Universidade Federal do Rio de Janeiro
Engenharia de Computação e Informação
Escola Politécnica
COS471 - Computadores e Sociedade

Relatório do Projeto Final - Software para busca de palavras-chave no site de candidaturas

Grupo	Arthur Monteiro, Carolina Falcão e Guilherme En Shih Hu
Professor	Luiz Arthur
Facilitador	Eduardo Diniz
Horário	Ter e Qui - 13:00-15:00

Rio de Janeiro, 16 de dezembro de 2024

Conteúdo

1 Introdução

O presente relatório documenta o desenvolvimento e a implementação do projeto de criação de um software que busca palavras-chaves no site de candidaturas do TSE, realizado para o trabalho final da disciplina de Computadores e Sociedade - COS471. Dentre as habilidades e conhecimentos propostos para o aprendizado nessa atividade estão o Web Scraping, o acesso a PDFs e o salvamento de dados em planilhas por meio de código em linguagem Python.

O projeto surgiu de uma ideia do professor da FGV, referência no tema de moedas sociais, Eduardo Diniz, que já havia realizado uma busca manual de propostas relacionadas à moeda social nas candidaturas à prefeitura em 2020. Dessa forma, esse projeto da disciplina de Computadores e Sociedade possibilita que, por meio de software, essa busca seja automatizada. Assim surgiu o nosso projeto de uma automação de busca de propostas de governo que citem moeda social ou correlatos nas candidaturas à prefeitura de 2024.

Segue abaixo o link para o repositório com o programa da automação e as planilhas com os resultados: https://github.com/guilherme-hu/Projeto-CompSoc/tree/main

2 Guia rápido para uso

O passo a passo detalhado está descrito dentro do arquivo "README"no repositório no GitHub, mas de um modo geral, possui o seguinte algoritmo:

1. Instalar o Python pelo site https://www.python.org/
2. Baixar o código do projeto disponível no link https://github.com/guilherme-hu/Projeto-CompSoc/tree/main
3. Instalar as bibliotecas necessárias, descritas no arquivo "requirements"no repositório GitHub. Para isso, basta escrever "pip install -r requirements.txt"no terminal do computador, já acessado o diretório do projeto
4. Executar o código "main"no terminal do computador

Vale notar que o código só funciona para o Sistema Operacional Windows. Isso ocorre em função das bibliotecas utilizadas na implementação do projeto, que não funcionam direito em SOs que não sejam o Windows.

3 Implementação

A implementação do projeto foi feita baseada no Web Scraping, processo de extração dados de sites da web de forma automatizada, de forma que são coletadas informações publicamente disponíveis na internet e organizadas de maneira estruturada, facilitando sua análise. Desse modo, a partir do site do TSE, que apresenta os dados e propostas dos candidatos que participaram das eleições municipais de cada estado e em cada ano, seria possível automatizar a coleta relacionadas às candidaturas.

A linguagem escolhida para a implementação do software foi o Python, já que ela é amplamente utilizada em projetos de web scraping devido à sua simplicidade, ampla gama de bibliotecas específicas para essa tarefa e facilidade de integração com ferramentas de manipulação e análise de dados. Nesse aspecto, foram utilizadas no código, por exemplo, as bibliotecas Selenium, para o acesso à página Web do site do TSE; PyPDF2, para leitura e análise dos PDFs que contém as propostas de cada candidato; unicode para uniformização dos textos presentes nos textos nos PDFs; e o Pandas e openpyxl para salvar os resultados encontrados em uma planilha csv e Excel, respectivamente.

O código foi baseado no seguinte processo para pegar os dados de interesse: a partir do home do site, é preciso entrar na seção que apresenta as regiões e seus respectivos estados do Brasil, de modo a selecionar a opção "Candidatos". A princípio, em nosso software, só são coletados apenas dados referentes ao estado do Rio de Janeiro, visto que a coleta para uma escala nacional requer muitas iterações a mais e exige uma manutenção ainda mais difícil em caso de atualização do site analisado (quanto mais automatizado for o processo de busca, maior a chance do código não executar corretamente em caso de uma possível atualização do site), assim, o estado "Rio de Janeiro"deve ser selecionado nessa parte. A partir disso, deve ser feita uma iteração para cada município, que seleciona o cargo prefeito e em seguida verifica todos os candidatos que participaram da eleição desse município. Para cada um desses candidatos, é acessado o PDF disponibilizado na parte de "Propostas", contendo as propostas de governo que essa pessoa possui, e analisado seu texto, verificando se há ocorrências das palavras-chaves buscadas. Dessa forma, temos o seguinte pseudo-algoritmo para esse projeto:

Algorithm 1 Busca de palavras-chaves no site de candidaturas do TSE

Inicializar variáveis, como a lista de palavras-chaves de interesse L e a lista vazia que conterá os resultados encontrados R
Acessar a página sobre as eleições municipais de 2024 no site do TSE
Acessar a região "Sudeste"
Acessar o estado "Rio de Janeiro", selecionando a parte "Candidaturas"
for cada municipio m do estado selecionado **do**
    Escolher a opção "Prefeito"no cargo a ser pesquisado
    Apertar para pesquisar
    for cada candidato c do município m **do**
        Acessar o perfil de c
        Acessar a aba de "Propostas"desse perfil de c
        Baixar o PDF que contém as propostas de governo de c
        Fazer a leitura desse PDF e procurar nele as palavras-chaves contidas em L, salvando as informações de c e as palavras encontradas em R
    end for
end for
Retornar R

Após a coleta dos dados requisitados, foi feita uma implementação de análise de informações também a partir da linguagem Python, de modo a gerar gráficos e o mapa com a descrição visual das informações juntadas. Tais implementações serão mais explicadas nas subseções a seguir.

3.1 Explicação direta do código - para desenvolvedores

O código criado é um script em Python que utiliza a biblioteca Selenium para automatizar a navegação em um site de divulgação de candidaturas eleitorais, extrair informações de PDFs baixados e salvar os resultados em diferentes formatos de arquivo.

Primeiramente, o script importa as bibliotecas necessárias, incluindo Selenium para automação de navegador, PyPDF2 para leitura de PDFs, pandas e openpyxl para manipulação de dados e unidecode para normalização de texto. São definidas duas listas de palavras-chave (keywords e keywords2) que serão usadas para buscar termos específicos nos PDFs baixados. A lista keywords apresenta palavras chaves relacionadas ao tema de moedas sociais, e a lista keyword2 apresenta palavras-chaves de temas sociais mais abrangentes.

As configurações do navegador Chrome são ajustadas para definir o diretório de download dos arquivos PDF e desativar pop-ups e prompts de download. O diretório de download é criado se não existir. O navegador Chrome é então iniciado com essas configurações.

O script navega até o site de divulgação de candidaturas eleitorais e interage com vários elementos da página para selecionar a eleição municipal de 2024 na região Sudeste, especificamente no estado do Rio de Janeiro. Em seguida, itera através dos municípios do estado, selecionando cada município e o cargo de prefeito.

Para cada município, o script clica no botão de pesquisa para obter a lista de candidatos. Em seguida, itera através da lista de candidatos, clicando em cada candidato para acessar sua página de detalhes. Na página de detalhes do candidato, o script clica em um elemento específico para acessar a seção de propostas e baixar o PDF correspondente.

Após o download do PDF, o script espera alguns segundos para garantir que o arquivo seja baixado completamente. Em seguida, verifica os PDFs baixados no diretório de download e extrai o texto de cada página do PDF. O texto extraído é normalizado para remover acentos e caracteres especiais.

O script então verifica se alguma das palavras-chave das listas keywords e keywords2 está presente no texto extraído do PDF. Se encontrar alguma correspondência, extrai informações adicionais da página do candidato, como nome, partido, município e situação. Essas informações, juntamente com as palavras-chave encontradas, são armazenadas em um dicionário e adicionadas a uma lista de resultados.

Após processar cada PDF, o script apaga o arquivo PDF do diretório de download para liberar espaço. Em seguida, retorna à página de lista de candidatos e repete o processo para o próximo candidato.

No bloco finally, o navegador é fechado e o diretório de download é limpo. Os resultados armazenados na lista são então salvos em três formatos diferentes: uma planilha Excel (resultados.xlsx), um arquivo CSV (resultados.csv) e um arquivo de texto (resultados.txt). Cada formato de arquivo contém as informações dos candidatos e as palavras-chave encontradas nos PDFs.

Por fim, uma mensagem é impressa no console indicando que os resultados foram salvos com sucesso. Este script automatiza a coleta e análise de informações de candidatos eleitorais, facilitando a extração de dados relevantes de documentos PDF e a organização desses dados em formatos de arquivo facilmente acessíveis.

3.2 Análise dos dados gerados pelo Web Scraping

Após a obtenção dos dados com Web Scraping e os dados em uma planilha CSV, ocorreu a análise de dados utilizando as bibliotecas: Pandas, para a manipulação de DataFrames obtidos a partir da planilha, Matplotlib, para a geração de gráficos, e Geopandas, para a geração do mapa do Estado do Rio de Janeiro de forma personalizada juntamente de dados geográficos disponíveis no site do IBGE.

Os DataFrames são "recortes"da planilha original, como os recortes de cidade, situação, partido e palavras chaves encontradas, e são a matéria-prima para a elaboração dos gráficos. Após alguns testes e feedbacks do facilitador Eduardo Diniz utilizamos gráficos de pizza e de barras.

A geração do mapa personalizado é um processo um pouco mais sofisticado com um pré-requisito: possuir os dados geográficos do IBGE para o estado do Rio de Janeiro, mais especificamente o arquivo shapefile da malha municipal do estado (https://www.ibge.gov.br/geociencias/organizacao-do-territorio/malhas-territoriais/15774-malhas.html). Com esse arquivo no mesmo diretório do arquivo python da análise de dados, utiliza-se a biblioteca Geopandas para atribuir os municípios e seus formatos do arquivo shapefile à uma lógica de grupos de municípios (citam apenas moeda social, apenas palavras-chaves amplas ou ambas) e cores representativas (azul, verde e turquesa, respectivamente). Os municípios que não estão em um grupo, os que não citam palavra-chave alguma que buscamos, ganham a cor cinza.

4 Processo de desenvolvimento

Nesta seção, será apresentado de forma detalhada o processo de desenvolvimento do projeto, abrangendo todas as etapas desde a sua concepção inicial até a implementação final. Serão discutidos os desafios encontrados ao longo do desenvolvimento, as dificuldades técnicas e conceituais que surgiram, bem como as alternativas adotadas para contorná-las. Além disso, serão descritas as soluções implementadas para resolver os problemas identificados, assegurando que os objetivos do projeto fossem alcançados de maneira eficaz. A seguir, abordaremos a modelagem inicial do sistema, as escolhas metodológicas e as ferramentas utilizadas, proporcionando uma visão abrangente das decisões tomadas durante o desenvolvimento e os resultados obtidos em cada fase do processo. Para mais detalhes da linha do tempo específica, acessar o kamban do trio 1.

Podemos dividir as etapas do desenvolvimento do projeto em 6 partes, cada uma a qual dedicamos um tempo para realização e que, às vezes, surgia um problema relacionado ao tópico: organização inicial, acesso inicial ao site, iteração de município e candidato, acesso ao PDF, estrutração da planilha de resultados e análise dos resultados.

4.1 Organização Inicial

Na etapa inicial do projeto, foi realizada a divisão de tarefas entre os integrantes do grupo, garantindo que cada membro tivesse atribuições claras e específicas para o desenvolvimento das atividades. Além disso, foi estabelecido contato com o facilitador Eduardo Diniz, o que permitiu alinhar expectativas e esclarecer eventuais dúvidas relacionadas à proposta do projeto, bem como pegar antiga planilha constuída pela busca manual feita por ele. Essa organização preliminar transcorreu sem dificuldades, assegurando o planejamento adequado para as etapas subsequentes, que foram separadas de acordo com as principais bibliotecas a qual estão relacionadas: o Selenium para acessar o site, o PyPDF2 para acessar PDFs das propostas e o Pandas para criação das planilhas.

4.2 Acesso Inicial ao Site

Durante essa etapa, o foco principal se voltou a aprender a utilizar a biblioteca Selenium para acessar o site do TSE e percorrê-lo. Dessa forma, o código protótipo relacionado a essa parte conseguia adentrar o site e chegar até a página no qual se escolhe os municípios e o cargo a ser pesquisado. De um modo geral, não houve problemas e nem muitas dificuldades para o uso do Selenium.

4.3 Iteração de Municípios e Candidatos

Durante o processo de iteração pelos municípios e candidatos, identificou-se um problema recorrente relacionado à ausência de propostas associadas a alguns candidatos ou à presença de links externos direcionando para outros sites. Essas situações resultaram em erros durante as iterações, dificultando a coleta completa das informações previstas.

A ausência de propostas exigiu a implementação de verificações adicionais no código para lidar com campos vazios, enquanto os links externos demandaram tratamento especial para evitar interrupções no fluxo de execução. Essa análise permitiu identificar a necessidade de aprimorar os filtros e a lógica de captura dos dados, garantindo maior robustez no processo automatizado.

4.4 Acesso ao PDFs

Durante o processo de acesso e manipulação dos arquivos PDF, enfrentamos diversos desafios técnicos. Inicialmente, houve problemas relacionados às diferenças nos Sistemas Operacionais utilizados pelos integrantes do grupo, o que sendo o maior obstáculo enfrentado pelo trio. O único código que estava sendo desenvolvido por dois integrantes do trio que utilizam dois SOs distintos, respectivamente Windows e MacOS, acabou tendo que ser repartido em duas versões desenvolvidas, por causa da diferença de funcionalidade das bibliotecas que realizam a leitura e acesso em PDFs nesses SOs. Para o MacOS, a biblioteca utilizada foi o "fitz", enquanto para o Windows a biblioteca utilizada foi o "PyPDF2": o front que deu mais certo acabou sendo o do Windows, por isso é a versão final do software usa o PyPDF2. Vale notar que o código feito utilizando a biblioteca "fitz"está disponível no GitHub na pasta "Old", caso seja de interesse de futuros desenvolvimentos fazê-lo no MacOS.

Outro problema relevante nessa etapa do desenvolvimento foi o tratamento de texto extraído dos PDFs, especialmente em relação ao Unicode. Os textos continham acentos e símbolos especiais, que nem sempre foram processados corretamente pelas bibliotecas, gerando inconsistências nos resultados. Além disso, palavras-chave presentes nos documentos que apresentavam letras maiúsculas acabavam não sendo identificadas, o que exigiu a aplicação de filtros e ajustes na lógica de busca que os tornavam em caracteres minúsculos para assegurar uma extração eficiente.

Foi criado um diretório dedicado para salvar temporariamente os PDFs baixados durante a execução do programa. Após a leitura e extração dos dados, os arquivos foram automaticamente excluídos para economizar espaço e manter o ambiente organizado. Esse fluxo contribuiu para a gestão eficiente dos recursos do sistema e evitou o acúmulo desnecessário de arquivos.

4.5 Estrutração da Planilha de Resultados

Para organizar os dados coletados durante o projeto, foi realizada a estruturação da planilha de resultados utilizando as bibliotecas pandas e openpyxl. O pandas foi empregado para manipulação e organização dos dados, permitindo sua transformação em um formato estruturado e de fácil interpretação, em csv. Já o openpyxl foi utilizado para exportar os dados para o formato Excel, possibilitando a criação de planilhas customizadas e adequadas para análise posterior. Não houve dificuldades para se aprender a utilizar essas bibliotecas, e essa etapa ocorreu tranquilamente.

4.6 Análise de Resultados

Após reuniões com o facilitador decidimos adicionar mais uma entrega do projeto: produzir análises e visualizações dos resultados da planilha. Assim, por meio de scripts Python e bibliotecas especializadas, produzimos gráficos de pizza e de barras e um mapa personalizado do estado do Rio de Janeiro.

A partir dessas visualizações, produzimos mais um entregável para o nosso facilitador: um relatório com foco nos resultados, complementar à planilha, diferentemente deste relatório mais detalhado e extenso.

5 Manual para evolução

São propostas duas possíveis melhorias em relação ao projeto:

1. Realização das buscas em todos os estados do Brasil
2. Uso de modelos de Machine Learning

Para a proposta 1, basta acrescentar mais iterações para cada estado de cada região do Brasil. Note que o código já feito para esse projeto não necessita ser alterado (em caso de não haver atualizações no site) e basta que ocorra a adição dessas iterações, com o código atual podendo ser usado como um módulo da nova extensão do projeto. Segue abaixo o pseudo-código que o trio propõe como continuação do projeto.

Vale notar que o tempo de execução apenas para o estado do Rio de Janeiro foi de 1 hora, dessa forma é possível que o tempo de execução dessa extensão do projeto demore bastante. Além disso, quanto mais automatizado for o processo de busca, maior a chance do código não executar corretamente em caso de uma possível atualização do site, portanto a restrição a cada estado é melhor.

Algorithm 2 Busca de palavras-chaves no site de candidaturas do TSE - modelo geral

```
Inicializar variáveis, como a lista de palavras-chaves de interesse L e a lista vazia que conterá os resultados encontrados R
Acessar a página sobre as eleições municipais de 2024 no site do TSE
for cada região do Brasil r do
    for cada estado e da região r do
        Acessar a aba do estado e, selecionando a parte "Candidaturas"
        for cada municipio m do estado e do
            Escolher a opção "Prefeito"no cargo a ser pesquisado
            Apertar para pesquisar
            for cada candidato c do município m do
                Acessar o perfil de c
                Acessar a aba de "Propostas"do perfil de c
                Baixar o PDF que contém as propostas de governo de c
                Fazer a leitura desse PDF e procurar nele as palavras-chaves contidas em L, salvando as informações de c e as palavras encontradas em R
            end for
        end for
    end for
end for
Retornar R
```

Para a proposta 2, seria necessário um projeto de mais longa duração para viabilizar a implementação de técnicas mais sofisticadas. Um desafio que nós nos deparamos foi o "engessamento"da busca por palavras-chaves fixas, pois candidatos que expressassem a mesma ideia mas com palavras fora da nossa relação ou com o sentido diluído em uma frase ou parágrafo não seriam contabilizados pelo nosso software. Uma maneira de solucionar essa questão seria utilizar tecnologias que focam não nos caractéres das palavras, como o nosso programa, mas sim em ferramentas que foquem no sentido semântico das palavras e frases. Algumas técnicas que compreendem sentido e poderiam ser aplicadas são: Large Languagem Models (LLM's), como o chat GPT, ou Transfer Learning, com Word Embedding, vetorizando as palavras.

Uma outra possível melhoria para o projeto seria estender a busca para outros navegadores, visto que a feita para esse projeto só funciona para navegadores de base Chromium. Também existe a possibilidade de se criar códigos que funcionem em outros Sistemas Operacionais diferentes do Windows, uma vez que a implementada atualmente apenas funciona para esse Sistema Operacional. Embora úteis, esses aprimoramentos foram considerados de menor prioridade no escopo atual, dado o foco nos objetivos principais do projeto. Implementá-las seria mais relevante em fases futuras ou em cenários que demandem maior abrangência de uso.

6 Colaborações

Durante a implementação do código do projeto, podemos citar duas importantes colaborações com outros trios que contribuíram no desenvolvimento de nosso software:

1. Trio 5: a ideia de salvar os arquivos PDFs baixados do site do TSE em um diretório específico surgiu de uma conversa que um dos desenvolvedores de nosso trio teve com um dos membros do trio 5, que possui experiência com o uso do Python. Essa conversa ocorreu durante a etapa de desenvolvimento em que o nosso trio se voltava ao acesso aos PDFs, e serviu como guia para a forma como os PDFs seriam acessados por nosso código: o PDF era baixado e salvo nessa pasta ligada ao código, isso facilitaria o acesso ao documento e possibitou que ele pudesse ser apagado logo após a busca de modo a não lotar o armazanemento durante a execução do código. Isso foi bastante relevante pois esse foi um dos problemas o qual encontramos ao lidar com o acesso aos PDFs: como baixá-los e acessá-los pelo código.

2. Trio 3: o trio 3 nos forneceu suporte diante do uso da biblioteca Pandas e openpyxl, já que um de seus membros possui experiência com o tratameto com planilhas por meio de códigos. Essa interação auxiliou nossa etapa de desenvolvimento relacionada ao salvamento dos dados nas planilhas, tornando-a simples e rápida no desenvolvimento.

7 Resultados

As planilhas contendo os resultados coletados com a execução do código está disponível em: https://docs.google.com/spreadsheets/d/176i8ukVybSI_248uLRSYHimavorrBBRg/edit?usp=sharing&ouid=100386229160682011467&rtpof=true&sd=true. Além disso, também estão disponíveis na pasta "Resultados"dentro do repositório GitHub de link https://github.com/guilherme-hu/Projeto-CompSoc/tree/main

A primeira planilha é a geral de candidatos encontrados. As planilhas eleitos, partidos e palavras são "recortes"obtidos da planilha geral para a feitura dos gráficos. As planilhas 2020x2024 e a planilha de resultados apenas das cidades elegíveis a segundo turno são para comparar com os resultados manuais obtidos nas eleições de 2020 pelo professor Eduardo Diniz.

7.1 Visualizações

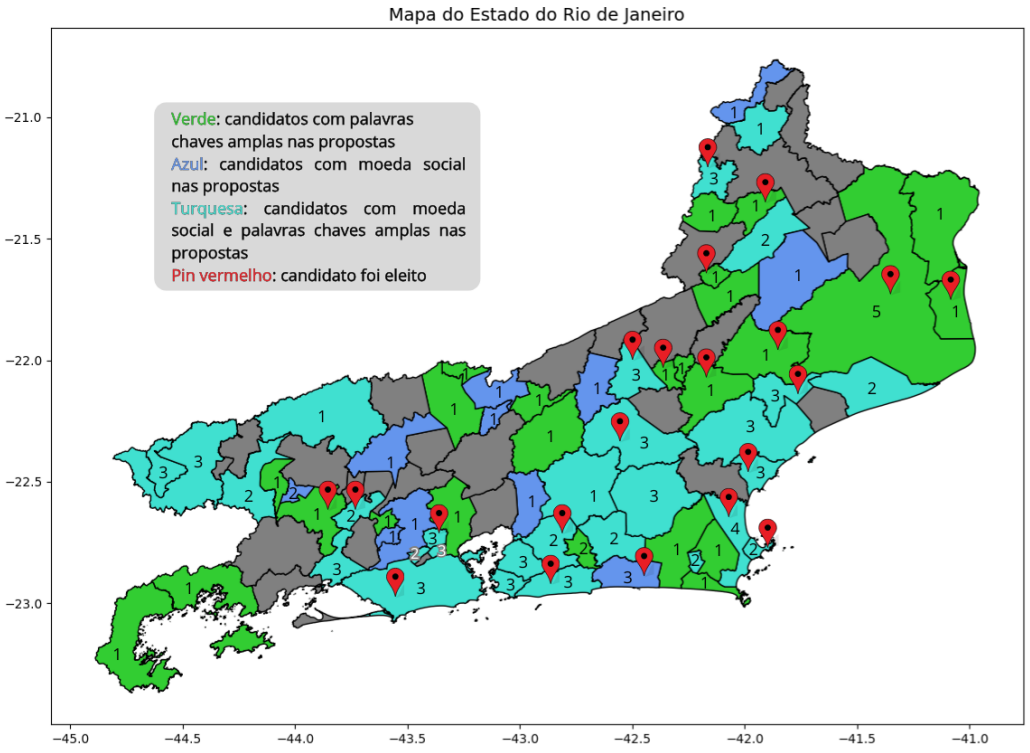

Figura 1: Mapa do Rio de Janeiro

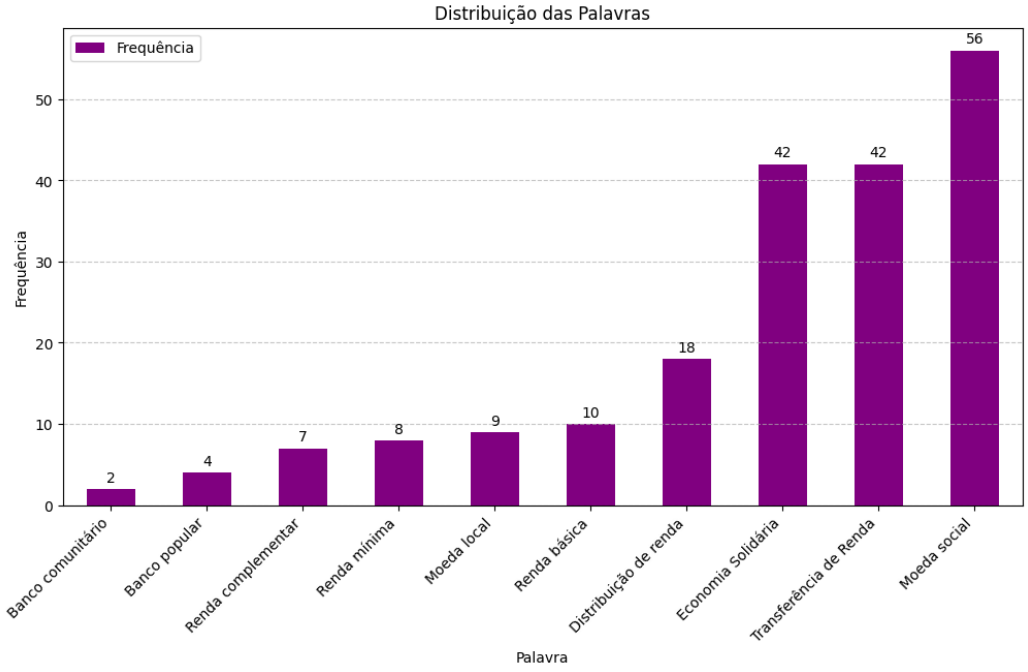

Figura 2: Palavras encontradas nas propostas de governo

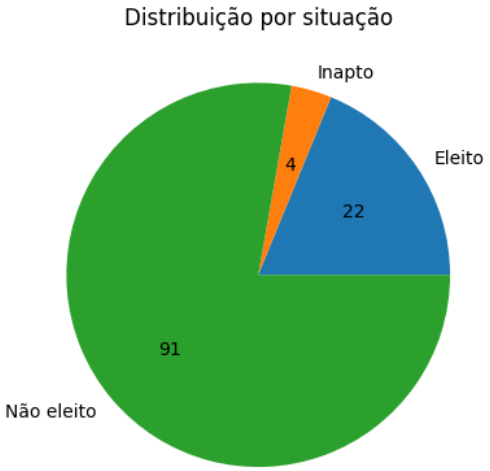

Figura 3: Situação dos candidatos encontrados

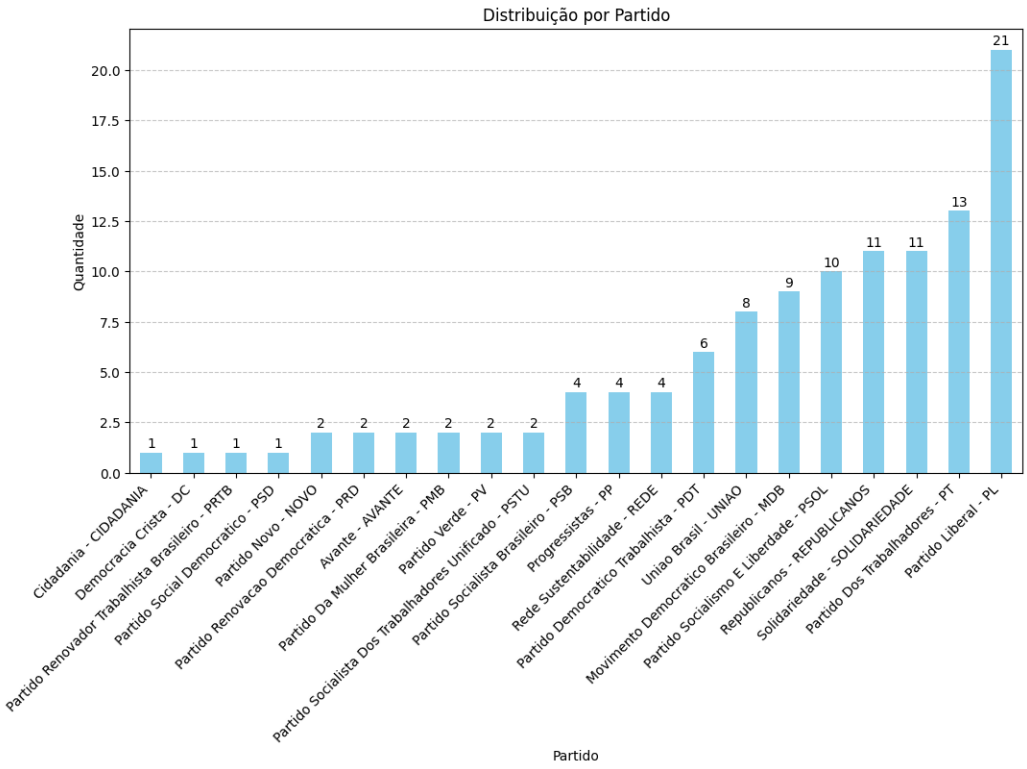

Figura 4: Partidos dos candidatos encontrados

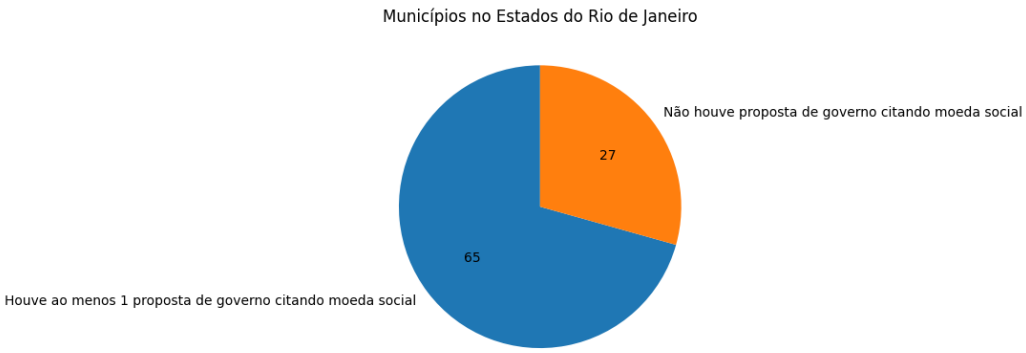

Figura 5: Quantidade de municípios

*moeda social e/ou palavras-chaves amplas

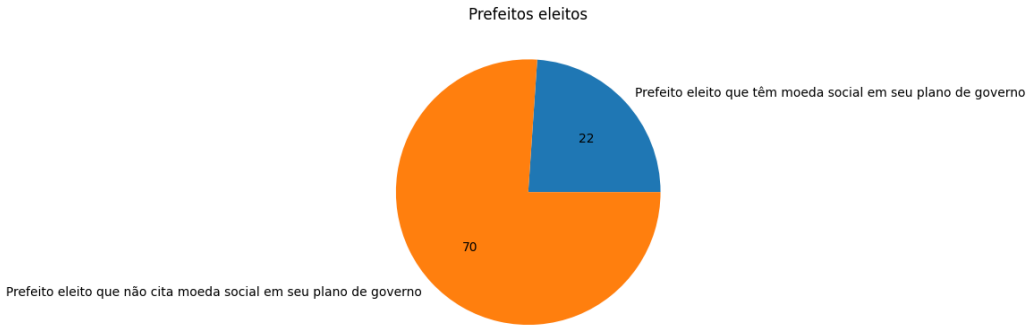

Figura 6: Propostas dos prefeitos eleitos

*moeda social e/ou palavras-chaves amplas

7.2 Discussão dos resultados

De um modo geral, em comparação a planilha de 2020, o número de candidatos que citam alguma proposta de moeda social ou temas correlatos cresceu. Além disso, o partido que mais teve candidatos filtrados foi o PL, o que seria inesperado devido a seu posicionamento politico do espectro ideológico mais à direita. Por fim, pode-se falar que, apesar do crescimento considerável em comparação a 2020, a maior parte dos prefeitos eleitos ainda não adotou quaisquer medidas relacionadas a implementação ou ampliação de uma política de economia solidária.

8 Conclusão

Neste trabalho, foi desenvolvido um software em Python capaz de realizar a busca automatizada de palavras-chaves no site de candidaturas. Enfrentamos desafios significativos, como a otimização do desempenho, a correção de erros e o completo entendimento das bibliotecas usadas. Este projeto não apenas reforçou nosso conhecimento teórico sobre Python, como também nos proporcionou a prática em programação, resolução de problemas, interação com cliente, uso de Web Scraping, acesso a PDFs e salvamento em planinhas, servindo como uma base para futuras experiências na área de interação com páginas Web.