Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Mayara Cassia de Melo

Análise Computacional das Decisões de Tribunais Superiores Brasileiros

São Paulo
Dezembro de 2015

Análise Computacional das Decisões de Tribunais Superiores Brasileiros

Monografia final da disciplina

MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Marcelo Finger

São Paulo

Dezembro de 2015

Resumo

O Supremo Tribunal Federal e o Superior Tribunal de Justiça proferem decisões definitivas que servem como referência para as demais instituições judiciais. Grande parte dessas decisões são públicas e estão disponíveis na Web, porém faltam análises sobre um escopo geral. Este trabalho tem como objetivo compilar estatísticas e propor métricas para investigar os processos que regem o sistema judiciário.

Para tanto, é realizado o Web Scraping das páginas dos tribunais para a extração dos dados das jurisprudências. A informação é extraída e persistida em um banco de dados não relacional. A partir de consultas ao banco, estatísticas gerais são geradas. As decisões são então modeladas como uma rede, na qual são conectadas por citações que usam para fundamentar seus argumentos com base em outras decisões. Sobre a rede, são aplicados os algoritmos de PageRank, a fim de obter um rankeamento por relevância das decisões e o algoritmo de Floyd-Warshall para detectar ciclos.

Palavras-chave: STF, STJ, acórdão, Web Scraping, PageRank, Floyd-Warshall.

Abstract

The Supreme Federal Court and the Superior Court make final decisions that serve as reference for the other judicial institutions. Large part of those decisions are public and are available on the Web, but lacks analysis of a general scope. This monograph aims to compile statistics and propose metrics to investigate the procedures governing the judiciary.

Therefore, is performed a Web Scraping of the pages of the courts for the extraction of cases data. The information is extracted and kept in a non-relational database. From queries to the database, general statistics are generated. Decisions are then modeled as a network, which are connected by using quotes to substantiate arguments based on other decisions. On the network, the PageRank algorithm to obtain a ranking by relevance of the decisions and the Floyd-Warshall algorithm to detect cycles are applied.

Keywords: supreme court, superior court, law case, Web Scraping, PageRank, Floyd-Warshall.

Sumário

Capítulo 1

Introdução

Impulsionada pela informatização de processos e pelo desenvolvimento de dispositivos eletrônicos cada vez mais poderosos e acessíveis, a geração de dados vêm crescendo exponencialmente. O potencial de tamanho montante de informações é imensurável, contudo a compilação de dados úteis e concisos se torna um desafio. Nesse sentido a problemática não se encontra mais na falta de informação, como no passado, mas sim na confusão que causa sua abundância. Organizar os dados a fim de descobrir os fatos relevantes e tomar decisões de melhoria de processos é o novo desafio da atualidade.

Os órgãos governamentais, em particular, são grandes produtores de informação, as quais são de extrema importância para a sociedade. Nesse sentido, há esforços para publicar estes dados de forma acessível, com o intuito de promover maior participação social. Espera-se, que a partir de uma maior democratização de conhecimentos, a população possa colaborar na construção de melhorias para o Estado[1].

O sistema judiciário tem avançado no processo de informatização e digitalização de seus processos. Vários tribunais disponibilizam, em seus endereços eletrônicos, grande parte de suas jurisprudências, inclusive por meio de mecanismos de buscas. Porém faltam dados sobre o conjunto das decisões, o que facilitaria análises de escopo geral sobre os mecanismos dos processos de decisão que regem o terceiro poder.

O Poder Judiciário é regulado pela Constituição Federal e constituído por diversos órgãos. Entre eles se destacam os tribunais superiores, que julgam casos de terceira e última instância, proferindo decisões finais sobre diversas matérias [2]. Seus expoentes

são o Supremo Tribunal Federal, órgão máximo do judiciário, cujas decisões não podem ser contestadas por nenhuma outra corte, e logo abaixo, o Superior Tribunal Judicial (STJ). Ao STF, cabe zelar pelo cumprimento da constituição e, ao STJ, pela interpretação uniforme da legislação federal [3].

A relevância do STF e do STJ e a existência de dados digitais abertos e razoavelmente bem estruturados inspiraram o projeto da qual este trabalho faz parte. O projeto foi idealizado pelo Professor Doutor Marcelo Finger e iniciado pelo bacharel Alessandro Caló[4]. Baseia-se especificamente nos dados sobre acórdãos, isto é decisões proferidas por órgãos colegiados dos tribunais [5]. Em contrapartida às decisões monocráticas, que advém de apenas um juiz, os acórdãos são o resultado da concordância total ou parcial de um grupo de julgadores. Sobre os dados são geradas estatísticas quantitativas em escopo geral e métricas são propostas para medir a relevância das jurisprudências.

O projeto inicial fazia o *download*, em arquivos de texto, das fichas-resumos das jurisprudências do STF através de um webcrawler. A partir das fichas, usando código Java, são extraídos o código identificador (ID), ministro-relator, unidade federativa, data de julgamento, indexação e citações dos acórdãos. Essas informações são persistidas em um banco não relacional, possibilitando a geração de estatísticas e um rankeamento dos acórdãos pela aplicação do algoritmo de PageRank [4]. Para este trabalho, pretendeu-se estender esses e outros métodos ao STJ, gerando novas estatísticas e métricas.

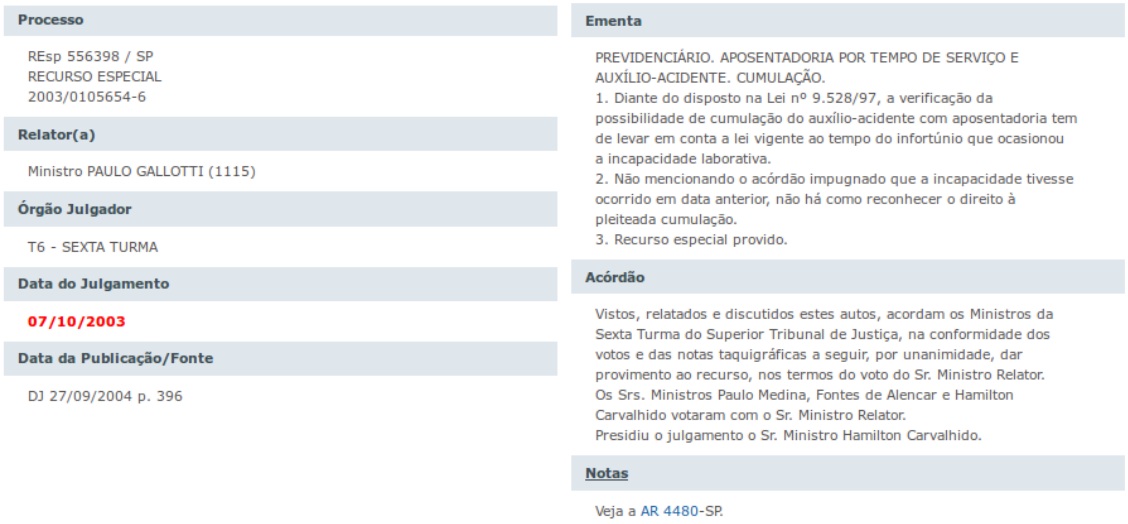
A seguir serão descritas as características dos dados, os métodos e recursos computacionais utilizados para extração de informação dos *websites* e de persistência. Será proposta uma modelagem dos acórdãos como grafo de citações. Também serão apresentados os algoritmos de PageRank e de Floyd-Warshall, suas implementações e objetivos no contexto deste trabalho. Finalmente, as estatísticas e resultados obtidos serão expostos.

Capítulo 2

Extração e Persistência dos Dados

2.1 Fichas-resumo dos acórdãos

Os dados utilizados neste trabalho são públicos e podem ser acessados por ferramentas de buscas nos endereços eletrônicos dos tribunais [6][7]. Consistem de fichas eletrônicas dos acordãos do STF e do STJ entre 2001 e 2013. Elas apresentam um resumo organizado, em seções, das informações essenciais das decisões dos tribunais. A estrutura das fichas é similar em ambos os tribunais e compreendem um código único, que neste trabalho chamamos de ID, data de julgamento, relator, indexação, ementa, decisão, citações a legislação, a doutrinas e a outros acórdaos, entre outras informações. As fichas do STF ainda informam as partes envolvidas e os acordãos similares.



Processo

REsp 556398 / SP
RECURSO ESPECIAL
2003/0105654-6

Relator(a)

Ministro PAULO GALLOTTI (1115)

Órgão Julgador

T6 - SEXTA TURMA

Data do Julgamento

07/10/2003

Data da Publicação/Fonte

DJ 27/09/2004 p. 396

Ementa

PREVIDENCIÁRIO. APOSENTADORIA POR TEMPO DE SERVIÇO E AUXÍLIO-ACIDENTE. CUMULAÇÃO.
1. Diante do disposto na Lei nº 9.528/97, a verificação da possibilidade de cumulação do auxílio-acidente com aposentadoria tem de levar em conta a lei vigente ao tempo do infortúnio que ocasionou a incapacidade laborativa.
2. Não mencionando o acórdão impugnado que a incapacidade tivesse ocorrido em data anterior, não há como reconhecer o direito à pleiteada cumulação.
3. Recurso especial provido.

Acórdão

Vistos, relatados e discutidos estes autos, acordam os Ministros da Sexta Turma do Superior Tribunal de Justiça, na conformidade dos votos e das notas taquigráficas a seguir, por unanimidade, dar provimento ao recurso, nos termos do voto do Sr. Ministro Relator.
Os Srs. Ministros Paulo Medina, Fontes de Alencar e Hamilton Carvalhido votaram com o Sr. Ministro Relator.
Presidiu o julgamento o Sr. Ministro Hamilton Carvalhido.

Notas

Veja a AR 4480-SP.

Figura 2.1: *Exemplo de uma ficha-resumo do STJ*

2.2 Web Scraping

Apesar dos progressos no processo de informatização do judiciário, o STF e o STJ não oferecerem os dados de suas jurisprudências em formatos otimizados para a manipulação por máquinas, como XML, CSV ou JSON. Portanto, foi necessário realizar Web Scraping para a extração dos dados a partir do contéudo HTML disponibilizado. *Web scraping* pode ser definida como uma técnica de extração de dados que geralmente tem como objetivo coletar dados não estruturados e transformá-los em um formato apropriado para um contexto fora do website original. A busca de informações é realizada por *web crawlers*, que podem inclusive imitar a exploração das páginas por um usuário humano.

Há diversos *softwares* que oferecem *frameworks* para *web scraping* mas, para este trabalho, o Scrapy[8] foi adotado. Esta escolha justifica-se pelo intuito de aproveitar a implementação da extração das decisões do STF do projeto inicial, que utilizava esta ferramenta, também por ser *open source* e oferecer *api* em linguagem Python, que é familiar à autora.

O Scrapy tem o seu próprio mecanismo de extração de dados, denominado *selector*, que possibilita a seleção de partes do documento HTML especificadas por Xpath ou CSS. Xpath[9], ou XML Path Notation, consiste de uma linguagem para navegar pelas tags do XML, mas também pode ser usada para documentos HTML. A implementação do *web crawler* e do pós processamento dos itens extraídos é feita por meio de classes Python.

O código do *web crawler* desenvolvido para os dados do STF aproveitou a implementação do projeto inicial, porém, neste trabalho todas as seções foram identificadas e inseridas no banco de forma estruturada. As citações também foram restringidas apenas a seção de citação dos acórdãos. Apesar das fichas dos acórdãos do STF e do STJ possuírem estruturas similares, seus códigos HTML são diferentes e, portanto, foi necessário desenvolver um segundo *web crawler* dedicado ao STJ. Também, inicialmente, o *download* das informações em arquivos de texto era realizado, para somente depois realizar a extração das informações e inserçao no banco. Neste trabalho, a extração e persistência de informações foram integradas.

Os web crawlers imitam um usuário ao preencher os filtros de data do mecanismo de

busca de jurisprudências disponiblizado nos sites dos órgãos[6][7]. A busca apresenta como resultado as fichas-resumo dos acórdãos. A extração de informação vale-se da estrutura bem definida em seções das fichas. Apesar das seções não seguirem sempre a mesma ordem, elas sempre apresentam os mesmos títulos e *tags* HTML específicas, possibilitando a identificação e extração dos dados por *selectors*.

Algumas seções contém mais de uma informação, logo necessitam de uma extração mais profunda. Por exemplo, o cabeçalho dos acórdãos contém o ID, ministro relator, data de julgamento e unidade federativa do acórdão. A seção que descreve os nomes e a função de cada parte envolvida nos processos do STF também é mais complexa, porque a mesma função pode ser escrita de formas diferentes, por exemplo: advogado(s), advg(s) e advg(o/a)(s). Por isso utilizou-se uma tabela de tradução para normalizar as funções. Deste modo é possível recuperar os advogados, promotores, requerentes e requeridos de cada processo diretamente. A citação à legislação, indica partes específicas das leis: os artigos, parágrafos, incisos e alíneas a qual se referem. Essas peculiaridades foram identificadas por expressões regulares e inseridas de forma estruturada no banco de dados.

Os códigos identificadores dos acórdãos são compostos pelo tipo e pelo número do processo, mas seguem convenções diferentes em cada tribunal. Particularmente, a variação está na ordem dos tipos de processos que aparecem no ID. Frequentemente, são julgadas questões sobre outras decisões, como, por exemplo, um recurso extraordinário sobre um ato constitucional. O site do STJ etiqueta as decisões desse tipo como *RE AI NumeroDoProcesso*, já o STF os nomeia com o ID *AI RE NumeroDoProcesso*. Como se quer trabalhar com dados padronizados, tanto os IDs principais, ao qual a ficha se refere, quanto os IDs citados ou similares são convertidos para o padrão do STJ.

Ao final da extração de cada ficha, as informações são aglutinadas em um objeto do tipo dicionário, uma implentação Python de estrutura chave-valor, na qual os valores podem ser de qualquer tipo, inclusive outros dicionários. Uma classe pré configurada do Scrapy, para o pós processamento dos itens extraídos, foi customizada para persistir as informações do dicionário da ficha.

2.3 Persistência

As informações dos acórdãos foram persistidas para poderem ser consultadas e analisadas posteriormente. Como os dados são basicamente fichas, um banco de dados não relacional orientado a documentos foi adotado para este fim.

O termo *NoSQL* se aplica para se referir aos bancos de dados não relacionais que oferecem uma alternativa aos tradicionais sistemas de gerenciamento de banco de dados. Este modelo surgiu, principalmente, da necessidade de tratar volumes enormes e crescentes de dados (*big data*), contudo também é apropriado em outros contextos devido a suas características[10]:

- Escalabilidade, rendendo alta performance também em grandes *clusters*;
- Não tem um esquema definido de dados, *Schemeless*, oferecendo flexibilidade no desenvolvimento e manutenção das aplicações;
- Facilidade de uso e inicialização rápida porque oferecem APIs simples ao invés da linguagem SQL e sua modelagem é mais natural;
- Consistência eventual, que resulta em melhor desempenho e alta disponibilidade;
- A maioria de suas implementações são *open source*.

Fawler[11] acredita que o modelo SQL é e continuará a ser o mais popular e usado na maioria das situações, porém já não são uma decisão automática. Assim, os bancos não relacionais contribuem para uma nova era baseada na persistência poliglota, em que cada parte das aplicações poderão usufluir de um tipo de banco específico, adequado a sua funcionalidade e a natureza de seus dados.

Há quatro modelos de bancos não relacionais: chave-valor, coluna-família, orientado a documentos e orientado a grafos sendo que, os três primeiros baseiam-se em objetos agregados. O MongoDB[12] vem do inglês *humongous* (enorme) e é o banco não relacional orientado a documentos mais popular. Foi utilizado neste projeto por ser *open source* e oferecer *APIs* para diversas linguagens, inclusive Python.

As consultas do MongoDB são baseadas em Javascript e os documentos tem formato BSON, que permite a representação de dados de forma natural, por meio de aninhamentos

em hierarquias complexas. O formato JavaScript Object Notation, JSON[13], foi criado para o armazenamento e troca de informações e apresenta-se basicamente com uma estrutura de chave-valor. Baseado em Javascript, porém armazenado como texto simples, oferece uma forma natural e legível a seres humanos ao mesmo tempo em que é facilmente analisado e gerado por computadores. O formato BSON [14] é uma representação binária do JSON, utilizada pelo MongoDB para armazenar seus documentos e que lhe provê rápida escaneabilidade dos dados.

Deste modo, um documento em MongoDB pode ser descrito como um conjunto de campos no formato chave-valor que se referem a um mesmo objeto. Já as coleções são conjuntos de documentos. É interessante também notar que os documentos não necessitam seguir um esquema fixo (*schemeless*) e podem guardar quaisquer tipos de dados.

Particularmente, a biblioteca Pymongo[15] oferece uma *API* simples, em linguagem Python e é usada tanto para inserção, quanto para a manipulação dos dados do banco. Deste modo, a biblioteca Pymongo possibilitou a integração entre a ferramenta Scrapy e o MongoDB.

Foram coletados pouco mais de 51 mil acórdãos do STF e 280 mil acórdãos do STJ. Os dados foram persistidos na máquina logprob em uma coleção na qual cada documento representa uma ficha. O conteúdo de cada ficha é guardado na íntegra e as informações derivadas também, em campos dedicados e por vezes aninhados do documento, como o caso da referência a legislação.

Capítulo 3

Grafo de citações e algoritmos

As citações à outras decisões são significativas no mundo jurídico e servem para fundamentar as decisões proferidas. Os acórdãos podem ser modelados como um grafo dirigido, em que cada acórdão representa um vértice e cada citação é uma aresta que o liga a outro acórdão. Sobre esta rede é possivel aplicar diversos algoritmos com a intenção de investigar as relações de citações entre os acórdãos. Particularmente, seria interessante poder inferir as decisões mais relevantes a fim de tentar caracterizar o comportamento dos tribunais. Para este fim, o algoritmo PageRank foi utilizado.

3.1 PageRank

3.1.1 Descrição

O algoritmo PageRank foi publicado por Lawrence Page e Sergey Brin, fundadores do Google em 1998[16], e consistiu do método base para o mecanismo de busca mais bem sucedido na web. O objetivo do algoritmo é obter um *ranking* das páginas mais relevantes para uma ordenação satisfatória dos resultados de busca. Fundamenta-se sobre a idéia de que a relevância de uma página é proporcional ao número de *hyperlinks* e a relevância de páginas que tem *hyperlinks* apontando para ela.

O pageRank de uma página é um valor numérico que mensura sua relevância e, em sua forma mais simpes, é calculado pela seguinte fórmula:

$$PR(a) = c \sum_{b \in L_a} \frac{PR(b)}{N_b}$$

onde a é uma página qualquer, L_a é o conjunto das páginas que apontam para a, N_b é a quantidade de *hyperlinks* da página b e c é uma constante de normalização.

Podemos observar que a contribuição ao valor do pageRank pelos *hyperlinks* se dá, não somente pela relevância de suas páginas de origem, mas também é influenciada pela quantidade total de *hyperlinks* a qual a página de origem aponta.

Os valores de pageRank também podem ser interpretados como a probabilidade de um usuário acessar uma página navegando por *hyperlinks* aleatoriamente. Para representar a probabilidade de um usuário acessar uma página sem utilizar os *hyperlinks* e também para lidar com os potenciais problemas de ciclos e páginas sem *hyperlinks*, a constante p foi incorporada ao cálculo do pageRank da seguinte forma:

$$PR(a) = p \sum_{b \in L_a} \frac{PR(b)}{N_b} + \frac{1-p}{N}$$

Como na época a Web já continha milhares de páginas, o algoritmo foi desenvolvido de forma iterativa, de maneira que mudança ou a inserção de novos *hyperlinks* pudessem ser computados a partir dos cálculos já existentes. Então, segue uma série de iterações onde os novos valores de pageRank são atualizados para cada página até a condição de parada ser satisfeita. A condição de parada pode ser implementada de diversas formas, por exemplo estabelecendo um período de tempo específico ou até que a diferença entre os pageRanks de iterações consecutivas seja muito pequena[17].

Apesar de ter sido projetado para a web, o algoritmo pode ser aplicado a outros grafos. No contexto dos acórdãos, os nós ou páginas podem ser vistas como os acórdãos e as arestas ou *hyperlinks*, são as citações. Isto posto, o algoritmo foi implementado para o grafo dos acórdãos em linguagem Python, utilizando os dados obtidos com a extração das fichas.

3.1.2 Implementação

No início do algoritmo, todos os acórdãos ganham pageRanks iguais e o critério de parada é a distância euclidiana entre os pageRanks de iterações consecutivas. O projeto inicial já trazia esta implementação, e seus resultados foram apresentados a especialistas do Direito, os quais sugeriram que se um acórdão faz muitas citações, isso não implica, necessariamente, que o peso de suas citações para o cálculo do pageRank deve ser menor. Então, uma versão modificada do algoritmo foi implementada, na qual o pageRank de uma decisão só depende da relevância dos acórdãos que a citam. Desta forma o cálculo base da versão modificada é:

$$PR(a) = c \sum_{b \in L_a} PR(b)$$

O código desenvolvido em Python faz uso da biblioteca Pymongo, mais uma vez. Primeiro, são construídos dicionários com as informações do banco que serão utilizadas. Todos os dicionários são indexados pelo código identificador de acórdãos. Um dicionário guarda um objeto da classe Acórdão com informações que se queira persistir além do pageRank, como por exemplo o tribunal e relator do acórdão. Os outros dicionários guardam os IDs dos acórdãos que citam e são citados pelos acórdãos da chave. A construção dos dicionários é feita pela classe GraphMaker que pode receber *queries* do banco para considerar subgrafos de citações. A classe PageRank carrega a implementação do algoritmo e dá a opção da versão original e da modificada. Essa independência entre os dados e o algoritmo facilita a geração de pageRanks para quaisquer conjuntos de acórdãos.

3.2 Floyd Warshall

Um estudo da topologia de um grafo pode ajudar a entender como as relações entre os nós se comportam e também medir a qualidade dos dados. No caso particular dos acórdãos, é interessante verificar se existem citações cíclicas, o que poderia influenciar por exemplo, o resultado do algoritmo de PageRank. Então, o algoritmo de Floyd-Warshall foi implementado para este proprósito.

Também conhecido como algorítimo de Floyd, Roy-Warshall, Roy-Floyd ou WFI, Floyd-Warshall encontra os caminhos mais curtos, ou menos custosos, entre todos os pares de vértices de um grafo. Robert Floyd formalizou o algoritmo em 1962[18], porém o algoritmo é equivalente as descobertas de Roy e Warshall para detectar o fechamento transitivo de um grafo, por isso a associação a diversos autores.

O algoritmo aplica programação dinâmica para comparar todos os caminhos possíveis entre pares de vértices em tempo $O(n^3)$. Funciona incrementalmente, melhorando a estimativa dos caminhos mais curtos a cada iteração até encontrar seu valor ótimo. Primeiramente, os vértices são numerados de 1 a N, onde N é a quantidade de nós do grafo. A cada iteração k, o algoritmo acha os caminhos mais curtos para todos os vértices usando os vértices entre 1 e k. Considere p o caminho mais curto entre i e j, com vértices intermediários restritos ao conjunto [1,...,k]. O algorítimo explora a relação entre p e os caminhos mais curtos entre i e j usando vértices de [1,...,k-1]:

1. Se p não passar por k, então p usa somente vértices entre [1...,k-1]

2. Se p passar por k, então p é composto pelo caminho mais curto entre i e k e pelo caminho mais curto entre k e j usando somente vértices entre 1 e $k-1$

Seja $D^k_{i,j}$ o tamanho do caminho mais curto entre i e j usando somente vértices entre 1 e k, segue uma definição recursiva do algoritmo:

$$D^k_{i,j} = \begin{cases} \text{tamanho da aresta entre } i \text{ e } j & \text{se } k > 0 \\ \min(D^{k-1}_{i,j}, D^{k-1}_{i,k} + D^{k-1}_{k,j}) & \text{se } k >= 1 \end{cases}$$

Considerando tamanho infinito para arestas inexistentes[19].

Para o grafo dos acórdãos, o tamanho inicial de cada aresta é inicializado com 1. Um dicionário é novamento utilizado, no qual as chaves são os números atribuidos aos acórdãos. Os valores são outros dicionários cujas chaves são os números dos acórdãos citados e os valores são os tamanhos das arestas que representam as citações. Por exemplo, o grafo da figura 3.1 é representado como:

$$links[1] = \{3 : -2\}$$

$$links[2] = \{1 : 4, 3 : 3\}$$

$$links[3] = \{4 : 2\}$$

$$links[4] = \{2 : -1\}$$

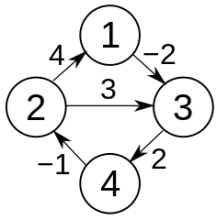


Figura 3.1: *Exemplo de grafo.*

Esta representação é proveitosa em relação a matricial, já que há muitos vértices e a matriz é esparsa. Esta representação ainda permite que somente os vértices ligados sejam percorridos, diminuindo o tempo de execução do algoritmo significativamente.

Tipicamente, o algoritmo determina apenas o tamanho dos caminhos mais curtos, mas a implementação foi modificada para determinar também os vértices que compõem os caminhos. Para isso, um dicionário de predecessores é atualizado sempre que um caminho mais curto é encontrado. O predecessor de i e j é o vértice diretamente anterior a j no caminho mais curto entre i e j. Seguindo os predecessores é possível reconstruir todo o caminho.

Calculados todos os caminhos, encontrar os ciclos é trivial. Para verificar se o vértice i pertence a um ciclo, basta verificar se algum vértice j citado por i, tem um caminho de tamanho finito para i.

Capítulo 4

Resultados

4.1 Estatísticas

4.1.1 Tipos de Acórdãos

As figuras 4.1, 4.2 e 4.3 ilustram a distribuição dos tipos mais representativos de processos jurídicos. Observa-se que os processos de recursos (RE e RESP) são predominantes em ambos os tribunais, o que confirma o caráter contestatório desses órgãos.

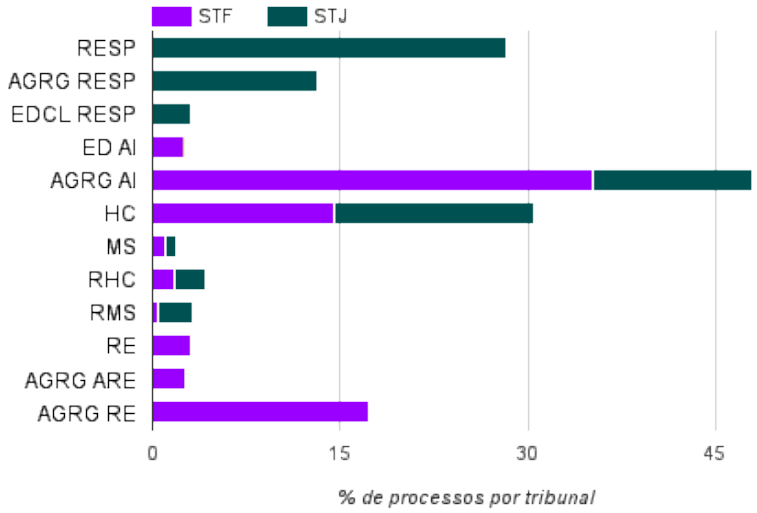


Figura 4.1: *Distribuição comparativa dos tipos de acórdãos do STF e do STJ*

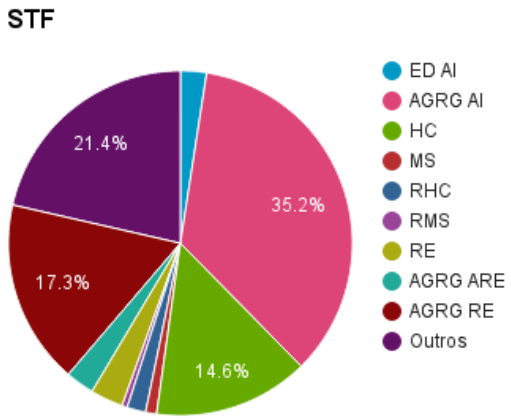


Figura 4.2: *Distribuição dos tipos de acórdãos do STF*

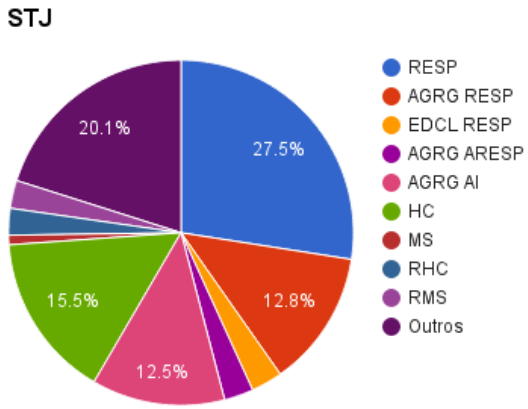


Figura 4.3: *Distribuição dos tipos de acórdãos do STJ*

4.1.2 Palavras mais Frequentes na indexação

As palavras mais empregadas como *tags* para os acórdãos estão representadas na tabela 4.1.

4.1.3 Citações a acórdãos

total de acórdãos: 51 mil STF + 285 mil STJ

total de citações: 1.007.163

número de acórdãos que citam: 208 mil

STF	STJ
ausência	hipótese
descabimento	caracterização
recurso extraordinário	decorrência
cível	objetivo
voto vencido	previsão
impossibilidade	impossibilidade
decorrência	necessidade

Tabela 4.1: *Palavras mais frequententes na indexação dos acórdãos do STF e do STJ*

média de citações por acórdão: 3

média de citações por acórdãos que citam: 4,8

acórdão que mais cita: RESP 1061530 com 113 citações

acórdão mais citado: ERESP 435835 com 1489 citações

4.2 Legislação

Referências a legislação são primordiais na fundamentação dos argumentos jurídicos. A tabela 4.4 indica as leis mais citadas nas decisões. Observa-se que há muitas referências a súmulas de regimentos internos dos tribunais. Também é interessante notar várias referências a Constituição Federal, por parte do STF, como era esperado.

Foram computadas cerca de 122 mil referências, a mais de 8 mil leis diferentes, resultando em uma média de 2.35 de leis apontadas por acórdão no STF. Para o STJ a média é de 1.85, com 513 mil citações a, aproximadamente, 10700 leis distintas.

4.3 PageRank

A tabela 4.2 lista os acórdãos de maior relevância encontrados pelo algoritmo de PageRank versão original. A predominância de acórdãos do STJ, quando considerados ambos os tribunais, pode se justificar pela maior quantidade de acórdãos do STJ, cerca de 250 mil em relação aos pouco mais de 50 mil acórdãos considerados do STF. De fato, ao observar a tabela 5.2 que indica os acórdãos mais relevantes segundo a versão do PageRank modificado, que leva em conta somente a relevância das citaçãoes, os acórdãos do STF se

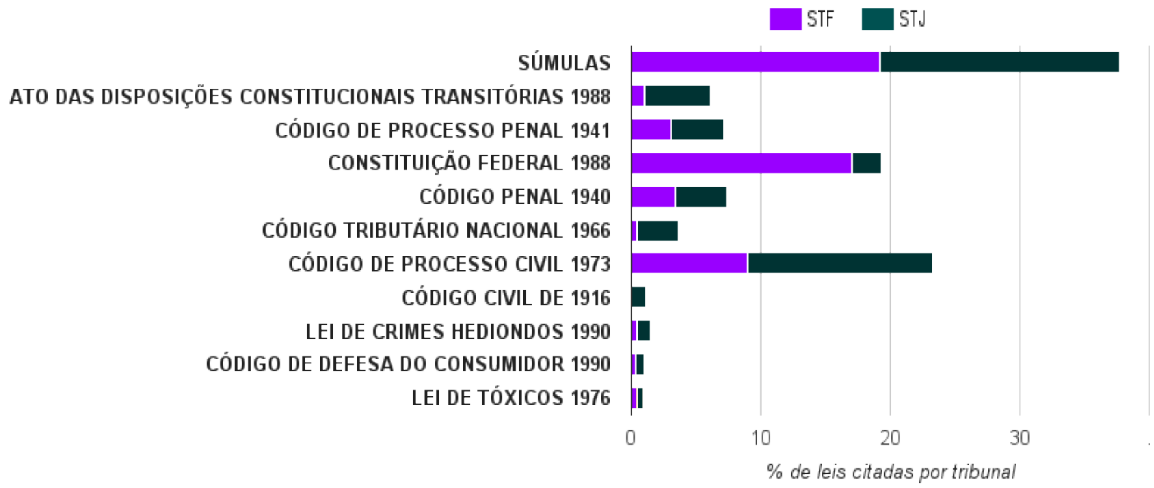


Figura 4.4: *Distribuição comparativa das leis citadas por acórdãos do STF e do STJ*

destacam.

STF	
ADC 4	1.246e-04
HC 82959	9.696e-05
RE 415454	7.571e-05
RE 416827	7.443e-05
ADI 1662	7.397e-05
RE 290079	7.228e-05
RE 258980	6.564e-05
ADI 3105	5.999e-05
HC 81611	5.925e-05
RE 420816	5.654e-05

STJ	
RESP 106888	1.725e-04
ERESP 435835	1.706e-04
RESP 198807	1.480e-04
RESP 271214	1.126e-04
RESP 248893	1.046e-04
RESP 407097	9.110e-05
RESP 298191	8.925e-05
AGRG RESP 744504	7.133e-05
ERESP 423994	6.412e-05
AI ERESP 644736	6.247e-05

ERESP 435835	1.273e-04	STJ
RESP 106888	1.191e-04	STJ
RESP 198807	1.022e-04	STJ
RESP 271214	8.430e-05	STJ
RESP 248893	7.819e-05	STJ
RESP 407097	6.818e-05	STJ
RESP 298191	6.670e-05	STJ
HC 82959	6.143e-05	STF
ERESP 423994	4.755e-05	STJ
AI ERESP 644736	4.591e-05	STJ

Tabela 4.2: *Ranking dos acórdãos pelo PageRank original*

HC 102088	8.440e-02	STF
HC 96202	8.440e-02	STF
HC 102080	8.440e-02	STJ
HC 111727	6.678e-02	STF
HC 110926	6.678e-02	STF
HC 111123	6.678e-02	STF
HC 110850	6.678e-02	STF
RHC 89624	5.627e-02	STF
HC 112319	5.284e-02	STF
HC 84412	3.557e-02	STF

Tabela 4.3: *Ranking dos acórdãos pelo PageRank modificado*

4.4 Ciclos

O algoritmo de Floyd-Warshall foi aplicado nos grafos do STF e no grafo conjunto dos tribunais para detectar ciclos de até 300 acórdãos.

No grafo conjunto foram identificados 1779 ciclos, porém, vários destes ciclos são equivalentes, isto é, são compostos pelos mesmos acórdãos. Não foi realizada uma análise para detectar a duplicidade dos ciclos. O maior ciclo encontrado é composto por 35 acórdãos, nominalmente:

$\rightarrow 66989 \rightarrow HC43402 \rightarrow HC20213 \rightarrow HC67710 \rightarrow HC59858 \rightarrow HC46291 \rightarrow HC43198 \rightarrow HC19406 \rightarrow HC77173 \rightarrow HC48611 \rightarrow RHC20901 \rightarrow HC77997 \rightarrow HC66182 \rightarrow HC43267 \rightarrow HC32949 \rightarrow HC27575 \rightarrow HC24505 \rightarrow HC59670 \rightarrow HC46596 \rightarrow HC46633 \rightarrow HC85900 \rightarrow HC58527 \rightarrow HC54802 \rightarrow HC68685 \rightarrow RHC18993 \rightarrow HC18813 \rightarrow HC32915 \rightarrow HC32537 \rightarrow HC30051 \rightarrow HC18985 \rightarrow RESP310936 \rightarrow HC118468 \rightarrow HC103259 \rightarrow HC72941 \rightarrow HC50875 \rightarrow$

No grafo do STF foram encontrados 365 ciclos, com no máximo 6 acórdãos, como por exemplo:

$\rightarrow ADI2050 \rightarrow ADI1901 \rightarrow ADI1628 \rightarrow ADI2220 \rightarrow ADI132 \rightarrow ADI102 \rightarrow$

Capítulo 5

Conclusões

A proposta deste projeto era de compilar análises que pudessem ser de utilidade pública. É difícil avaliar se o objetivo foi alcançado, porque é necessário um maior entendimento do mundo jurídico. Porém, foi possível a implementação dos algoritmos de forma a permitir análises sobre diferentes conjuntos de dados e diversas estatísticas podem ser geradas a partir dos dados extraídos.

5.1 Trabalhos Futuros

Ainda há muitas possibilidades de expansão e melhoria deste projeto. Uma das mais promissoras talvez seja desenvolver uma aplicação de visualização dinâmica dos grafos que permitisse a geração de estatísticas sobre um conjunto parametrizável de jurisprudências. Isso facilitaria o acesso aos dados por especialistas do Direito, que em contrapartida poderiam avaliar as medidas propostas e feedbacks para o projeto.

Interações recentes com os especialistas do domínio jurídico, esclareceram que grande parte dos acórdãos versam essencialmente sobre formalidades. Seria proveitoso conseguir separar esses acórdãos dos que decidem sobre o conteúdo das matérias de fato, afim de analisa-las separadamente. Neste sentido, é possível também analisar diferentes subconjuntos de decisões.

A extração de dados pode ser expandida para outros tipos de decisões, como as monocráticas e para outros tribunais como o Tribunal Superior do Trabalho e os Tribunais de

Justiça estaduais. Também seria interessante extrair a íntegra das decisões, já que algumas informações importantes, por vezes, estão ausentes nas fichas, como a descrição da decisão final.

5.2 Disciplinas Relevantes para o trabalho

- **MAC0328 - Algoritmos em Grafos:** ofereceu a base para o entendimento dos grafos, sua representação em estruturas de dados e modelagem a cenários reais
- **MAC0439 - Laboratório de Bancos de Dados:** auxiliou-me a entender os conceitos de banco de dados não relacionais
- **MAC0323 - Estrutura de Dados:** ensinou-me a manipular as estruturas de dados
- **MAC0441 - Programação Orientada a Objetos:** aprofundou conceitos de orientação a objeto

Referências Bibliográficas

[1] W3C, "Manual dados abertos." http://www.w3c.br/pub/Materiais/PublicacoesW3C/manual_dados_abertos_desenvolvedores_web.pdf, 2011. Último acesso em 16/09/2015. 1

[2] Supremo Tribunal Federal, "Sistema judiciário brasileiro: organização e competências." http://www.stf.jus.br/portal/cms/verNoticiaDetalhe.asp?idConteudo=169462, 2011. Último acesso em 01/10/2015. 1

[3] Conselho Nacional de Justiça, "Tribunais superiores: Quais são? o que fazem?." http://www.cnj.jus.br/noticias/cnj/59218-tribunais-superiores-quais-sao-o-que-fazem, 2012. Último acesso em 01/10/2015. 2

[4] A. Calò, "Extração e análise de informações jurídicas públicas." https://www.linux.ime.usp.br/~sandro/mac0499/Monografia.pdf, 2014. Último acesso em 15/11/2015. 2

[5] Tribunal Superior do Trabalho, "Vocabulário jurídico." http://www.tst.jus.br/vocabulario-juridico, 2015. Último acesso em 10/08/2015. 2

[6] "Pesquisa de jurisprudência stf." http://www.stf.jus.br/portal/jurisprudencia/pesquisarJurisprudencia.asp. Último acesso em 23/11/2015. 3, 5

[7] "Pesquisa de jurisprudência stj." http://www.stj.jus.br/SCON/. Último acesso em 23/11/2015. 3, 5

[8] Scrapy, "Scrapy at a glance." http://doc.scrapy.org/en/latest/intro/overview.html. Último acesso em 19/11/2015. 4

[9] W3C, "Xml path language (xpath) 3.1." http://www.w3.org/TR/xpath-31/, 2014. Último acesso em 10/11/2015. 4

[10] P. Sadalage, "Nosql databases: An overview." https://www.thoughtworks.com/insights/blog/nosql-databases-overview, 2014. Último acesso em 10/11/2015. 6

[11] M. Fawler, "Polyglot persistence." http://martinfowler.com/bliki/PolyglotPersistence.html, 2011. Último acesso em 10/11/2015. 6

[12] MongoDB, "The mongodb 3.0 manual." https://docs.mongodb.org/manual/, 2015. Último acesso em 10/11/2015. 6

[13] json.org, "Introducing json." http://json.org/, 2015. Último acesso em 25/11/2015. 7

[14] BSON, "Bson." http://bsonspec.org/, 2015. Último acesso em 25/11/2015. 7

[15] MongoDB, "Pymongo 3.1.1 documentation." https://api.mongodb.org/python/current/, 2015. Último acesso em 25/11/2015. 7

[16] L. Page and S. Brin, "The pagerank citation ranking: Bringing order to the web." http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf. Último acesso em 07/11/2015. 9

[17] L. Page and S. Brin, "The anatomy of a large-scale hypertextual web search engine." http://infolab.stanford.edu/~backrub/google.html. Último acesso em 07/11/2015. 10

[18] R. W. Floyd, "Algorithm 97: Shortest Path," *Communications of the ACM*, vol. 5, p. 345, June 1962. 12

[19] T. H. Cormem, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. The MIT Press, 2nd ed., 1990. 12