Relatório Projeto 2 - Megadados 2020.2
Analisando dados com Spark

Lucas Muchaluat & Luiz Vitor Germanos
https://github.com/lucasmuchaluat/SparkAnalysis

O trabalho:

O projeto consiste em uma análise das páginas da internet brasileira a fim de descobrir padrões. Para tal, serão usados índices referentes à frequência (IDF e FREQ) com o objetivo de traçar e identificar as palavras mais relevantes de cada documento no qual as palavras escolhidas aparecem. Foi feita uma seleção de duas palavras chave a serem estudadas tanto conjuntamente, quanto separadamente.

Palavras escolhidas:

Com a corrida eleitoral para a prefeitura de São Paulo a todo vapor, as palavras escolhidas para serem analisadas foram os sobrenomes dos dois candidatos ao segundo turno, Covas e Boulos. Em meio a essa disputa, espera-se inferir se as palavras mais relacionadas ao candidato dizem respeito a suas propostas e ideias.

A análise:

Inicialmente, para fazer uma análise acurada do vocabulário comum da internet no Brasil foi necessário implementar um filtro. Palavras que não possuem significado prático, como conectivos, artigos ou preposições, foram removidas. Sem contar que palavras com poucas ocorrências também foram tiradas, visto que não possuem impacto significativo na amostra de massa. Para tal, foi necessário estipular uma quantidade mínima de documentos em que apareçam qualquer uma das palavras e também limitar o teto de ocorrências para não entrarmos no caso de artigos e preposições. O número mínimo de documentos escolhido foi 10 e o número máximo foi 70% do total de documentos. Assim, palavras que não se inserem entre esses limites foram excluídas.

Feita toda essa filtragem, uma tabela de frequência inversa de documentos (inverse document frequency – IDF) foi construída. Ela mapeia cada palavra para o logaritmo da frequência inversa do número de documentos em que aquela palavra aparece:

$$\mathrm{IDF}(\text{palavra}) = \log_{10}\left(\frac{N}{\mathrm{df}(\text{palavra})}\right)$$

onde df(palavra) é o número de documentos em que a palavra aparece, e *N* é o número total de documentos da base.

Palavras com um baixo IDF são palavras comuns, que pouco agregam para a análise de textos em relação a alguma entidade que se queira caracterizar. Palavras com IDF muito alto são exóticas, que podem ser devidas simplesmente a erros de grafia ou são palavras específicas, como códigos numéricos ou marcações de página. Em ambos os casos, queremos ignorar estas palavras.

Em seguida, é necessário analisar o vocabulário específico dos documentos filtrados. Nessa etapa, se torna necessário o cálculo da frequência normalizada de cada palavra que se insere nesses documentos. Ela segue a seguinte distribuição:

$$\text{freq}(\text{palavra}) = \log_{10}\big(1 + \text{contagem}(\text{palavra})\big)$$

onde a contagem (palavra) é o número absoluto de vezes que a palavra ocorre no conjunto completo de textos.

Terminado o cálculo da frequência, é possível obter a relevância das palavras. Ela é dada pela frequência de ocorrência de cada palavra multiplicado pelo IDF da mesma:

$$\text{relevancia}(\text{palavra}) = \text{freq}(\text{palavra}) \times \text{IDF}(\text{palavra})$$

O resultado da relevância de cada palavra será analisada na seção Resultados a seguir.

Explicação do código:

1. **Funções**
   - **conta_palavras_em_doc**
     - ➢ Esta função é responsável por contar todas as palavras presentes em um documento da web.

```python
def conta_palavras_em_doc(item):
    url, text = item
    words = text.strip().split()
    return [(word.lower(), 1) for word in set(words)]
```

   - **junta_contagens**
     - ➢ Uma função de agregação, que é responsável por somar todas as palavras.

```python
def junta_contagens(nova_contagem, contagem_atual):
    return nova_contagem + contagem_atual
```

   - **filtra_doc_freq**
     - ➢ Função responsável por filtrar palavras que não passam de uma threshold mínima e máxima de aparições.

```python
def filtra_doc_freq(item):
    palavra, contagem = item
    return (contagem < DOC_COUNT_MAX) and (contagem >= DOC_COUNT_MIN)
```

- **computa_idf**
  - ➢ Esta função calcula o idf de cada palavra

```
def computa_idf(item):
    palavra, contagem = item
    idf = math.log10(N / contagem)
    return(palavra, idf)
```

- **computa_freq**
  - ➢ Esta função calcula a frequência de cada palavra

```
def computa_idf(item):
    palavra, contagem = item
    idf = math.log10(N / contagem)
    return(palavra, idf)
```

- **conta_palavras_total**
  - ➢ Esta função separa as palavras únicas de um documento

```
def conta_palavras_total(item):
    url, text = item
    words = text.strip().split()
    words = [w for w in words if w.isalpha()]
    words_filtered = [w.lower() for w in words if len(w) > 3]
    return [(word.lower(), 1) for word in words_filtered]
```

- **computa_relevancia**
  - ➢ Esta função calcula a relevância de uma palavra a partir do seu idf e

```
def computa_relevancia(item):
    palavra, valor = item
    freq, idf = valor
    relevancia = freq*idf
    return (palavra, relevancia)
```

freq

- **encontra_palavra**

➢ Esta função é responsável por aferir se um texto contém uma palavra em específico.

```
def encontra_palavra(item, palavra):
    url, text = item
    words = text.strip().split()
    words_list = [word.lower() for word in words]
    if palavra in words_list:
        return [item]
    return []
```

- **encontra_palavra1 e encontra_palavra2**
  ➢ Esta função é utilizada para passarmos as palavras de estudo para função **encontra_palavra**.

```
def encontra_palavra1(item):
    return encontra_palavra(item, palavra1)

def encontra_palavra2(item):
    return encontra_palavra(item, palavra2)
```

- **conta_palavras_local**
  ➢ Esta função é quase análoga a função conta_palavras, porém ela é restrita a um vocabulário local, contando apenas palavras a uma distância de 5 sentenças das palavras estudadas.

```
def conta_palavras_local(item):
    url, text = item
    words = text.strip().split()
    words = [w for w in words if w.isalpha()]
    filtered_words = []
    for i in range(len(words)):
        if i < 5:
            if palavra1 in words[: i + 5] or palavra2 in words[: i + 5]:
                filtered_words.append(words[i])
        elif i > len(words) - 5:
            if palavra1 in words[i - 5 :] or palavra2 in words[i - 5 :]:
                filtered_words.append(words[i])
        else:
            if (palavra1 in words[i - 5 : i + 5] or palavra2 in words[i - 5 : i + 5]):
                filtered_words.append(words[i])

    filtered_words = [w.lower() for w in filtered_words if len(w) > 3]
    return [(w.lower(), 1) for w in filtered_words]
```

2. **Pipeline**
    1. Inicialmente definem-se os termos a serem estudados, no nosso caso é “boulos” e “covas”.
    2. Feito isso, os documentos que contêm cada uma das palavras são filtrados e separados em 2 rdds.
    3. É calculado o IDF de todas as palavras.
    4. É calculado o número absoluto de vezes que a palavra ocorre no conjunto completo de textos, usando a função “conta_palavras_total”.
    5. Para cada caso é calculada a frequência normalizada das palavras, usando o resultado do passo anterior, com a função “computa_freq”.
    6. Com esses dados é calculado o índice de relevância de cada palavra, usando a função "computa_relevancia".
    7. Por último, é gerado uma tabela com as 100 palavras mais relevantes.
    8. A pipeline do passo 1 ao 7 é repetida para cada palavra e para a interseção no cenário local e global,

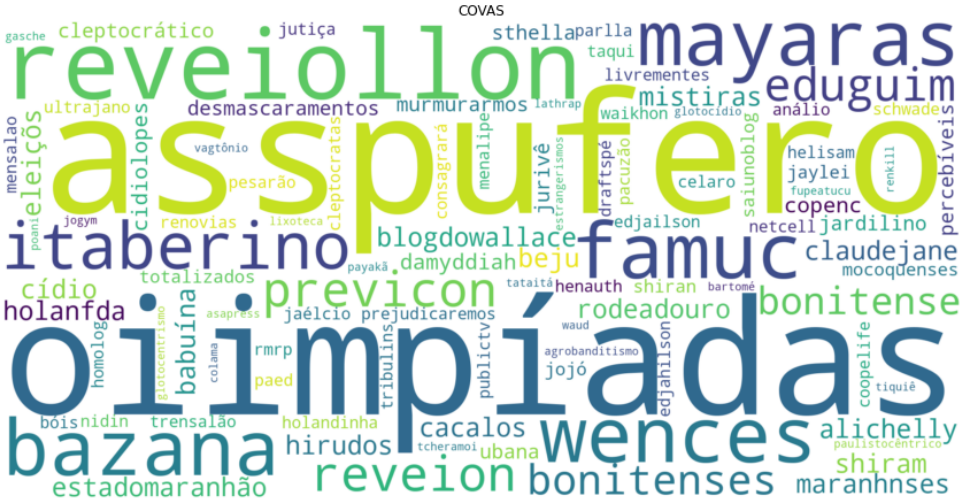
Resultados:

Após rodarmos a pipeline no cluster, chegamos nos resultados globais e locais das palavras estudadas. Para facilitar a visualização, foi montado uma *WordCloud* a fim de destacar pontos de dados textuais importantes. Segue o resultado abaixo. Como é possível perceber, as palavras resultantes do método global não nos dizem muita coisa. A maioria não está muito relacionada com a palavra, mas temos alguns resultados interessantes.

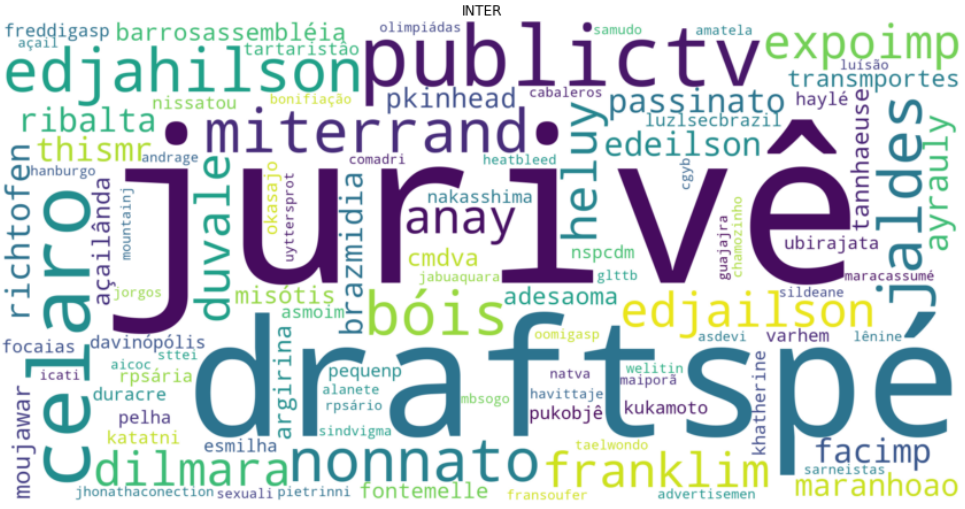
Globais:

Para o caso de “boulos”, temos algumas aparições relevantes que fazem sentido com o contexto do candidato. Como a palavra “boitempo”, que aparece algumas vezes na formação de sentenças. Tal palavra é referente ao nome de um jornal que Guilherme Boulos é escritor. Temos também "blogdofranklin", um nome de um blog de política que o mencionou em alguma de suas reportagens. Já a palavra “antissionista”, nome dado ao

movimento de oposição política contra o estado de Israel, aparece com alguma relevância devido a algumas matérias, onde Boulos acusou o estado de praticar atos terroristas.
Contudo, grande parte dos resultados obtidos, não aparentam ter relação direta com o político. Isto acontece por conta de páginas que possuem anúncios e chamadas, fora do texto principal, que possuem o seu nome.



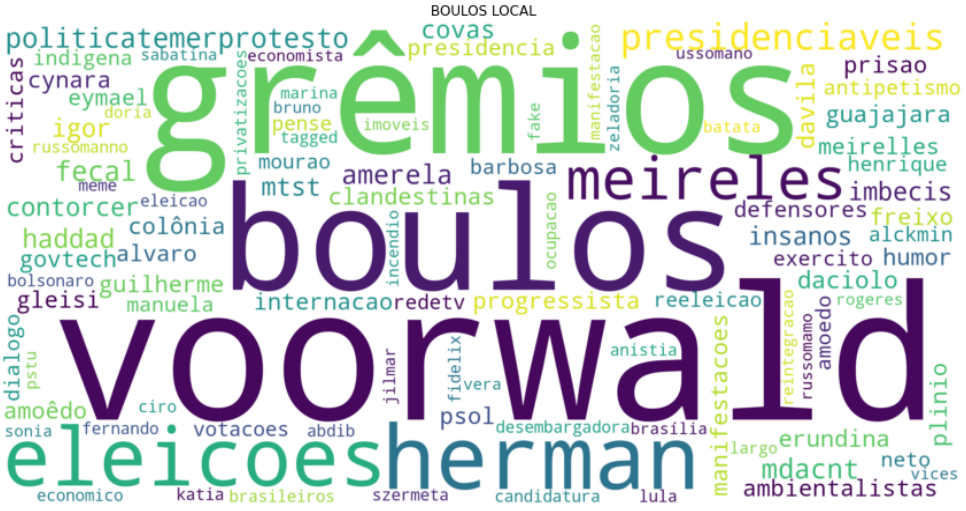
No caso de “covas”, os resultados foram um pouco mais promissores. Quanto às palavras mais relevantes, “asspufero” (associação dos servidores públicos federais e estaduais de rondônia) e “oiimpíadas” (olimpíadas gramaticalmente mal escrito) não apresentam relações claras com o candidato à prefeitura. No entanto, algumas palavras aparentam ter significado propriamente dito. Quanto à “réveillon”, Bruno Covas foi o responsável pelo cancelamento da festa de fim de ano na avenida paulista em 2020 por conta do Covid-19. Já “Mayaras” tem grandes chances de fazer menção à filha do candidato a Vice-Prefeitura da chapa PSDB, Ricardo Nunes. “Eduguim” é um jornalista, autor e escritor de artigos políticos. Por fim, é minimamente interessante a palavra “desmascaramentos” ter aparecido. A analogia ao uso de “máscaras” dessa nova geração política a fim de enganar o eleitor aparenta estar cada vez surtindo menos efeito.



Já os resultados para a interseção das palavras, aparentam ter pouca relação. As aparições mais relevantes, “jurive” e “draftspe”, são sentenças que aparentam não remeter nada aos

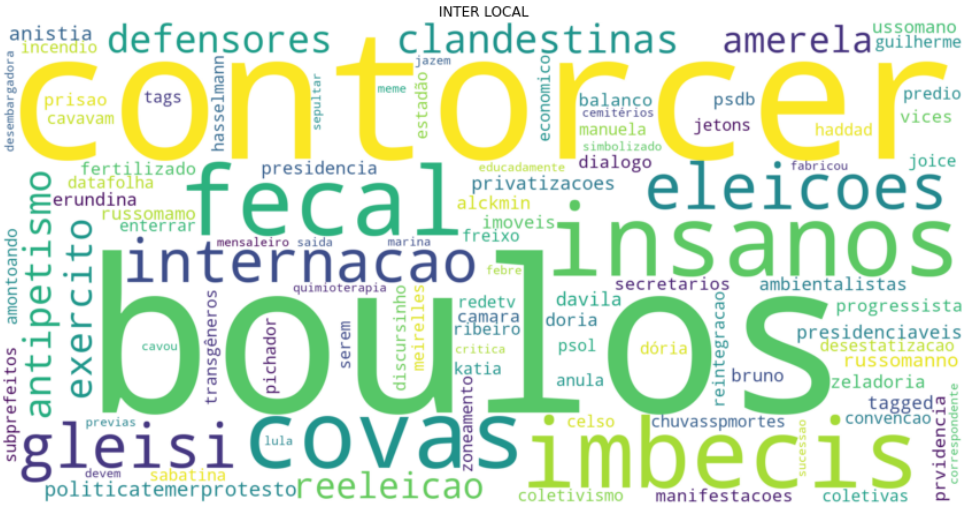
políticos. Porém, é possível observar resultados recorrentes como “bois”, um possível indicativo do jornal Boitempo, e “franklim”, referência provável ao blog político “blogdofranklim”, ambos encontrados na WordCloud de “boulos”. Temos também a aparição do estado do Maranhão, tanto diretamente através de “maranhoao”, que acreditamos significar “maranhão”, quanto indiretamente através das palavras "açailândia", uma cidade maranhense onde Boulos fez campanha em 2018 para a presidência, “expoimp”, uma exposição agrícola no estado e “facimp”, uma faculdade presente também no estado. Acreditamos que essa forte presença se dá pela campanha pesada feita no nordeste pelo candidato do PSOL na corrida presidencial em 2018. Por fim, existe também a recorrência da palavra “oiimpíadas”, aparecendo anteriormente na WordCloud de “covas”, mas ainda não aparenta ter uma relação clara com os candidatos.

Locais:



Os resultados locais da palavra “boulos” aparentam ser muito mais satisfatórios em comparação às suas contrapartes globais. Temos como uma das principais aparições a palavra “voorwald”, sobrenome de Henry Voorwald, ex secretário da educação do estado de São Paulo, entre os anos de 2011 a 2015, o qual Guilherme Boulos faz críticas em sua coluna no jornal Folha de São Paulo. A palavra "grêmios" também tem bastante relevância, tendo em vista que a proposta de educação do candidato dá ênfase à participação ativa de grêmios estudantis, nas suas respectivas escolas. Um resultado interessante é a aparição de “meireles”, sobrenome do candidato a presidente em 2018 e ex-ministro da fazenda Henrique Meirelles, concorrente de Guilherme, nesta eleição. Na mesma linha, temos a aparição com menos relevância, de nomes de políticos relevantes das últimas eleições como “ciro”, “daciolo”, “manuela”, “bolsonaro”, “gleisi”, “amoedo”, “haddad” entre outros. Palavras referentes ao contexto do candidato aparecem igualmente, como “psol”, sigla do partido em que Boulos é filiado, e “mtst”, sigla do Movimento dos Trabalhadores sem Teto o qual ele é uma liderança importante.

Os resultados da palavra “covas” localmente trazem alguns insights curiosos. A começar pelas mais relevantes. O termo “internação” pode levar para dois caminhos. O primeiro referente a internação de Bruno Covas em 2020 por conta de uma infecção desenvolvida no cólon. O segundo referente a ala de internação do Hospital Estadual Mário Covas, reconhecido em SP, feito em nome de seu avô. A palavra “enterrasse” pode não significar algo relacionado ao prefeito, visto que o ato de enterrar é conectado ao significado literal do substantivo covas. O termo “zeladoriaspcriticasbruno” faz menção às tags de pesquisa em diversas revistas e jornais sobre ZELADORIA/SP/CRÍTICAS/BRUNO COVAS. As palavras que remetem à água (“chuva” e “rasas”) são referentes às fortes enchentes e alagamentos que São Paulo enfrentou no começo do ano de 2020, quando o prefeito era questionado sobre como mitigar as perdas e contornar a situação. “Abraciclo” é uma ocorrência interessante, pois diz respeito ao aumento de produção de bicicletas decorrente da promessa de Bruno Covas de criar 173,5 km de novas ciclovias em São Paulo em 2020. “antipetismo” e “ativismo” podem ser decorrentes da filiação partidária de Covas ao PSDB que luta para diminuir a influência do partido concorrente PT nas eleições.



Por fim, a intersecção local dos sobrenomes dos candidatos, fornecem um parecer aparentemente superior à sua contraparte global. Porém, um resultado que causa indagações é “contorcer”, ele aparece com tanta relevância pois a cantora sertaneja Maraísa, acabou virando notícia em diversos tabloides, por postar uma fotografia se contorcendo enquanto fazia yoga, estes sites divulgam em seu footer resultados de pesquisas eleitorais, algo que acabou fazendo com que a palavra ganhasse proeminência. Em compensação, temos como uma das aparições mais relevantes as palavras “boulos” e “covas”, com a palavra “boulos” tendo uma maior relevância, indicando que a presença na internet do candidato é mais significativa. Um insight que não era possível retirar a partir do resultado anterior. Temos também a palavra “fecal”, que faz parte de um trocadilho “Boulos fecal” que internautas de oposição a Guilherme utilizam para se referir a ele. Já a aparição de “insanos” pode ser proveniente do uso frequente dessa expressão por críticos de ambas as frentes que fazem menções aos candidatos, o mesmo pode ser dito a expressão “imbecis”, algo que mostra o nível da polarização política no Brasil.

Conclusão:

Após uma interpretação dos dados acima é possível concluir que os resultados locais trazem mais insights interessantes, sobre os objetos de estudo, que as suas contrapartes globais. Acreditamos, que isto se deve pelo fato das páginas da internet estarem povoadas de links e menções a respeito de tópicos não relacionados ao seu conteúdo principal, algo que acaba por poluir os resultados de uma análise global de relevância. Já a análise local acaba por mitigar este efeito, mas não por completo, devido a como as páginas são interpretadas pelo crawler, algumas palavras sem relação acabam por ganhar relevância superestimada, como foi o caso da palavra “contorcer” no WordCloud da interseção local. Para uma futura iteração do projeto um filtro local mais elaborado poderia ser construído, para lidar com essas incoerências.