos é determinada de acordo com a sequência que aparecem no quadro.

No trecho "participou participar VMIS3S0", a palavra "participou" trata-se de um verbo e no Linguakit, os verbos flexionam-se de quatro maneiras: **modo**, composto por indicativo, subjuntivo e imperativo, as formas nominais do verbo (infinitivo, gerúndio e particípio) também estão inclusas no modo; **tempo**, composto por presente, futuro e pretérito; **pessoa**, composto por primeira, segunda e terceira; e **número**, composto por singular e plural. É importante salientar que a flexão verbal **voz** não é abordada pelo Linguakit, sendo incorporada a outras flexões.

Além das flexões verbais, há a **categoria** na qual a palavra pertence, o **tipo** da palavra que, nos verbos é composto por principal, auxiliar e semi-auxiliar e o **gênero** (masculino ou feminino) do verbo. Dito isto, o quadro referente à classe gramatical verbo, é elaborado de acordo com o Quadro 3.

Quadro 3: Quadro de apoio para a elaboração da etiqueta.

VERBOS			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Verbo	V
2	Tipo	Principal	M
		Auxiliar	A
		Semi Auxiliar	S
3	Modo	Indicativo	I

		Subjuntivo	S
		Imperativo	M
		Infinitivo	N
		Gerúndio	G
		Particípio	P
4	Tempo	Presente	P
		Passado	S
		Pretérito Imperfeito	I
		Futuro	F
		Futuro do Pretérito	C
		-	0
5	Pessoa	Primeira	1
		Segunda	2
		Terceira	3
6	Número	Singular	S
		Plural	P
7	Gênero	Masculino	M
		Feminino	F

Logo, a palavra “participou” no contexto em está inserida, trata-se de um verbo (“V”), principal (“M”), indicativo (“I”), que se encontra no passado (“S”), pertencente à terceira pessoa (“3”) do singular (“S”). Há casos em que algum atributo não se aplica à palavra que está sendo analisada, nestas situações o valor do código é 0. Portanto, a palavra “participou” recebe a etiqueta “VMIS3S0”. Os quadros contendo as descrições sobre as outras classes gramaticais, encontram-se nos ANEXOS.

4.5 PROCESSAMENTO DOS DADOS EXTRAÍDOS

O processamento dos dados tem por função buscar e organizar informações nos artigos extraídos, como as entidades ou adjetivos que descrevem estas entidades, por exemplo. Para a organização destes dados, foram estabelecidas 6 regras que descrevem e determinam como esta

organização se dará. No decorrer do desenvolvimento destas regras foram realizados inúmeros testes, a fim trazer resultados satisfatórios com o que foi estabelecido, onde em cada um dos testes eram feitas iterações até que o resultado desejado fosse encontrado.

4.5.1 Regra 1: relacionar entidades simples às entidades compostas

No decorrer do artigo, uma mesma entidade pode possuir diferentes representações, ocorrendo, frequentemente, com entidades compostas que podem ser representadas por sua integralidade (nome1_nome2) ou por uma parte (nome2), ocasionando em redundâncias, visto que, onde deveria haver apenas uma entidade (seja ela um ser, objeto, lugar, etc.), existem duas. Para a resolução desta problemática, foi criada uma função que realiza uma busca por todos os substantivos do artigo, armazenando os substantivos compostos em um dicionário, onde estes substantivos são estabelecidos como chaves deste dicionário que recebe como valor uma lista de substantivos simples. Esta busca pelos substantivos, é feita por meio da análise das etiquetas dos *tokens* que, no caso dos substantivos, recebem o código N como primeiro elemento (Ex.: "Flamengo flamengo **NP000O0**"). Observe o Exemplo 4:

Exemplo 4: "**André Fernandes** joga bem no Flamengo, porém, o futuro de **Fernandes** é incerto no clube".

Nos elementos em destaque na frase, é possível observar que há dois substantivos distintos, mas que fazem referência a uma mesma entidade, para que não haja redundâncias entre as entidades, a regra criada tem por objetivo direcionar uma busca a todos os substantivos compostos ("André Fernandes") e os armazenar como chaves de um dicionário ({"André_Fernandes: [ ]}), se durante esta busca for encontrado um substantivo simples, será, primeiramente, armazenado em uma lista paralela com todos os substantivos simples presentes no artigo (["Flamengo", "futuro", "Fernandes", "clube"]). Ao final da busca pelos substantivos, será feita uma verificação na lista paralela, que tem por objetivo buscar qual(is) elemento(s) desta lista constam na chave do dicionário, ou seja, irá verificar se "Flamengo" é exatamente igual a "André" ou "Fernandes", se a verificação for positiva, "Flamengo" será armazenado como um dos valores da chave corrente, se não, irá passar para o próximo item da lista. No fim da execução da função, o resultado será o dicionário {"André_Fernandes": ["Fernandes"]}, portanto, sempre que "Fernandes" – ou "André" – for citado em uma frase, será interpretado que o sujeito ali presente, trata-se de "André Fernandes".

No entanto, este tipo de abordagem causa um problema quanto aos nomes parecidos. No elenco de jogadores do Flamengo, por exemplo, existem dois jogadores com o mesmo nome, o Diego Alves e o Diego Ribas, porém, o Diego Ribas é habitualmente chamado apenas de Diego. Em um artigo onde apenas o Alves ou Ribas é citado, a regra será aplicada sem quaisquer problemas, entretanto, quando ambos forem citados e o Ribas for chamado apenas de Diego, a regra interpretará que Diego trata-se de Alves ou da primeira ocorrência do nome composto, ou seja, se Diego Alves for citado primeiro, a sua chave no dicionário estará à frente da de Ribas, portanto, terá uma preferência nas inserções dos substantivos simples encontrados.

4.5.2 Regra 2: identificar lista de entidades

Ao longo de alguns artigos, é comum que haja a presença de lista de entidades, sendo todos os elementos da lista de um mesmo tipo, seja uma lista de jogadores, de times de futebol, de estádios, entre outros. Ao analisar 50 artigos, foi identificada a presença destas listas de entidades em 26 destes artigos, das quais 82% apresentam um padrão para representar estas listas entidades, onde uma lista é composta por duas ou mais entidades, sendo a menor representada por duas entidades, ligadas por uma conjunção: [entidade] + [conjunção] + [entidade]. E a maior contendo um número finito de elemento inclusos:

$$e_1 + p_1 + e_2 + p_2 + \ldots + e_{n-1} + c_1 + e_n \quad ,$$

onde e representa a entidade, p indica a pontuação e c, a conjunção.

Esta regra visa, além da busca destas listas de entidades, padronizar as etiquetas das entidades, uma vez que, o Linguakit apresenta falhas na etiquetagem dos elementos, ocorrendo principalmente quando não há uma descrição detalhada do que a entidade que está sendo etiquetada representa. O Exemplo 5 demonstra uma ocorrência desse problema.

Exemplo 5: “O Flamengo terá o reforço de jogadores como Pedro, Vitinho e Hugo Souza”.

O Quadro 4 apresenta a etiquetagem do Exemplo 5.

Quadro 4: Quadro com a etiquetagem do Exemplo 5.

FORMA	LEMA	ETIQUETA
O	o	DA0MS0

Flamengo	flamengo	NP00V00
terá	ter	VMIF3S0
o	o	DA0MS0
reforço	reforço	NCMS000
de	de	SPS00
jogadores	jogador	NCMP000
como	como	RG
Pedro	pedro	NP00SP0
,	,	Fc
Vitinho	vitinho	NP00V00
e	e	CC
Hugo_Souza	hugo_souza	NP00SP0

Nas entidades referentes aos jogadores, encontra-se os *tokens* “Pedro pedro NP00SP0”, “Vitinho vitinho NP00V00” e “Hugo_Souza hugo_souza NP00SP0”, nota-se que as entidades “Pedro” e “Hugo Sousa” possuem, corretamente, a etiqueta referente à uma pessoa “NP00SP0” – o quadro demonstrando como decorreu a elaboração desta etiqueta está no ANEXO D. Entretanto, o Linguakit não foi capaz de compreender que a entidade “Vitinho” também trata-se de uma pessoa e, ao não conseguir inferir o tipo da entidade, a etiquetou como uma entidade variada, representada pela etiqueta “NP00V00”.

Para corrigir este problema, é feita uma verificação, onde são contabilizados os tipos de etiquetas presentes na lista. A etiqueta com maior frequência será a etiqueta dominante, com isto, nos *tokens* que tiverem uma etiquetagem diferente da dominante, será feita a substituição da etiqueta.

No Exemplo 5, a lista encontrada possui três *tokens*, cada um contendo sua respectiva etiqueta “NP00SP0”, “NP00V00” e “NP00SP0”, então é feita a contagem dos tipos de etiquetas, onde de um total de 3 etiquetas (100%) são encontradas 2 etiquetas “NP00SP0” (67%) e 1 etiqueta “NP00V00” (33%). Por conta da etiqueta “NP00SP0” possuir uma presença maior na lista, torna-se a etiqueta dominante e seu rótulo será aplicado a todos os outros *tokens* da lista. Agora, sempre que a entidade “Vitinho” for mencionada ao longo do texto, sua etiquetagem passará a ser “NP00SP0” e não mais “NP00V00”.

Ao executar o Linguakit em uma lista contendo 100 nomes de jogadores, 82 nomes foram etiquetados como “NP00SP0”, 15 nomes foram etiquetados como “NP00V00” e 3 como “NP00G00” – a lista de nomes de jogadores utilizados para a contagem, está no APÊNDICE A. Se na contagem de etiquetas, houver um empate, 50% para um grupo de etiquetas e 50% para o outro, e, um dos grupos for composto pela etiqueta “NP00SP0”, será dada a preferência para esta etiqueta, uma vez que, há uma alto nível de assertividade (82%) na inferência das etiquetas.

Esta regra é aplicada somente para as entidades do tipo “pessoa”, ou seja, para os *tokens* que possuam a etiqueta “NP00SP0”, visto que as informações necessárias sobre os times de futebol e estádios foram extraídas nas etapas anteriores.

4.5.3 Regra 3: substituir etiquetas dos tokens dos times

De acordo com o contexto, uma entidade pode apresentar diferentes significados, neste trabalho, estes significados se traduzem em etiquetas, como a entidade “Fortaleza” que pode apresentar, tanto a etiqueta “NP00G00” que corresponde à uma localidade (cidade), quanto a “NP00O00” que corresponde à uma organização (time de futebol), caso não seja possível identificar o tipo da entidade “Fortaleza”, a etiqueta a ser atribuída será “NP00V00”, correspondendo a uma entidade variada/diversa. Esta regra tem por objetivo inferir a etiquetagem das entidades, quando no contexto em que estiverem inseridas, se referirem a um time de futebol. Observe o Exemplo 6:

Exemplo 6: “O Flamengo vai enfrentar o Palmeiras em São Paulo”.

Executando o Linguakit com o módulo *tagger -nec,* é gerada a seguinte etiquetagem:

Quadro 5: Quadro com a etiquetagem do Exemplo 6.

FORMA	LEMA	ETIQUETA
O	o	DA0MS0
Flamengo	flamengo	NP00V00
vai	ir	VMIP3S0
enfrentar	enfrentar	VMN03S0
o	o	DA0MS0
Palmeiras	palmeiras	NP00G00

em	em	SPS00
São_Paulo	são_paulo	NP00G00

Na frase, é possível observar a presença de três entidades: Flamengo, Palmeiras e São Paulo, onde duas destas são times de futebol (Flamengo e Palmeiras) e a outra é uma localidade (São Paulo). No entanto, o Linguakit não conseguiu inferir a etiquetagem adequada para as entidades, para isto, cada entidade precisaria ter um detalhamento maior, como a inserção do trecho "time de futebol", por exemplo, antes dos nomes dos times, porém, isso nem sempre ocorre. Para resolver este problema, foi criada esta regra que busca por todos os *tokens* cujos códigos iniciais das etiquetas sejam "NP" (N - Nome, P - Próprio) e, verifica se o *token* existe no arquivo contendo os nomes dos times de futebol que foram extraídos nas etapas anteriores deste trabalho. Se o token for exatamente igual a um dos nomes contidos no arquivo, a sua etiqueta será alterada para "NP00O00".

Entretanto, esta regra também seria erroneamente aplicada ao token "São_Paulo são_paulo NP00G00", visto que São Paulo, além de uma cidade, corresponde ao nome de um time pertencente à série A do campeonato brasileiro. Para que isto não ocorra, antes da substituição das etiquetas, foi feita uma verificação que realiza uma busca por preposições nos *tokens* que antecedem a entidade que está em análise, uma vez que, quando uma localidade é mencionada, ela é, frequentemente, precedida por uma preposição ("[...] **em** São Paulo" ou "[...] na cidade **de** São Paulo", por exemplo), onde as preposições são representadas pela etiqueta "SPS00", em outras palavras, é verificado se a entidade em análise é precedida por um *token* com a etiqueta "SPS00".

Como pode ser observado no quadro, apesar da entidade "Palmeiras" possuir a etiqueta de localidade, ela é precedida pelo artigo definido "o" que possui a etiqueta "DA0MS0", o que o torna elegível para a substituição de etiquetas, enquanto a entidade "São Paulo" é precedida pela preposição "em" ("SPS00"), tornando-o inelegível para a substituição de etiquetas, mantendo sua etiquetagem original.

Há casos em que a entidade, quando se refere a um time, também é precedida por uma preposição, "[...] **no** Flamengo", por exemplo, no entanto, a preposição "no" trata-se de uma contração da preposição "em" com o artigo definido "o". Quando o Linguakit realiza a análise, ele identifica que há essa contração e retorna a preposição "no" para o seu "estado natural" ("[...] em o Flamengo"), logo, a entidade Flamengo, será precedida de um artigo definido ("o o

DA0MS0") e não de uma preposição ("em em SPS00"), tornando-o elegível para a substituição de etiquetas.

4.5.4 Regra 4: encontrar adjetivos

No decorrer do artigo, encontram-se informações que vão além das entidades nomeadas em si, como os adjetivos, que trazem mais informações sobre estas entidades. Na língua portuguesa, os adjetivos são regularmente precedidos de um substantivo (substantivo + adjetivo), "noite escura", por exemplo, embora não seja raro encontrar outras classes gramaticais ligando-os, "Neymar é genial" (substantivo + verbo + adjetivo).

Com base nisso, foi criada esta regra que visa buscar por estas informações adicionais das entidades. Para isso, é verificado se as iniciais do *token* correspondem a "NP", certificando-se de que se trata de uma entidade, essa entidade é adicionada em um dicionário que recebe uma lista de listas como valor, onde a chave do dicionário corresponde ao próprio nome da entidade e, então, busca-se por adjetivos nos tokens seguintes. Se entre o adjetivo e o substantivo existir um verbo, este verbo será adicionado na lista de adjetivos, juntamente com o adjetivo encontrado, se não, adotou-se o verbo "é (ser)" como verbo de ligação padrão.

Exemplo 7: "Pelé foi genial".

No Exemplo 7 há três tokens, aplicando-se o Linguakit o resultado gerado é: o substantivo "Pelé pelé NP00SP0", o verbo "foi ser VMIS3S0" e o adjetivo "genial genial AQ0CS0". De acordo com o que foi estabelecido na regra, o substantivo assumiria como chave do dicionário e os outros *tokens* seriam seus valores – {"Pelé": [["foi", "genial"]]}. Caso não existisse o verbo "foi (ser)", no exemplo citado acima ("Pelé genial", por exemplo), o primeiro elemento da lista seria o verbo "é (ser)", resultando no seguinte dicionário {"Pelé": [["é", "genial"]]}.

Além do verbo de ligação, podem existir outros elementos que separam o verbo do adjetivo ("Pelé foi **muito** genial", por exemplo), no entanto, estes elementos são descartados pela regra, visto que a informação já disposta pelo adjetivo é suficiente para descrever o substantivo.

Encontrada uma entidade e durante a busca por um adjetivo, for encontrado outro substantivo ou uma pontuação (vírgula, ponto final, entre outras), a entidade corrente no dicionário será descartada e uma nova entidade assumirá sua posição, em busca de um adjetivo que a descreva.

4.5.5 Regra 5: buscar dia da próxima partida

Durante o artigo, é comum que sejam citadas informações sobre a próxima partida do time, entre estas informações podem estar o dia da semana da partida, o horário, o adversário, entre outras. Após analisar 50 artigos e, dos artigos que apresentam informações sobre a próxima, em 85% destes artigos seguem um padrão organizacional informando o dia e horário da partida: "[dia da semana], às [horário], [preposição] [local da partida ou adversário], [preposição] [local da partida ou adversário]", também é encontrado no padrão "[adversário], [preposição] [dia da semana], às [horário], [preposição] [local da partida]". O primeiro padrão pode ser observado no Exemplo 8. Os artigos analisados estão disponíveis no APÊNDICE B.

Exemplo 8: "A próxima partida do Flamengo será neste domingo, às 16h, contra o Fluminense, no Maracanã".

Identificado o padrão, esta regra tem por função buscar as informações sobre a partida e armazená-las em um dicionário, onde cada informação extraída da frase, terá uma chave para identificá-la. A primeira informação sobre a partida corresponde ao dia da semana (domingo, segunda-feira, terça-feira, etc.) que são rotulados com uma etiqueta de código único "W". Em seguida, busca-se o horário da partida que é rotulado pela etiqueta "NC00000", encontrada esta etiqueta, a informação sobre o horário é armazenada.

Na terceira e quarta etapa, são feitas verificações para identificar se o *token* que está sob análise, se refere ao adversário ou ao local da partida. Seguindo a frase do Exemplo 8, (i) é verificado se o *token* está contido no arquivo JSON onde os nomes dos times extraídos estão armazenados, se a condição for verdadeira, o *token* em questão trata-se de um time de futebol, portanto, o adversário para a próxima partida, se for falsa, (ii) será verificado se o *token* está contido no arquivo JSON com apelidos dos estádios extraídos, se a condição for verdadeira, o *token* trata-se do apelido de um estádio, portanto, o local da partida. Se o apelido do estádio for identificado durante este processo, além de armazenar o apelido, também será armazenado o nome da cidade e do estado em que o estádio está sediado.

No Exemplo 8 é informado o dia da semana em que o jogo ocorrerá, porém, não há a informação do dia do mês, para resolver este problema, é feita a extração da data de publicação do artigo, com esta data e com a informação do dia da semana do jogo, é possível mensurar o dia que a partida ocorrerá. Para isto, foi utilizado uma ontologia contendo os dias da semana ["segunda-feira", "terça-feira", "quarta-feira", "quinta-feira", "sexta-feira", "sábado", "domingo"]. Feito isto, por meio da função *.weekday()* do Python é identificado o índice do dia

da semana em que o artigo foi publicado, por exemplo, se o artigo foi publicado no dia 02/07/2021, o dia da semana foi uma sexta-feira, portanto, o índice é 4. E com a função *.index()* também do Python, é identificado o índice do dia da semana em que a partida ocorreu, como foi informado no Exemplo 8, foi em um domingo, portanto o índice é 6. Com estes índices, agora é possível descobrir o dia da partida por meio de algumas verificações.

Se o índice do dia da semana for maior que o índice do dia da data de publicação do artigo, é feito um somatório (com o auxílio do função *timedelta* que converte números em duração de tempo) entre a data de publicação do artigo e o resultado da subtração entre o índice do dia da semana e o índice da data de publicação, em resumo, "Σ = 02/07/2021 + timedelta(days = 6 - 4)", logo, o data da partida foi 04/07/2021.

Se o índice do dia da semana for menor que o índice da data de publicação do artigo, o processo será semelhante ao da etapa anterior, com a exceção de que após a subtração, deve-se somar o resultado com 7. Isso é necessário, pois se o dia da semana do jogo for quarta-feira (índice 2) e a data de publicação do artigo for 04/07/2021, em um domingo (índice 6), a subtração destes índices resultará em um valor negativo (-4) que quando for somado à data de publicação do artigo, retornará uma data anterior à data de publicação do mesmo. Por conta disto, deve-se somar o valor 7 após a subtração, para que este erro seja corrigido. Logo, o somatório será "Σ = 04/07/2021 + timedelta(days = 7 + (2 - 6))", resultando na data 07/07/2021.

Por fim, se o índice do dia da semana for igual ao índice da data de publicação do artigo, será retornada a data de publicação do artigo.

4.5.6 Regra 6: relacionar pessoas aos times

No decorrer de um artigo publicado em um portal esportivo, são mencionados os nomes de diversas entidades, sendo as mais frequentes, nomes de times e de pessoas. Após analisar 50 artigos (APÊNDICE B) de 5 *blogs* distintos, foi constatado que quando uma pessoa é citada no artigo, em 91% das situações, esta pessoa tinha alguma relação com o time do qual o *blog*, onde o artigo se encontra, é especializado, seja essa pessoa um jogador, técnico, dirigente, entre outras funções.

Com base nisso, foi criada esta regra que tem por objetivo criar uma relação entre as pessoas citadas no artigo ao time do *blog*. Em outras palavras, se o artigo em análise estiver no blog do Flamengo e o nome do jogador Gabriel Barbosa for citado no decorrer do artigo, a regra presumirá, com base nas estatísticas, que Gabriel Barbosa, pertence ao Flamengo.

4.6 CODIFICAÇÃO PARA O NEO4J

Aplicado-se as regras e com os dados devidamente processados e organizados, deve-se inseri-los no Neo4j, mas para que isso ocorra, os dados precisam ser convertidos para a linguagem *Cypher Query Language* (CQL), que é uma linguagem de consulta de grafo declarativa que permite a consulta, atualização e administração do grafo utilizado no Neo4j (NEO4J, 2021). O *Cypher* é dividido em duas partes: as entidades e suas relações. Observe o Exemplo 9:

Exemplo 9:

***blog*:** Flamengo
data de publicação: 06/07/2021
artigo: "O Flamengo está pronto e já escalou os jogadores Gabigol, Arrascaeta e Bruno Henrique para a partida desta quinta-feira, às 21h, contra o Palmeiras, no Allianz Parque".

Assumindo que o artigo já foi extraído e o Linguakit já foi executado, será feito, então, o processamento dos dados:

4.6.1 Aplicação das regras

- **Regra 1 - relacionar entidades simples às entidades compostas:** nesta regra os *tokens* "Bruno_Henrique bruno_henrique NP00SP0" e "Allianz_Parque allianz_parque NP00G00" atenderam aos requisitos da regra, porém, como não há ocorrências de entidades simples que correspondam às compostas, não foi tomada qualquer medida;
- **Regra 2 - identificar lista de entidades:** foi feita a busca por um padrão de lista de entidades, nesta busca, foi encontrada a lista contendo as entidades "Gabigol gabigol NP00SP0", "Arrascaeta arrascaeta NP00SP0" e "Bruno_Henrique bruno_henrique NP00SP0", então foi feita a contagem dos tipos de etiquetas, porém, como todas apresentaram a mesma etiquetagem, não foi necessário realizar quaisquer alterações em seus valores;
- **Regra 3 - substituir etiquetas dos tokens dos times:** nesta regra, foram identificados os *tokens* referentes aos times "Flamengo flamengo NP00V00" e "Palmeiras palmeiras NP00G00", que tiveram suas etiquetas substituídas por

"NP00O00", visto que é mais adequada para o contexto que os *tokens* estão inseridos;

- **Regra 4 - encontrar adjetivos:** nesta regra, foi feita a busca de adjetivos que estejam descrevendo substantivos que os precedem. Isto ocorre no trecho "O Flamengo está pronto [...]", onde "Flamengo" é o substantivo, "está" é o verbo de ligação e "pronto" o adjetivo;
- **Regra 5 - buscar dia da próxima partida:** aplicando-se a quinta regra, foram encontradas informações sobre um próximo jogo do Flamengo, que ocorreu no dia 08/07/2021, às 21h, contra o Palmeiras, no estádio Allianz Parque, na cidade de São Paulo, no estado de São Paulo;
- **Regra 6 - relacionar pessoas aos times:** na sexta regra, todas as ocorrências de *tokens* com a etiqueta "NP00SP0" foram associadas ao Flamengo, pelo fato do artigo ter sido postado no *blog* do clube.

Cada uma destas informações foram armazenadas em dicionários para permitir uma facilitação na manipulação dos dados posteriormente.

Aplicado-se as regras, as seguintes informações foram obtidas:

- **Times:** Flamengo e Palmeiras;
- **Pessoas:** Gabigol, Arrascaeta e Bruno Henrique;
- **Lugares:** Allianz Parque, São Paulo, São Paulo (estado);
- **Data da partida:** quinta-feira, 21h, 08/07/2021
- **Adjetivo:** pronto.

4.6.2 Criação do Cypher

Nesta etapa, foi criado uma *query Cypher* para cada um dos itens obtidos na seção anterior e, também para suas relações, armazenando-os em um arquivo externo do tipo ".cql" que é extraído e processado pela codificação do Neo4j. Primeiramente, foram criadas, no *Cypher*, as *queries* correspondentes às entidades e suas informações complementares que nada mais são que os nós do grafo. Essas *queries* são elaboradas da seguinte maneira: "CREATE (variavel: Classe {id: valor, atributo: valor})", onde o "id" varia conforme a quantidade de elementos com a mesma Classe, como pode ser observado na Figura 22

Figura 22: Query em Cypher.

```
CREATE (flamengo:Time {id:1, nome:'Flamengo'})
CREATE (palmeiras:Time {id:2, nome:'Palmeiras'})

CREATE (gabigol:Jogador {id:1, nome:'Gabigol'})
CREATE (arrascaeta:Jogador {id:2, nome:'Arrascaeta'})
CREATE (bruno_henrique:Jogador {id:3, nome:'Bruno Henrique'})

CREATE (pronto:Adjetivo {id:1, nome:'pronto'})

CREATE (partida:Partida {id:1, nome:'Partida'})
CREATE (quinta_feira:Dia_semana {id:1, nome:'quinta-feira'})
CREATE (horas_21h:Horario {id:1, nome:'21h'})
CREATE (allianz_parque:Estadio {id:1, nome:'Allianz Parque'})
CREATE (sao_paulo_cidade:Cidade {id:1, nome:'São Paulo'})
CREATE (sao_paulo_estado:Estado {id:1, nome:'São Paulo'})
CREATE (data_17_06_21:Data {id:1, nome:'17/06/21'})
```

Criados os elementos que irão compor o grafo, agora deve-se criar as relações existentes entre eles. As *queries* de relações do *Cypher*, seguem a seguinte estrutura: “CREATE (variavel_a)-[:RELACAO]->(variavel_b)”, onde “variavel_n” corresponde ao nó do grafo e “RELACAO” a aresta. A Figura 23, mostra as relações que foram criadas:

Figura 23: Relações no Cypher.

```
CREATE (gabigol)-[:PERTENCE_AO]->(flamengo)
CREATE (arrascaeta)-[:PERTENCE_AO]->(flamengo)
CREATE (bruno_henrique)-[:PERTENCE_AO]->(flamengo)

CREATE (flamengo)-[:ESTA]->(pronto)

CREATE (flamengo)-[:PROXIMA_PARTIDA]->(partida)
CREATE (partida)-[:DIA_SEMANA]->(quinta_feira)
CREATE (quinta_feira)-[:HORARIO]->(horas_21h)
CREATE (partida)-[:ADVERSARIO]->(palmeiras)
CREATE (partida)-[:ESTADIO]->(allianz_parque)
CREATE (allianz_parque)-[:CIDADE]->(sao_paulo_cidade)
CREATE (sao_paulo_cidade)-[:ESTADO]->(sao_paulo_estado)
CREATE (horas_21h)-[:DATA]->(data_17_06_21)
```

Começando pelos jogadores, e, como estabelecido na Regra 6, todas as entidades do tipo “pessoa” foram relacionadas ao clube, na qual o *blog* onde o artigo está situado, é especializado.

No Exemplo 9, o *blog* é especializado no clube Flamengo, logo, Gabigol, Arrascaeta e Bruno Henrique foram relacionados ao Flamengo.

Em seguida foram feitas as relações das informações complementares: o adjetivo “pronto”, assim, como no texto de exemplo, foi relacionado ao “Flamengo” e o verbo de ligação “está” foi utilizado como nome da relação.

Nas informações sobre o próximo jogo, foi criado a *query* auxiliar “partida” que permitiu organizar estas informações, em outras palavras, todas as informações sobre o jogo foram inseridas dentro do nó “partida”. Feito isto, foi criada a relação entre as informações da partida, onde a informação sobre o dia da semana em que a partida ocorreu, o próximo adversário e o nome do estádio foram contidos no nó “partida”. O horário foi contido no nó do dia da semana. As informações referentes à cidade e estado em que o estádio está sediado foram contidas no nó do estádio. Por fim, a data da partida foi contida no nó do horário. Se em uma eventualidade, alguma informação sobre a partida não existir, o seu nó “filho” correspondente, será inserido no nó mais próximo, ou seja, se a informação sobre o dia da semana não for informada, o nó do horário será inserido diretamente no nó “partida”.

4.6.3 Inserção do código Cypher no Neo4j

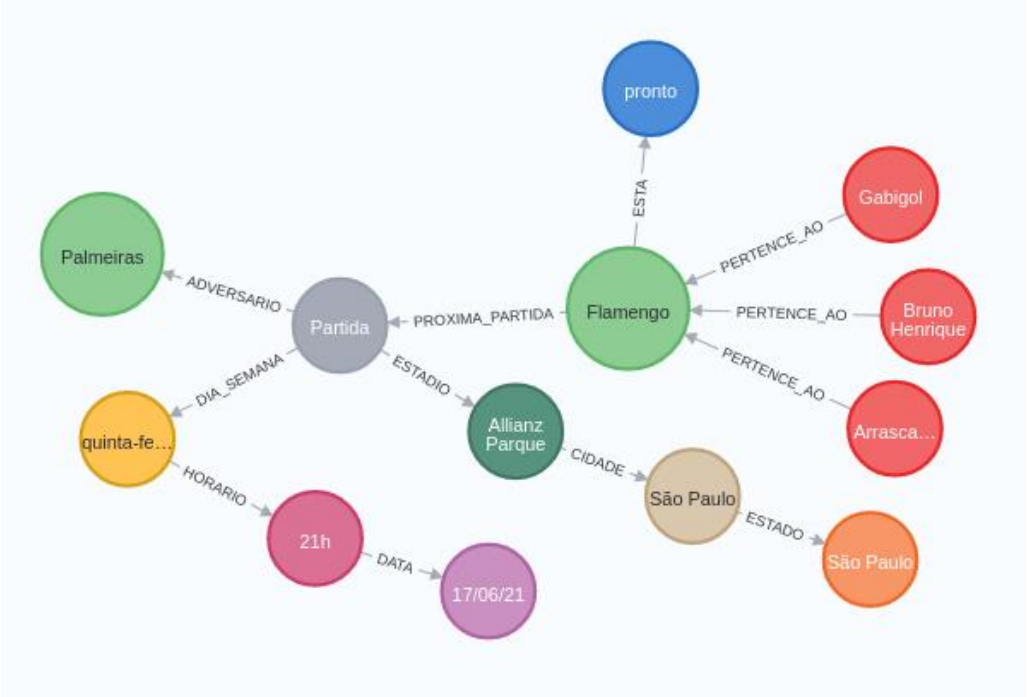
O Neo4j dispõe de suporte para várias linguagens de programação, permitindo a manipulação dos dados por meio destas linguagens. Para este trabalho, foi utilizado o suporte para o Python, sendo utilizado exclusivamente para inserir as *queries* do arquivo “.cql”, criado na seção anterior, no Neo4j.

Figura 24: Inserir dados no Neo4j.

```python
def criar_e_retorar_dados(tx, message):
    with open('codeForNeo4j.cql', 'r') as cqlFile:
        text = cqlFile.read()

    query = (
        f"{text}"
        "RETURN $message"
    )

    result = tx.run(
        query,
        message=message
    )

    return result.single()[0]
```

Como pode ser observado na Figura 24, o código Python acessou o arquivo ".cql", extraiu todos os seus dados e os armazenou na variável "*text*" que foi inserida em uma "*superquery"* e, posteriormente, executada pela transação do Neo4j "*tx.run*" que é um recipiente para que várias consultas *Cyphers* sejam executadas em um único contexto. Ao fim da execução do código Python, o Neo4j compilou as *queries*, resultando no grafo da Figura 25:

Figura 25: Grafo de Conhecimento do artigo do Exemplo 9.



Na Figura 25 é apresentado o grafo com os elementos obtidos do artigo do Exemplo 9, onde os círculos são os nós do grafo e nestes nós estão as entidades extraídas do artigo, como Flamengo, Arrascaeta, Gabigol, entre outros, bem como, as informações complementares. Entre estes nós tem-se as arestas devidamente nomeadas descrevendo suas relações.

Aplicando a ferramenta em um artigo publicado no portal do GE, tem-se o seguinte resultado (Figura 26):

Figura 26: Grafo de Conhecimento de um cenário real.

A Figura 26 mostra o grafo gerado em uma notícia publicada no portal do Globo Esporte, no *blog* especializado no clube Atlético Mineiro, disponível para acesso no *link* (https://ge.globo.com/futebol/times/atletico-mg/noticia/atletico-mg-x-chapecoense-com-covid-19-nacho-fernandez-nathan-e-outros-tres-jogadores-viram-baixas-de-ultima-hora-no-

galo.ghtml). No artigo são encontradas informações sobre integrantes do Atlético-MG que foram devidamente relacionados ao clube, embora haja um erro de inferência na entidade “Galo”, que não representa o nome e/ou apelido de um jogador, e sim, o apelido do clube Atlético-MG e na entidade “Igor Rabelo”, que se trata de um jornalista, a assertividade das inferências foi de 88,24%. Além disso, também são encontradas informações sobre outras entidades informações e complementares, como os dados da próxima partida que o Atlético-MG teve.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve por objetivo desenvolver uma ferramenta que automatizasse o preenchimento de um grafo de conhecimento aplicado a artigos de um portal de notícias esportivas. Para isso, o trabalho foi dividido entre extração dos dados, processamento e alimentação do grafo de conhecimento.

Na extração dos dados, o Scrapy demonstrou ser uma ferramenta eficaz quanto a extração dos artigos, suportando a entrada de diversos *links*, extraindo cada um dos dados indicados em seus parâmetros por meio de *tags*. Além disso, permitia realizar o tratamento dos dados antes de armazenamento, evitando o desenvolvimento de algoritmos externos para esta função.

O processamento dos dados pode ser dividido em dois pontos principais: análise dos dados por meio do Linguakit e o processamento, propriamente disto, destes dados.

No geral, o Linguakit mostrou ser uma boa alternativa para a tokenização e etiquetagem do texto de acordo com suas classes gramaticais, embora ainda apresente limitações quanto a inferência correta das etiquetas em determinados contextos.

Essa falha é evidente quando há ambiguações nas entidades, como no caso da entidade "Fortaleza" que representa tanto uma localidade, quanto uma organização, o Linguakit, por vezes, etiquetava a entidade "Fortaleza" como uma localidade, porém, no contexto que a entidade estava inserida, sua representação correta era de uma organização.

Outro ponto importante, envolve os apelidos das entidades, onde "Flamengo" pode ser representado como "Mengão", Palmeiras como "Porco" ou Gabriel Barbosa como "Gabigol", por exemplo. Quando haviam apelidos inclusos no artigo, o Linguakit não conseguia aferir da maneira correta qual a etiqueta mais adequada para cada situação, uma vez que "Porco" era, por vezes, etiquetado como "NP00SP0" (pessoa), quando deveria ser "NP00O00" (organização). Embora, em alguns casos a etiquetagem fosse totalmente assertiva, como pode ser visto na seção 4.6.1, Regra 2.

O módulo de processamento foi desenvolvido com o objetivo de organizar as informações extraídas pelo Linguakit. Durante esta organização, foi necessária a extração dos nomes dos times e estádios, a fim de aumentar a assertividade dos resultados, diluindo parte dos problemas causados pelo mau etiquetamento. Estes dados extraídos eram comparados com as entidades localizadas pelo Linguakit e por meio de averiguações era feito a substituição ou manutenção da etiqueta da entidade.

Ao etiquetar as entidades, o Linguakit as reconhecia como sendo distintas umas das outras, uma vez que, uma mesma entidade era etiquetada de maneiras diferentes ao longo do artigo. Este problema era transferido para o processamento dos dados que identificava a maior

ocorrência de uma etiqueta para uma mesma entidade e estabelecia um etiquetamento padrão. Porém, quando se tratava de apelidos, como Gabriel “Gabigol” Barbosa, este problema persistia, onde Gabriel Barbosa e Gabigol eram considerados como entidades distintas e sem ligação. Sendo esta questão, um problema a ser resolvido em trabalhos futuros.

Além dos apelidos, as entidades compostas, apresentam outro problema a ser resolvido, como o apresentado na seção 4.5.1, onde é exemplificado o caso dos jogadores do Flamengo, Diego Ribas e Diego Alves, onde o Ribas é frequentemente mencionado apenas como Diego, porém, se no artigo os dois jogadores forem mencionados e ao aplicar a Regra 1 da seção 4.5, a entidade Diego poderá ser entendida como referência ao Diego Alves e não como uma entidade própria.

Para trabalhos futuros, além da correção das problemáticas mencionadas acima, novas funcionalidades podem ser adicionadas na ferramenta, como por exemplo: a busca por novos padrões que incidem na descoberta de mais informações sobre as entidades, encorpando o resultado final apresentado no grafo; a ampliação da extração dos dados presentes no artigo, como o conteúdo que não se encontra no texto corrido, sendo um exemplo disto, as citações dos entrevistados pelo repórter; por fim, a aplicação da ferramenta em múltiplos artigos de maneira simultânea, visto que o trabalho, como se encontra atualmente, funciona de maneira eficiente para um artigo por vez, se mais de um artigo for utilizado na ferramenta, haverá duplicações no de entidades no grafo, ou seja, se em um artigo estiver citando sobre o Flamengo e no segundo artigo, também, haverá dois nós “Flamengo” no grafo.

REFERÊNCIAS

AMARAL, Daniela Oliveira Ferreira do. **Reconhecimento de Entidades Nomeadas na Área da Geologia**: Bacias Sedimentares Brasileiras. Orientador: Profa. Renata Vieira. 2017. 107 p. Dissertação (Doutorado) - Faculdade de Informática Programa de Pós-Graduação em Ciência da Computação Doutorado em Ciência da Computação, Porto Alegre, Brasil, 2017. Disponível em: http://tede2.pucrs.br/tede2/bitstream/tede/8035/2/DANIELA_OLIVEIRA_FERREIRA_DO_AMARAL_TES.pdf. Acesso em: 27 jan. 2021.

ANDERSON, D. **Kanban:** Successful Evolutionary Change for your Technology Business. Blue Hole Press. ISBN 0-9845214-0-2, 2010·

BORGES, Luiz Eduardo. **Python para Desenvolvedores**. 1. ed. São Paulo, Brasil: Novatec Editora Ltda., 2014. 315 p. ISBN 978-85-7522-405-2.

CALZOLARI, N. et al, (ed.). Introdução à iniciativa EAGLES. In: CALZOLARI, N.; MCNAUGHT, J.; ZAMPOLLI, A. (ed.). **EAGLES**: Editor‘s Introduction. Pisa, Itália, 1996. Disponível em: http://www.ilc.cnr.it/EAGLES96/edintro/node6.html#SECTION00040000000000000000. Acesso em: 7 jul. 2021.

COPPIN, Ben. **Inteligência Artificial**. Rio de Janeiro: Ltc, 2013. 636 p.

CULICOVER, Peter. Overview. *In*: CULICOVER, Peter W. **Natural Language Syntax**. Nova York, EUA: Oxford University Press, 2009. cap. 1, p. 1-10. ISBN 978-0-19-923017-4 (Hbk.).

DALE, Rober. **Classical Approaches to Natural Language Processing.** In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing. 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 1. p. 3-7.

HIPPISLEY, Andrew. **Lexical Analysis**. In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing. 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 2. p. 31-58.

GODDARD, Cliff; SCHALLEY, Andrea C. **Semantic Analysis**. In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing. 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 2. p. 31-58.

GOLDMAN, Alfredo; KON, Fabio; JUNIOR, Francisco Pereira; POLATO, Ivanilton; PEREIRA, Rosangela de Fátima. Capítulo 3: **Apache Hadoop**: Conceitos teóricos e práticos, evolução e novas possibilidades. In: Alberto Ferreira de Souza; Roberto Marcondes Cesar Junior; Ranata Galante. (Org.). XXXI Jornadas de atualizações em informática. 1ed. Porto Alegre: SBC, 2012, v., p. 88-136.

HÉGARET, Philippe Le. **What is the Document Object Model?** W3C, 7 abr. 2004. Disponível em: https://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/introduction.html. Acesso em: 21 abr. 2021.

KOUZIS-LOUKAS, Dimitrios. **Leaning Scrapy**: Learn the art of efficient web scraping and crawling with Python. [S. l.]: Packt Publishing, 2016. 243 p. ISBN 978-1-78439-978-8.

LEVISON, Stephen C. **Pragmática**. 1. ed. São Paulo: WMF Martins Fontes, 2007. 576 p. ISBN 9788533623323.

LONGEN, Andrei Silveira. **O Que é uma Query em um Banco de Dados?**. Estados Unidos: Hostinger, 2021. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-query. Acesso em: 27 maio 2021.

LOPES, Dener Cesar Ferreira. **Grafos de Conhecimento**: Perspectivas e Desafios para a Organização e Representação do Conhecimento. Orientador: Prof. Dr. Rogério Ap. Sá Ramalho. 2020. 71 p. Dissertação (Mestrado) - Universidade Federal de São Carlos, São Carlos, 2020. Disponível em: https://repositorio.ufscar.br/bitstream/handle/ufscar/13055/GRAFOS%20DE%20CONHECIMENTO%20PERSPECTIVAS%20E%20DESAFIOS%20PARA%20A%20ORGANIZA%c3%87%c3%83O%20E%20REPRESENTA%c3%87%c3%83O%20DO%20CONHECIMENTO_1.pdf?sequence=3&isAllowed=y. Acesso em: 26 set. 2020.

LUTZ, Mark. A Python Q&A Session. *In*: LUTZ, Mark. **Learning Python**. 5. ed. California, EUA: O'reilly, 2013. cap. 1, p. 3-26. ISBN 978-1-119-35573-9.

LYONS, John. Language, speech and writing. *In*: LYONS, John. **Natural Language and Universal Grammar**: Essays in Linguistic Theory. Cambridge, Inglaterra: Cambridge University Press, 1991. v. 1, cap. 1, p. 1-11. ISBN 9781139165877.

NAGAO, Makto. **Scientist1**: MAKOTO NAGAO (Roll No:6). Disponível em: https://lbsitbytes2010.wordpress.com/2013/03/27/scientist1-makoto-nagao-roll-no6/. Acesso em: 10 nov 2020

NEO4J (Malmö, Sweden). **O que é Neo4j?** Estados Unidos: Neo4j, 2020. As referências foram retiradas dos tópicos: "O que é Neo4j" e "A vantagem do Native Graph". Disponível em: https://neo4j.com/. Acesso em: 24 set. 2020.

NEO4J (Malmö, Sweden). **Cypher Query Language**. [S. l.], 2021. Disponível em: https://neo4j.com/developer/cypher/. Acesso em: 9 jul. 2021.

PALMER, David D. **Text Preprocessing**. In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing. 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 2. p. 9-30.

PAN, Jeff Z.; VETERE, Guido; PEREZ, Jose Manuel Gomez-; WU, Honghan. **Exploiting Linked Data and Knowledge Graphs in Large Organizations**. Cham, Suiça: Springer, 2017. 265 p. ISBN 978-3-319-45654-6. *E-book*.

PAULHEIM, Heiko. **Knowledge graph refinement**: A survey of approaches and evaluation methods. Semantic Web Journal, [S. l.], ano 2017, v. 8, n. 3, p. 486-508, 6 dez. 2016. DOI 10.3233/SW-160218. Disponível em: https://www.researchgate.net/publication/311479070_Knowledge_graph_refinement_A_survey_of_approaches_and_evaluation_methods. Acesso em: 18 nov. 2020.

RODRIGUES, Alexander. **Grafo do Conhecimento**. [*S. l.*], [201-?]. Disponível em: https://semantico.com.br/dicionario-seo/grafo-do-conhecimento/. Acesso em: 1 out. 2020.

ROSA, João Luís Garcia. Processamento de Linguagem Natural. In: ROSA, João Luís Garcia. **Fundamentos da Inteligência Artificial**. Rio de Janeiro: Livros Técnicos e Científicos Editora, 2011. cap. 8, p. 136-171. ISBN 978-85-216-0593-5.

ROSSUM, Guido van. **Foreword for "Programming Python" (1st ed.)**. 1. ed. Virginia, Estados Unidos: Python, 1996. Disponível em: https://www.python.org/doc/essays/foreword/. Acesso em: 24 set. 2020.

SANCHES, Matheus Ferraroni. **Processamento e Entendimento de Linguagem Natural no Gerenciamento de Emergências para Obtenção de Consciência Situacional**. Orientador: Prof. Dr. Leonardo Castro Botega. 2017. 75 p. Monografia (Bacharelado) - Centro Universitário Eurípedes de Marília, Marília - SP, 2017. Disponível em: https://aberto.univem.edu.br/bitstream/handle/11077/1662/Matheus%20Ferraroni%20Sanches.pdf?sequence=1&isAllowed=y. Acesso em: 10 nov. 2020.

SANTOS, Diana. **Introdução ao processamento de linguagem natural através das aplicações**. Caminho, Lisboa, Portugal, p. 229-259, 2001. Disponível em: https://www.linguateca.pt/Diana/download/Santos2001Aplicacoes.pdf. Acesso em: 22 set. 2020

SHAO, Bin; WANG, Haixun; LI, Yatao. **Trinity**: A Distributed Graph Engine on a Memory Cloud. Proceedings of the ACM SIGMOD International Conference on Management of Data, [s. l.], p. 505-516, 2013. DOI 10.1145/2463676.2467799. Disponível em: https://www.semanticscholar.org/paper/Trinity%3A-a-distributed-graph-engine-on-a-memory-Shao-Wang/8d0656c7e89894dfd1be5dda7c6bf3f7b5ab8616. Acesso em: 29 nov. 2020.

SILVA, Daniela Filipa Macedo Braga Moreira da. **Algoritmos de Processamento da Linguagem Natural para Sistemas de Conversão Texto-Fala em Português**. 187 p. Dissertação (Doutorado) - Facultade de Filoloxía da Universidade da Coruña, [*S. l.*], 2008. Disponível em: https://ruc.udc.es/dspace/bitstream/handle/2183/1011/Braga_DanielaFilipaMacedoMoreiradaSilva_TD_2008.pdf. Acesso em: 23 set. 2020.

SINGHAL, Amit. **Introducing the Knowledge Graph**: things, not strings. 16 maio 2012. Disponível em: https://www.blog.google/products/search/introducing-knowledge-graph-things-not/. Acesso em: 23 set. 2020.

TOFFLER, Alvin. Terceira Onda. **Unisinos**, [*s. l.*], 1995. Disponível em: http://www.projeto.unisinos.br/humanismo/antropos/Terceira_Onda.pdf. Acesso em: 22 set. 2020.

VIEIRA, Renata; LIMA, Vera Lucia Strube. **Linguística computacional**: princípios e aplicações. In: IX Escola de Informática da SBC-Sul. Luciana Nedel (Ed.) Passo Fundo, Maringá, São José. SBC-Sul, 2001. Disponível em: https://www.inf.pucrs.br/linatural/Recursos/jaia-2001.pdf. Acesso em: 20 nov. 2020.

VIEIRA, Renata; LOPES, Luciene. **Processamento de Linguagem Natural e o Tratamento Computacional de Linguagens Científicas**. In: PERNA, Cristina Becker Lopes; DELGADO, Heloisa Orsi Koch; FINATTO, Maria José Bocorny. Linguagens Especializadas em Corpora: Modo de dizer e interfaces de pesquisa. Porto Alegre: EDIPUCRS, 2010. p. 183-201. ISBN 978-85-397-0024-0. Disponível em: https://bibliodigital.unijui.edu.br:8443/xmlui/bitstream/handle/123456789/1496/Linguagens%20especializadas%20em%20corpora%20modos%20de%20dizer%20e%20interfaces%20de%20pesquisa%20.pdf?sequence=1&isAllowed=y. Acesso em: 20 nov. 2020.

VILARINHO, Sabrina. **Semântica**. [*S. l.*], [20--]. Disponível em: https://brasilescola.uol.com.br/portugues/semantica.htm. Acesso em: 2 out. 2020.

VISUAL STUDIO CODE (ed.). **Começando**. Estados Unidos, 2021. Disponível em: https://code.visualstudio.com/docs. Acesso em: 11 maio 2021.

YAN, Jihong; WANG, Chengyu; CHENG, Wenliang; GAO, Ming; ZHOU, Aoying. **A retrospective of knowledge graphs**. Frontiers Computer Science, Shanghai, p. 55-74, 26 set. 2016. DOI: https://doi.org/10.1007/s11704-016-5228-9. Disponível em: https://www.researchgate.net/publication/308691086_A_retrospective_of_knowledge_graphs. Acesso em: 23 set. 2020.

APÊNDICE

O quadro abaixo apresenta a lista de 100 *tokens* referentes aos nomes de jogadores utilizados no cálculo apresentado no tópico 4.5.2. A pontuação (“,”) e a conjunção (“e”) foram dispensadas, sendo apresentado somente os *tokens*.

APÊNDICE A - Quadro da etiquetagem dos nomes dos jogadores.

FORMA	LEMA	ETIQUETA
Diego_Alves	diego_alves	NP00SP0
Matheus_Cunha	matheus_cunha	NP00SP0
Gabriel_Batista	gabriel_batista	NP00SP0
César	césar	NP00SP0
Hugo_Nogueira	hugo_nogueira	NP00SP0
Joao_Fernando_Monteiro_Siqueira	joao_fernando_monteiro_siqueira	NP00SP0
Jose_Italo_Monteiro_Fidelis_Dos_Santos	jose_italo_monteiro_fidelis_dos_santos	NP00SP0
Gustavo_Henrique	gustavo_henrique	NP00SP0
Rodinei	rodinei	NP00V00
Rodrigo_Caio	rodrigo_caio	NP00SP0
Léo_Pereira	léo_pereira	NP00SP0
René	rené	NP00SP0
Filipe_Luís	filipe_luís	NP00SP0
Otavio_Ataide_de_a_Silve	otavio_ataide_de_a_silve	NP00SP0
Diego_Damasceno	diego_damasceno	NP00SP0
Luan_Sales_Do_Nascimento	luan_sales_do_nascimento	NP00SP0
Bruno_Viana	bruno_viana	NP00SP0
Mateuzinho	mateuzinho	NP00V00
Ramon_Ramos_Lima	ramon_ramos_lima	NP00SP0
Gabriel_Noga	gabriel_noga	NP00SP0
Mauricio_Isla	mauricio_isla	NP00SP0
Willian_Arão	willian_arão	NP00SP0

Éverton_Ribeiro	éverton_ribeiro	NP00V00
Diego	diego	NP00SP0
Giorgian_de_Arrascaeta	giorgian_de_arrascaeta	NP00V00
Yuri	yuri	NP00SP0
Hugo_Moura	hugo_moura	NP00SP0
Fabrizio_Peralta	fabrizio_peralta	NP00SP0
Robert_Piris_Da_Motta	robert_piris_da_motta	NP00SP0
Max	max	NP00SP0
Thiago_Maia	thiago_maia	NP00SP0
João_Gomes	joão_gomes	NP00SP0
Daniel_Cabral	daniel_cabral	NP00SP0
Werton_De_Almeida_Rêgo	werton_de_almeida_rêgo	NP00SP0
Pedro_Machado	pedro_machado	NP00SP0
Gabriel	gabriel	NP00SP0
Vitinho	vitinho	NP00V00
Michael_Delgado	michael_delgado	NP00SP0
Lazaro	lazaro	NP00SP0
Pedro	pedro	NP00SP0
Lucas_Silva	lucas_silva	NP00SP0
Bruno_Henrique	bruno_henrique	NP00SP0
Thiago_Fernandes	thiago_fernandes	NP00SP0
Gabriel_De_Sousa_Barros	gabriel_de_sousa_barros	NP00SP0
Matheus_França	matheus_França	NP00SP0
Rodrigo_Muniz	rodrigo_muniz	NP00SP0
Mateus_Lima	mateus_lima	NP00SP0
Ryan	ryan	NP00G00
Guilherme_Arana	guilherme_arana	NP00SP0
Nathan_Da_Silva	nathan_da_silva	NP00SP0

Júnior_Alonso	júnior_alonso	NP00SP0
Réver	réver	NP00SP0
Dodô	dodô	NP00V00
Carlos_Daniel	carlos_daniel	NP00SP0
Igor_Rabello	igor_rabello	NP00SP0
Leonardo_Simoni	leonardo_simoni	NP00SP0
Micael	micael	NP00SP0
Rómulo_Otero	rómulo_otero	NP00SP0
Federico_Zaracho	federico_zaracho	NP00SP0
Dylan_Borrero	dylan_borrero	NP00SP0
Hyoran	hyoran	NP00V00
Alan_Franco	alan_franco	NP00SP0
Nathan	nathan	NP00SP0
Daniel_Borges	daniel_borges	NP00SP0
Edinho	edinho	NP00V00
Ignacio_Fernández	ignacio_fernández	NP00SP0
Calebe	calebe	NP00V00
Allan	allan	NP00SP0
Gabriel_Souza	gabriel_souza	NP00SP0
Tchê_Tchê	tchê_tchê	NP00SP0
Marrony	marrony	NP00V00
Júlio_Cesar	júlio_cesar	NP00SP0
Neto	neto	NP00SP0
Rubens	rubens	NP00SP0
Echaporã	echaporã	NP00G00
Ruan_Nascimento_Dos_Santos	ruan_nascimento_dos_santos	NP00V00
Hulk	hulk	NP00SP0
Eduardo_Vargas	eduardo_vargas	NP00SP0

Keno	keno	NP00V00
Eduardo_Sasha	eduardo_sasha	NP00SP0
Jefferson_Savarino	jefferson_savarino	NP00SP0
Roberto_Fernández	roberto_fernández	NP00SP0
Diego_Cavalieri	diego_cavalieri	NP00SP0
Douglas_Borges	douglas_borges	NP00SP0
Diego_Terra_Loureiro	diego_terra_loureiro	NP00SP0
Andrew_Da_Silva_Ventura	andrew_da_silva_ventura	NP00SP0
Igo_Gabriel_Santos_Pereira	igo_gabriel_santos_pereira	NP00SP0
Cascardo	cascardo	NP00V00
Vitor_Marinho	vitor_marinho	NP00SP0
Kanu	kanu	NP00SP0
Joel_Carli	joel_carli	NP00SP0
Gilvan	gilvan	NP00SP0
Jonathan_Lemos	jonathan_lemos	NP00SP0
Hugo	hugo	NP00SP0
Luis	luis	NP00SP0
Ricardinho	ricardinho	NP00V00
Diego	diego	NP00SP0
Felipe	felipe	NP00SP0
Romildo_Del_Piage_De_Souza	romildo_del_piage_de_souza	NP00SP0
Barreto	barreto	NP00G00

APÊNDICE B - Quadro com os artigos utilizados nas estatísticas.

TÍTULO	URL
FLAMENGO	
Análise: Flamengo atinge seu grau mais alto de maturidade na temporada; segundo gol simboliza	https://ge.globo.com/futebol/times/flamengo/noticia/analise-flamengo-atinge-seu-grau-mais-alto-de-maturidade-na-temporada-segundo-gol-simboliza.ghtml
Tribunal Pleno do STJD nega pedido do Flamengo de paralisação do Brasileiro	https://ge.globo.com/futebol/times/flamengo/noticia/tribunal-pleno-do-stjd-nega-pedido-do-flamengo-de-paralisacao-do-campeonato-brasileiro.ghtml
Após período de quarentena depois do positivo para Covid, Ceni volta ao comando do Flamengo	https://ge.globo.com/futebol/times/flamengo/noticia/apos-periodo-de-quarentena-depois-do-positivo-para-covid-ceni-volta-ao-comando-do-flamengo.ghtml
Atuações Flamengo: Vitinho, Michael, Matheuzinho e Gerson se destacam na vitória sobre o Coxa	https://ge.globo.com/futebol/times/flamengo/noticia/atuacoes-flamengo-vitinho-michael-matheuzinho-e-gerson-se-destacam-na-vitoria-sobre-o-coxa.ghtml
Mauricinho passa o bastão para Ceni e elogia o Flamengo: "Apresentações que nos deixam felizes	https://ge.globo.com/futebol/times/flamengo/noticia/mauricinho-passa-o-bastao-para-ceni-e-elogia-o-flamengo-apresentacoes-que-nos-deixam-felizes.ghtml
Destaque na vitória do Flamengo, Gerson evita discurso de adeus: "Ainda tenho um dever a cumprir"	https://ge.globo.com/futebol/times/flamengo/noticia/destaque-na-vitoria-do-flamengo-gerson-evita-discurso-de-despedida-ainda-tenho-um-dever-a-cumprir.ghtml
Escalação do Flamengo: Rogério Ceni é a novidade, e time será o mesmo do jogo contra o América-MG	https://ge.globo.com/futebol/times/flamengo/noticia/flamengo-x-bragantino-rogerio-ceni-e-a-novidade-e-repete-time-que-venceu-o-america-mg.ghtml
Inspirado em Gabigol e Pedro, e no embalo de Muniz, Ryan Luka é a cara nova entre os "9" do Flamengo	https://ge.globo.com/futebol/times/flamengo/noticia/inspirado-em-gabigol-e-pedro-e-no-embalo-de-muniz-ryan-luka-e-a-cara-nova-entre-os-9-do-flamengo.ghtml
Pedro testa negativo para Covid-19, mas ainda desfalca o Flamengo contra o Bragantino	https://ge.globo.com/futebol/times/flamengo/noticia/pedro-testa-negativo-para-covid-19-mas-ainda-desfalca-o-flamengo-contra-o-bragantino.ghtml
Chamou a responsabilidade: sem medalhões,	https://ge.globo.com/futebol/times/flamengo/

Bruno Henrique melhora seu desempenho	noticia/chamou-a-responsabilidade-sem-medalhoes-bruno-henrique-melhora-seu-desempenho.ghtml
PALMEIRAS	
Palmeiras faz corrente por título e não descarta nem Lucas Lima: "Ninguém fica para trás"	https://ge.globo.com/futebol/times/palmeiras/noticia/palmeiras-faz-corrente-por-titulo-e-nao-descarta-nem-lucas-lima-ninguem-fica-para-tras.ghtml
Gustavo Scarpa e Raphael Veiga brincam sobre autoria do segundo gol do Palmeiras contra o Bahia	https://ge.globo.com/futebol/times/palmeiras/noticia/noticias-palmeiras-gol-scarpa-veiga.ghtml
Análise: Palmeiras é premiado por luta, mas precisa levar menos gols para seguir brigando no topo	https://ge.globo.com/futebol/times/palmeiras/noticia/analise-palmeiras-e-premiado-por-luta-mas-precisa-levar-menos-gols-para-seguir-brigando-no-topo.ghtml
Abel quer diminuir reclamações no banco do Palmeiras: "Tenho tentado me portar melhor"	https://ge.globo.com/futebol/times/palmeiras/noticia/noticias-palmeiras-abel-quer-diminuir-reclamacoes-banco-reservas.ghtml
Abel diz ter relação extraordinária com a diretoria: "O Palmeiras é o sonho de qualquer treinador"	https://ge.globo.com/futebol/times/palmeiras/noticia/abel-diz-ter-relacao-extraordinaria-com-a-diretoria-o-palmeiras-e-o-sonho-de-qualquer-treinador.ghtml
Atuações do Palmeiras: Scarpa faz golaço, dá assistência e decide em virada no fim	https://ge.globo.com/futebol/times/palmeiras/noticia/atuacoes-do-palmeiras-scarpa-faz-golaco-da-duas-assistencias-e-decide-em-virada-no-fim.ghtml
Paz nos bastidores, vaga no G-4 e confiança: o que vale para o Palmeiras o jogo contra o Bahia	https://ge.globo.com/futebol/times/palmeiras/noticia/noticias-palmeiras-jogo-bahia-bastidores-vaga-g4-confianca.ghtml
Palmeiras anuncia renovações de contrato com Willian, Zé Rafael, Raphael Veiga e Rony	https://ge.globo.com/futebol/times/palmeiras/noticia/noticias-palmeiras-renovacao-renovacoes-contrato-willian-ze-rafael-veiga-rony.ghtml
Escalação do Palmeiras: Luan e Danilo estão à disposição de Abel Ferreira para duelo com o Bahia	https://ge.globo.com/futebol/times/palmeiras/noticia/escalacao-do-palmeiras-luan-e-danilo-estao-a-disposicao-de-abel-ferreira-para-duelo-com-o-bahia.ghtml
Willian se aproxima dos 100 jogos na arena do Palmeiras: "Momentos inesquecíveis"	https://ge.globo.com/futebol/times/palmeiras/noticia/noticias-palmeiras-willian-100-jogos-allianz-parque.ghtml

FLUMINENSE	
Atuações do Fluminense: Cazares faz valer a "Lei do Ex", e Abel Hernández é o pior em campo	https://ge.globo.com/futebol/times/fluminense/noticia/atuacoes-do-fluminense-cazares-faz-valer-a-lei-do-ex-e-abel-hernandez-e-o-pior-em-campo.ghtml
Cazares celebra primeiro gol pelo Fluminense, e explica falta de comemoração: "Tem que respeitar"	https://ge.globo.com/futebol/times/fluminense/noticia/cazares-fica-feliz-ao-marcar-primeiro-gol-pelo-fluminense-e-explica-motivo-de-nao-comemorar.ghtml
Fred e Nenê são poupados no Fluminense e não enfrentam o Corinthians; veja relacionados	https://ge.globo.com/futebol/times/fluminense/noticia/fred-e-nene-sao-poupados-no-fluminense-e-nao-enfrentam-o-corinthians-veja-relacionados.ghtml
Escalação do Fluminense: Caio Paulista não treina e também vira desfalque contra o Corinthians	https://ge.globo.com/futebol/times/fluminense/noticia/escalacao-do-fluminense-caio-paulista-nao-treina-e-tambem-vira-desfalque-contra-o-corinthians.ghtml
Hudson prepara corpo e mente no Fluminense por volta e não pensa no futuro: "Um passo de cada vez"	https://ge.globo.com/futebol/times/fluminense/noticia/hudson-prepara-corpo-e-mente-no-fluminense-por-volta-e-nao-pensa-no-futuro-um-passo-de-cada-vez.ghtml
Escalação do Fluminense: Samuel Xavier segue fora e deve ser única ausência contra o Corinthians	https://ge.globo.com/futebol/times/fluminense/noticia/escalacao-do-fluminense-samuel-xavier-segue-fora-e-deve-ser-unica-ausencia-contra-o-corinthians.ghtml
Neto de zagueiro que marcou Pelé, sueco de 21 anos passa por intercâmbio no Fluminense	https://ge.globo.com/futebol/times/fluminense/noticia/neto-de-zagueiro-que-marcou-pele-sueco-de-21-anos-passa-por-intercambio-no-fluminense.ghtml
Nino renova com Fluminense até fim de 2024: "Pretendo retribuir essa confiança dentro de campo"	https://ge.globo.com/futebol/times/fluminense/noticia/fluminense-renova-contrato-com-nino-e-prorroga-vinculo-de-matheus-ferraz.ghtml
Após percorrer mais de 30 mil km, Fluminense inicia sequência de quatro jogos no Rio de Janeiro	https://ge.globo.com/futebol/times/fluminense/noticia/apos-percorrer-mais-de-30-mil-km-fluminense-inicia-sequencia-de-quatro-jogos-no-rio-de-janeiro.ghtml
Análise: mexidas tardias e falta de repertório custam caro para um já previsível Fluminense	https://ge.globo.com/futebol/times/fluminense/noticia/analise-mexidas-tardias-e-falta-de-repertorio-custam-caro-para-um-ja-previsivel-fluminense.ghtml

ATLÉTICO-MG	
Sobrinhos de Hulk participam de processo para avaliação nas categorias de base do Atlético-MG	https://ge.globo.com/futebol/times/atletico-mg/noticia/sobrinhos-de-hulk-participam-de-processo-para-avaliacao-nas-categorias-de-base-do-atletico-mg.ghtml
Escalação do Atlético-MG: Dodô vai a campo e Galo não divulga os relacionados contra o Corinthians	https://ge.globo.com/futebol/times/atletico-mg/noticia/escalacao-do-atletico-mg-dodo-vai-a-campo-e-galo-nao-divulga-os-relacionados-contra-o-corinthians.ghtml
Perto do 30º jogo pelo Atlético-MG em 2021, Everson reduz média de gols sofridos para a metade	https://ge.globo.com/futebol/times/atletico-mg/noticia/perto-do-30o-jogo-pelo-atletico-mg-em-2021-everson-reduz-media-de-gols-sofridos-para-a-metade.ghtml
Contra o Corinthians, Atlético-MG busca rara trinca de vitórias como visitante no Brasileirão	https://ge.globo.com/futebol/times/atletico-mg/noticia/contra-o-corinthians-atletico-mg-busca-rara-trinca-de-vitorias-como-visitante-no-brasileirao.ghtml
Com monitoramento, Atlético-MG mantém 23 atletas emprestados em 2021; veja situação de cada um	https://ge.globo.com/futebol/times/atletico-mg/noticia/com-monitoramento-atletico-mg-mantem-23-atletas-emprestados-em-2021-veja-situacao-de-cada-um.ghtml
Desfalque no Atlético-MG, lateral Guilherme Arana é titular pela seleção olímpica em amistoso	https://ge.globo.com/futebol/times/atletico-mg/noticia/desfalque-no-atletico-mg-lateral-guilherme-arana-e-titular-pela-selecao-olimpica-em-amistoso.ghtml
Escalação do Atlético-MG: Réver tem inflamação no cotovelo, e Dodô trata entorse na fisioterapia	https://ge.globo.com/futebol/times/atletico-mg/noticia/escalacao-do-atletico-mg-rever-tem-inflamacao-no-cotovelo-e-dodo-trata-entorse-na-fisioterapia.ghtml
Primeiro gol, Bombonera, e espaço no time: Dylan vive semana de metas cumpridas no Atlético-MG	https://ge.globo.com/futebol/times/atletico-mg/noticia/primeiro-gol-bombonera-e-espaco-no-time-dylan-vive-semana-de-metas-cumpridas-no-atletico-mg.ghtml
Vendas, rescisões e fim de vínculo: Atlético-MG alivia folha e gera economia de R$ 3,5 milhões/mês	https://ge.globo.com/futebol/times/atletico-mg/noticia/vendas-rescisoes-e-fim-de-vinculo-atletico-mg-alivia-folha-e-gera-economia-de-r-35-milhoesmes.ghtml
Cuca indica questão física e Atlético-MG tende a usar time alternativo diante do Corinthians	https://ge.globo.com/futebol/times/atletico-mg/noticia/cuca-indica-questao-fisica-e-atletico-mg-tende-a-usar-time-alternativo-diante-do-corinthians.ghtml

CORINTHIANS	
Escalação do Corinthians: Sylvinho relaciona zagueiro da base para jogo contra o Atlético-MG	https://ge.globo.com/futebol/times/corinthians/noticia/escalacao-do-corinthians-sylvinho-relaciona-zagueiro-da-base-para-jogo-contra-o-atletico-mg.ghtml
Corinthians trata contratação de Paulinho como improvável	https://ge.globo.com/futebol/times/corinthians/noticia/perto-de-giuliano-corinthians-trata-contratacao-de-paulinho-como-improvavel.ghtml
https://ge.globo.com/futebol/times/corinthians/noticia/e-o-renato-augusto-corinthians-insistira-na-contratacao-mesmo-apos-provavel-acerto-com-giuliano.ghtml	https://ge.globo.com/futebol/times/corinthians/noticia/e-o-renato-augusto-corinthians-insistira-na-contratacao-mesmo-apos-provavel-acerto-com-giuliano.ghtml
Corinthians quita um mês e diminui pendências em ajuda de custo de atletas da base	https://ge.globo.com/futebol/times/corinthians/noticia/noticias-corinthians-base-salarios-atraso.ghtml
Gustavo Mosquito, do Corinthians, revela outra perda por Covid na família	https://ge.globo.com/futebol/times/corinthians/noticia/noticias-corinthians-gustavo-mosquito-covid.ghtml
Corinthians cria setor para monitorar emprestados e terá Márcio Bittencourt como observador	https://ge.globo.com/futebol/times/corinthians/noticia/noticias-corinthians-emprestados-marcio-bittencourt.ghtml
Veja a provável escalação do Corinthians para enfrentar o Atlético-MG	https://ge.globo.com/futebol/times/corinthians/noticia/veja-a-provavel-escalacao-do-corinthians-para-enfrentar-o-atletico-mg.ghtml
Mantuan avança em recuperação em semana livre para treinos no Corinthians	https://ge.globo.com/futebol/times/corinthians/noticia/mantuan-avanca-em-recuperacao-em-semana-livre-para-treinos-no-corinthians.ghtml
Jogadores da base do Corinthians estão com quase três meses de salários atrasados	https://ge.globo.com/futebol/times/corinthians/noticia/noticias-corinthians-base-salarios-atrasados.ghtml
Escalação do Corinthians: time finaliza preparação em Fortaleza e deve ter Cantillo em campo	https://ge.globo.com/futebol/times/corinthians/noticia/noticias-corinthians-escalacao-jogo-fortaleza-cantillo-treina.ghtml

ANEXO

ANEXO A - Quadro da classe gramatical - adjetivos.

ADJETIVOS			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Adjetivo	A
2	Tipo	Qualificativo	Q
		Ordinal	O
		-	0
3	Grau	-	0
		Aumentativo	A
		Diminutivo	C
		Superlativo	S
4	Gênero	Masculino	M
		Feminino	F
		Comum	C
5	Número	Singular	S
		Plural	P
		Invariável	N
6	Função	-	0
		Particípio	P

ANEXO B - Quadro da classe gramatical - advérbios.

ADVÉRBIOS			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Advérbio	R
2	Tipo	Principal	G
		Negativo	N

ANEXO C - Quadro da classe gramatical - determinantes.

DETERMINANTES			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Determinante	D
2	Tipo	Demonstrativo	D
		Possessivo	P
		Interrogativo	T
		Exclamativo	E
		Indefinido	I
		Artigo	A
3	Pessoa	Primeira	1
		Segunda	2
		Terceira	3
4	Gênero	Masculino	M
		Feminino	F

		Comum	C
		Neutro	N
5	Número	Singular	S
		Plural	P
		Invariável	N
6	Suporte	Singular	S
		Plural	P

ANEXO D - Quadro da classe gramatical - substantivos.

SUBSTANTIVOS			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Substantivo	N
2	Tipo	Comum	C
		Próprio	P
3	Gênero	Masculino	M
		Feminino	F
		Comum	C
4	Número	Singular	P
		Plural	S
		Invariável	N
5	Classificação Semântica	Pessoa	SP

		Lugar	G0
		Organização	O0
		Variados	V0
6	Grau	Aumentativo	A
		Diminutivo	D

ANEXO E - Quadro da classe gramatical - verbos.

VERBOS			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Verbo	V
2	Tipo	Principal	M
		Auxiliar	A
		Semi Auxiliar	S
3	Modo	Indicativo	I
		Subjuntivo	S
		Imperativo	M
		Infinitivo	N
		Gerúndio	G
		Particípio	P
4	Tempo	Presente	P
		Passado	S

		Pretérito Imperfeito	I
		Futuro	F
		Futuro do Pretérito	C
		-	0
5	Pessoa	Primeira	1
		Segunda	2
		Terceira	3
6	Número	Singular	S
		Plural	P
7	Gênero	Masculino	M
		Feminino	F

ANEXO F - Quadro da classe gramatical - pronomes.

PRONOMES			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Pronome	P
2	Tipo	Pessoal	P
		Demonstrativo	D
		Possessivo	X
		Indefinido	I
		Interrogativo	T

		Relativo	R
		Exclamativo	E
3	Pessoa	Primeira	1
		Segunda	2
		Terceira	3
4	Gênero	Masculino	M
		Feminino	F
		Comum	C
		Neutro	N
5	Pessoa	Primeira	1
		Segunda	2
		Terceira	3
6	Número	Singular	S
		Plural	P
		Invariável	N
6	Caso	Nominativo	N
		Acusativo	A
		Dativo	D
		Oblíquo	O
7	Suporte	Singular	S
		Plural	P

8	Polidez	Polite	P

ANEXO G - Quadro da classe gramatical - conjunções.

CONJUNÇÕES			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Conjunção	C
2	Tipo	Coordenada	C
		Subordinada	S

ANEXO H - Quadro da classe gramatical - Interjeição

INTERJEIÇÕES			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Interjeição	I

ANEXO I - Quadro da classe gramatical - preposições.

PREPOSIÇÕES			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Adposição	S
2	Tipo	Preposição	P
3	Forma	Simples	S
		Contraído	C
6	Gênero	Masculino	M

7	Número	Singular	S

ANEXO J - Quadro da classe gramatical - sinais de pontuação.

SINAIS DE PONTUAÇÃO			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Pontuação	F

ANEXO K - Quadro da classe gramatical - cifras e numerais.

VERBOS			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Cifra	Z
2	Tipo	Fração	d
		Moeda	m
		Porcentagem	p
		Unidade	u

ANEXO M - Quadro da classe gramatical - datas e horas.

DATAS E HORAS			
Posição	**Atributo**	**Valor**	**Código**
1	Categoria	Data/Hora	W