Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

Extração e Análise de Informações Jurídicas Públicas

2014
Alessandro Calò
Orientação: Prof. Dr. Marcelo Finger

Resumo

A modernização da computação levou à criação de grandes conjuntos de dados em todas as áreas de conhecimento. Grande parte destas informações é livremente acessível para pesquisa, mas não está otimizada para análises cujo escopo excede cada documento individualmente. Um importante exemplo é dado pelos acórdãos do Supremo Tribunal Federal. Os seus textos estão disponíveis para consulta por parte de qualquer cidadão, mas uma análise do conjunto inteiro dos processos ainda requer um extenso processamento destes dados para que possam ser gerados resultados significativos.

Adicionalmente, os acórdãos possuem uma característica importante, permitindo análises ainda mais profundas: eles citam uns aos outros. É possível identificar referências a outros processos, gerando uma série de resultados importantes. A análise estatística deste grande volume de dados pode, portanto, revelar muitas informações sobre os mesmos e sobre o Judiciário brasileiro: torna-se possível uma visão grande-angular sobre o tema, revelando propriedades interessantes e de grande utilidade pública. Este trabalho analisa as decisões de última instância e as suas conexões para gerar dados inéditos.

Mostra-se como foram obtidos os textos dos acórdãos, e como deles foram extraídas e, em seguida, persistidas as informações relevantes. Em seguida, detalha-se o funcionamento do Page Rank, um importante algoritmo que foi aplicado sobre os dados. Mostra-se os resultados obtidos a cada passo e ao final exibem-se algumas estatísticas interessantes.

Palavras-chave: STF, acórdão, citação, web scraping, page rank

Abstract

Advancements in the field of Computer Science have led to the creation of large datasets in all areas of knowledge. A large quantity of information is freely accessible for research purposes, but it is not optimised for analyses for which the scope is larger than each individual document. An important example is given by the rulings of Brazil's Supreme Court (*Supremo Tribunal Federal*). Its texts are available for research, but an analysis of the entire set of rulings requires extensive data processing, so that significative results can be generated.

Moreover, these rulings present an important characteristic, allowing for even more meaningful analyses: they reference each other. It is possible to identify links to other rulings, generating a series of important results. Therefore, the statistical analysis of this large dataset can reveal a great deal of information about itself and also about Brazil's Judiciary: it enables a broader view on the subject, revealing interesting properties that are of great public utility. This dissertation analises the aforementioned rulings and their connections in order to generate unprecedented results.

It shows how the texts were obtained, and how relevant information was extracted from them and then persisted. Then, it details the mechanism for Page Rank, and important algorithm that was run against such data. Results are shown after each step and at the end some interesting statistical results are presented.

Keywords: supreme court, ruling, citation, web scraping, page rank

Capítulo 1: Introdução

1. Contexto

A evolução da computação, nos últimos anos, incentivou a criação de grandes volumes de dados em todas as áreas de conhecimento. Observou-se um grande esforço no sentido de digitalizar, organizar e disponibilizar o histórico de organizações, pesquisas, e processos produtivos de inúmeros campos. Atualmente, grande parte destes dados encontra-se disponível para consulta, porém cada elemento tem pouca relevância quando analisado individualmente. Desde então, foram concebidos novos métodos de análise cuja finalidade principal é gerar novos significados a partir de grandes quantidades de dados brutos.

Consequentemente, a computação começou a ser usada em áreas que usualmente não eram sujeitas a tais tipos de análise. Foram desenvolvidas novas estruturas de dados e algoritmos para lidar com diferentes relações existentes entre os dados. Foram aplicados modelos estatísticos para melhor visualizar propriedades dos conjuntos de informações. Percebeu-se que processamento, filtragem e organização podem gerar resultados surpreendentes.

Um caso típico de aplicação destas técnicas é a integração de diversas bases de dados de uma instituição. Por exemplo, em hospitais, ao integrar os dados de diversos departamentos (i.e. oncologia, neurologia, etc.) é possível obter um quadro geral da situação de um paciente, elaborada por diferentes médicos e compreensível para mais pessoas. Além disso, uma análise estatística sobre tal quadro pode evidenciar — ou até prever — interferências entre tratamentos em diversas áreas, como efeitos colaterais ou anulação de princípio ativo.

Outro importante exemplo refere-se ao caso do Judiciário Brasileiro. Nos últimos anos, o setor vem implementando uma série de modernizações, uma delas sendo a digitalização dos textos dos acórdãos por parte do Supremo Tribunal Federal[1]. O site do Tribunal permite consultar dezenas de milhares de processos. Eles foram catalogados e os seus conteúdos foram divididos em seções: “Partes”, “Ementa”, “Indexação”, entre outras. Os usuários podem fazer pesquisas complexas usando diversos campos como “Ministro”, “Órgão Julgador”, etc. ou fazer uma pesquisa livre usando operadores lógicos.

Uma interessante característica dos acórdãos é que eles se referenciam. Uma decisão pode ser feita em base a outra decisão tomada anteriormente em um acórdão similar, ou em base a uma lei. O acórdão e a legislação citados constam no texto do acórdão que os cita.

Isto é feito porque a citação é uma parte importante da decisão tomada, e considera-se que as leis ou acórdãos citados são tão relevantes quanto o acórdão consultado. Naturalmente, esta hierarquia lógica tem uma grande importância por si só, porém um olhar computacional sobre o tema revela que há muito mais a ser descoberto sobre o comportamento do Judiciário Brasileiro do que é encontrado em cada texto.

Como sugerido pelos exemplos citados anteriormente, a análise estatística de grandes volumes de dados pode revelar muitas informações sobre os mesmos. Torna-se possível uma visão grande-angular sobre o tema, revelando propriedades interessantes e gerando resultados de grande utilidade pública. O universo dos acórdãos do STF não é exceção. Este trabalho propõe-se a analisar as conexões entre decisões de última instância do Judiciário para gerar dados inéditos, de utilidade imediata. Refere-se a este conceito como "utilidade pública" do conhecimento: o objetivo da pesquisa é ajudar a medir e entender as atividades do setor, para gerar resultados úteis à população.

É importante notar a relevância social do trabalho. Todos os dados utilizados para a pesquisa são públicos, disponíveis integralmente online, assim como os algoritmos e estruturas de dados usados para obtê-los. O que for desenvolvido especificamente para este estudo, juntamente com os todos os resultados obtidos, ficará disponível para consulta e utilização no formato mais conveniente possível. Há uma preocupação especial em tornar tais resultados acessíveis em termos de linguagem, especialmente em uma área com vocabulário tão particular quanto o Direito.

Além da utilidade pública, deve-se notar a existência da utilidade privada da pesquisa: as modificações e adaptações dos algoritmos utilizados poderão ser de interesse privado à área da Ciência da Computação, em oposição ao público em geral.

2. Motivação

A escolha deste tema foi naturalmente motivada pela recente disponibilização dos acórdãos por parte do STF. Contudo, o motivo principal da escolha deste âmbito para a pesquisa deve-se ao fato que este é um cenário adequado para o tipo de análise feita: o volume de dados é grande, e o texto de cada acórdão tem pouca relevância por si só. Somente através de um olhar global sobre o conjunto de todos os

processos, pode-se observar as suas tendências, estudar a sua natureza e fazer previsões em relação ao futuro do setor.

O direito é uma área cujo domínio sempre dependeu de um conhecimento extensivo sobre o tema estudado: para prever resultados de processos, advogados e juízes devem conhecer as tendências, o histórico e outras características de julgamentos relacionados. Notoriamente, o vocabulário (ou "tesauro") presente nos textos é de difícil compreensão para leigos. Por isso, é interessante que seja possível extrair conhecimento sem prescindir de grande domínio prévio dos temas abordados.

Adicionalmente, é estimulante trabalhar em prol de um resultado mais abrangente do que a pesquisa em si: a utilidade pública e privada dos resultados obtidos oferece grande motivação para a realização (e, posteriormente, avanço) do projeto.

Além disso, este é um interessante exercício multidisciplinar, em dois sentidos. Em primeiro lugar está a multidisciplinaridade externa com as áreas do direito e da estatística: é necessário pesquisar temas muito além do que é normal para um estudante de computação, e procurar obter auxílio de pessoas cujo escopo de trabalho é completamente diferente. Tarefas diferentes da escrita de código tomam uma importância muito grande. Em segundo lugar está a multidisciplinaridade interna, entre os diversos campos da computação a serem utilizados na pesquisa (linguística computacional, *web scraping*, teoria dos grafos, busca de padrões, bancos de dados, etc.). O desafio de integrá-los em um sistema coeso e produzir resultados eficientemente é, certamente, muito estimulante.

3. Objetivo

O objetivo principal deste trabalho é produzir resultados interessantes, inéditos e úteis para a população. Deseja-se que eles sejam de interesse social, permitindo entender melhor decisões que podem nos afetar diretamente, trazendo assim o judiciário mais perto do público em geral.

Naturalmente, outro objetivo importante é aprender a utilizar as ferramentas necessárias para a obtenção e processamento dos resultados. Como mencionado anteriormente, são utilizados conhecimentos de diversas áreas da computação, e é necessário integrar os resultados eficientemente.

Capítulo 2: Obtenção dos Dados

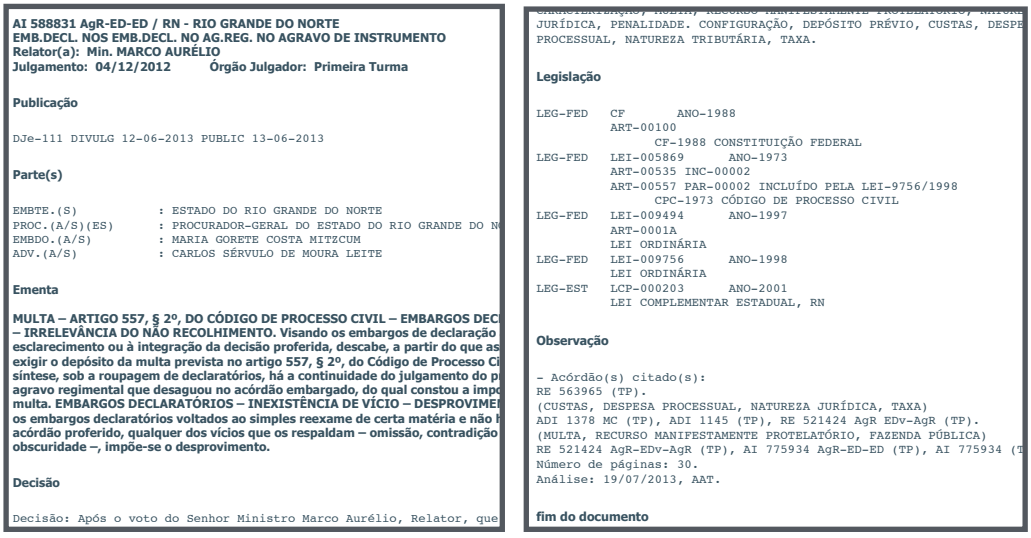
1. Formato dos dados

1.1 Acórdãos

Um acórdão é uma decisão do órgão colegiado de um tribunal. No caso deste projeto, trata-se do Supremo Tribunal Federal, doravante referenciado pela sigla STF. Concebido em 1824, após a proclamação da República e outorga da primeira constituição brasileira por Dom Pedro I, é a mais alta instância do poder judiciário brasileiro. Entre as diversas competências do órgão, a principal é o "controle de constitucionalidade", ou seja, a verificação da conformidade de uma lei em relação à constituição[2].

Por "órgão colegiado" entende-se um grupo de juízes cuja função é julgar, conjuntamente ou não, uma série de processos. Para o STF, este conjunto é composto por seus onze ministros. Apesar de serem assim denominados, a sua função assemelha-se mais à de um juiz do que à de um ministro de órgão de governo (por exemplo, o ministro da saúde).

Um acórdão do STF, portanto, é uma decisão tomada por um dos ministros do tribunal. Em outras palavras, é uma representação resumida da conclusão a que se chegou, não abrangendo toda a extensão e discussão em que se pautou o julgado.



AI 588831 AgR-ED-ED / RN - RIO GRANDE DO NORTE
EMB.DECL. NOS EMB.DECL. NO AG.REG. NO AGRAVO DE INSTRUMENTO
Relator(a): Min. MARCO AURÉLIO
Julgamento: 04/12/2012 Órgão Julgador: Primeira Turma

Publicação

DJe-111 DIVULG 12-06-2013 PUBLIC 13-06-2013

Parte(s)

EMBTE.(S) : ESTADO DO RIO GRANDE DO NORTE
PROC.(A/S)(ES) : PROCURADOR-GERAL DO ESTADO DO RIO GRANDE DO N
EMBDO.(A/S) : MARIA GORETE COSTA MITZCUM
ADV.(A/S) : CARLOS SÉRVULO DE MOURA LEITE

Ementa

MULTA – ARTIGO 557, § 2º, DO CÓDIGO DE PROCESSO CIVIL – EMBARGOS DEC – IRRELEVÂNCIA DO NÃO RECOLHIMENTO. Visando os embargos de declaração esclarecimento ou à integração da decisão proferida, descabe, a partir do que as exigir o depósito da multa prevista no artigo 557, § 2º, do Código de Processo Ci síntese, sob a roupagem de declaratórios, há a continuidade do julgamento do p agravo regimental que desaguou no acórdão embargado, do qual constou a impo multa. EMBARGOS DECLARATÓRIOS – INEXISTÊNCIA DE VÍCIO – DESPROVIME os embargos declaratórios voltados ao simples reexame de certa matéria e não acórdão proferido, qualquer dos vícios que os respaldam – omissão, contradição obscuridade –, impõe-se o desprovimento.

Decisão

Decisão: Após o voto do Senhor Ministro Marco Aurélio, Relator, que

JURÍDICA, PENALIDADE. CONFIGURAÇÃO, DEPÓSITO PRÉVIO, CUSTAS, DESPE
PROCESSUAL, NATUREZA TRIBUTÁRIA, TAXA.

Legislação

LEG-FED CF ANO-1988
ART-00100
CF-1988 CONSTITUIÇÃO FEDERAL
LEG-FED LEI-005869 ANO-1973
ART-00535 INC-00002
ART-00557 PAR-00002 INCLUÍDO PELA LEI-9756/1998
CPC-1973 CÓDIGO DE PROCESSO CIVIL
LEG-FED LEI-009494 ANO-1997
ART-0001A
LEI ORDINÁRIA
LEG-FED LEI-009756 ANO-1998
LEI ORDINÁRIA
LEG-EST LCP-000203 ANO-2001
LEI COMPLEMENTAR ESTADUAL, RN

Observação

- Acórdão(s) citado(s):
RE 563965 (TP).
(CUSTAS, DESPESA PROCESSUAL, NATUREZA JURÍDICA, TAXA)
ADI 1378 MC (TP), ADI 1145 (TP), RE 521424 AgR EDv-AgR (TP).
(MULTA, RECURSO MANIFESTAMENTE PROTELATÓRIO, FAZENDA PÚBLICA)
RE 521424 AgR-EDv-AgR (TP), AI 775934 AgR-ED-ED (TP), AI 775934 (T
Número de páginas: 30.
Análise: 19/07/2013, AAT.

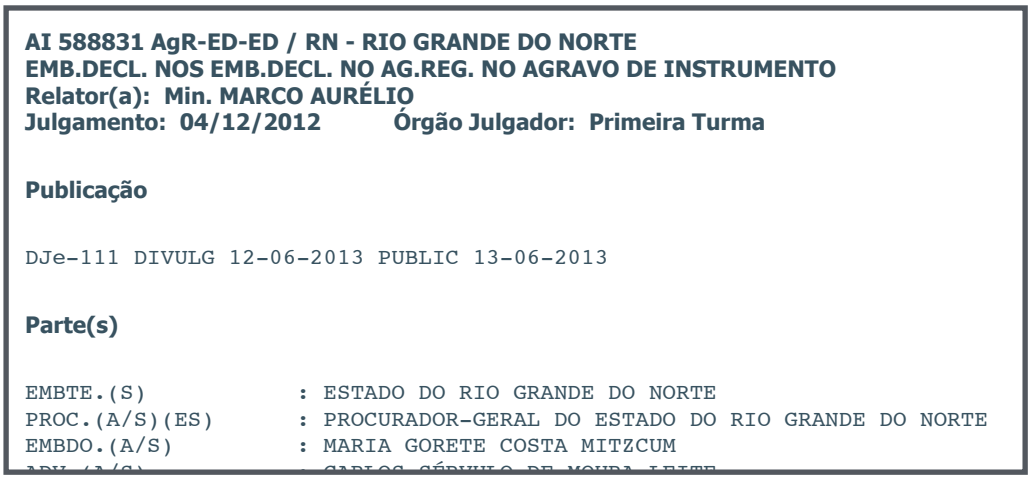
fim do documento

Imagem: Exemplo de acórdão do STF

O texto de um acórdão é dividido em seções, para melhor categorizar o conteúdo e permitir um certo nível de padronização. Há diversas seções, e nem todas estão presentes em todos os acórdãos. Os títulos mais comuns são: “Ementa” (descrição do desfecho do processo), “Partes” (enumeração dos réus, procuradores, advogados, etc.), “Indexação” (lista de palavras-chave) e “Legislação” (lista de leis sobre as quais o acórdão trata). Há ainda uma seção adicional, sem título, no começo de cada acórdão, análoga a um cabeçalho.

1.2 Elementos de um acórdão

No texto de um acórdão do STF, podem-se identificar alguns elementos comuns. O primeiro deles é um nome identificador, uma sequência de letras e números que identifica univocamente o acórdão em questão. Cada acórdão também possui alguns atributos informativos, como o estado do qual provém o julgamento, o nome do ministro que o redigiu e a data. Estas informações estão no cabeçalho do texto.



AI 588831 AgR-ED-ED / RN - RIO GRANDE DO NORTE
EMB.DECL. NOS EMB.DECL. NO AG.REG. NO AGRAVO DE INSTRUMENTO
Relator(a): Min. MARCO AURÉLIO
Julgamento: 04/12/2012 Órgão Julgador: Primeira Turma

Publicação

DJe-111 DIVULG 12-06-2013 PUBLIC 13-06-2013

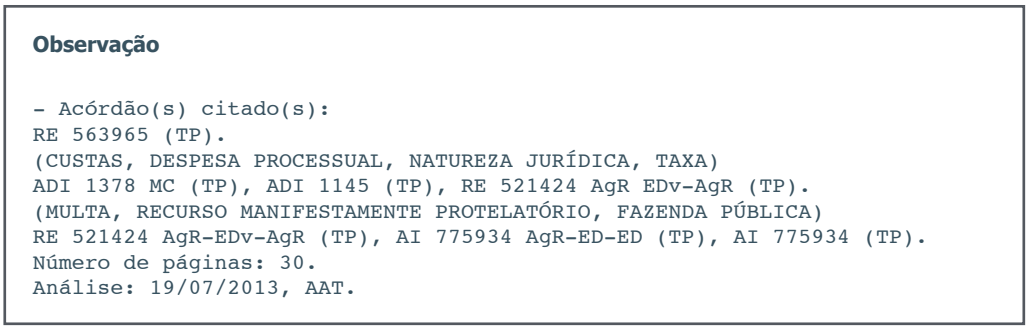
Parte(s)

EMBTE.(S) : ESTADO DO RIO GRANDE DO NORTE
PROC.(A/S)(ES) : PROCURADOR-GERAL DO ESTADO DO RIO GRANDE DO NORTE
EMBDO.(A/S) : MARIA GORETE COSTA MITZCUM

Imagem: cabeçalho de um acórdão do STF

Em seguida encontra-se, na seção “Indexação”, uma série de palavras-chave que categorizam o assunto do acórdão, permitindo obter um rápido entendimento sobre ele. Por fim, certa parte dos acórdãos possui uma seção denominada “Observação”, na qual há um parágrafo descrevendo os “acórdãos citados”. Trata-se de uma lista de identificadores (os mesmos descritos acima), útil para identificar

outros processos com os quais o processo em questão compartilha o assunto ou direcionamento jurídico. Supõe-se que, ao julgar a procedência de um processo, o ministro responsável baseie a sua decisão no resultado de outros, e então registre esse direcionamento sob forma de um acórdão citado.


Observação

```
- Acórdão(s) citado(s):
RE 563965 (TP).
(CUSTAS, DESPESA PROCESSUAL, NATUREZA JURÍDICA, TAXA)
ADI 1378 MC (TP), ADI 1145 (TP), RE 521424 AgR EDv-AgR (TP).
(MULTA, RECURSO MANIFESTAMENTE PROTELATÓRIO, FAZENDA PÚBLICA)
RE 521424 AgR-EDv-AgR (TP), AI 775934 AgR-ED-ED (TP), AI 775934 (TP).
Número de páginas: 30.
Análise: 19/07/2013, AAT.
```

Imagem: detalhe de um acórdão do STF

2. Download

O site do STF permite realizar pesquisas sobre os acórdãos[3], e os documentos resultantes são exibidos um após o outro, com um limite de dez por página. Não há um modo de fazer o download de todos ao mesmo tempo.

Por isso, para este projeto, ponderou-se a possibilidade de fazer uma requisição formal para o STF, de modo a obter os dados necessários para a análise de forma estruturada (por exemplo, um banco de dados ou uma lista de valores em formato CSV). Contudo, este processo é burocrático e de baixa prioridade, e até a data de publicação deste documento os dados não haviam sido enviados. Por isso, utilizou-se uma técnica conhecida como *web scraping*.

2.1 Web Scraping

Web scraping, ou *web crawling*, é o processo de coleta automática de informação a partir de páginas da internet. Técnicas desde tipo são usadas, por exemplo, para indexar e categorizar websites, tornando-os acessíveis a partir de ferramentas de busca, como o Google. Em outros casos, o processo é utilizado para ler, estruturar e fazer download de

informações específicas, para que possam ser persistidas localmente. O poder deste método jaz na sua capacidade de navegar pelas *tags* HTML de uma página e extrair somente as informações desejadas.

Foi utilizada uma ferramenta chamada Scrapy[4]: um *framework* de código aberto que permite a criação de *spiders*[5], responsáveis por acessar, formatar e fazer o download dos dados. Cada *spider* é uma instância de uma classe Python, escrita pelo usuário, capaz de navegar pela página desejada, usando XPaths[6] para acessar hyperlinks e extrair partes específicas. As informações podem então ser adicionadas, em forma de campos, a um item. O item então pode ser exportado em diversos formatos, entre os quais JSON e CSV.

2.2 Aplicação

Para poder realizar análises significativas, é necessário ter um grande número de documentos. O *scraping* foi feito então sobre as páginas de resultados oriundas de uma busca feita por todos os acórdãos publicados entre 2001 e 2012. Foram assim obtidos pouco mais de cinquenta mil acórdãos.

Após obter resultados satisfatórios para um subconjunto pequeno dos dados, a *spider* foi aplicada a todos os acórdãos, e então ocorreu um erro significativo. Alguns acórdãos têm seções ordenadas de maneira diferente, o que impede que elas sejam distinguíveis uma da outra no momento de navegar as *tags* HTML. Por isso, os itens mencionados anteriormente não puderam ser criados, e optou-se por fazer o download do texto integral dos acórdãos executando somente a remoção de todo o código HTML.

Finalmente, foi gerado um arquivo de texto para cada acórdão. Os arquivos foram armazenados no servidor LogProb do IME para análise, como será descrito a seguir.

3. Parsing

3.1. Implementação

Para extrair as informações necessárias do texto de cada acórdão, foi escrito um *parser* em Java. A escolha da linguagem foi feita em base a dois critérios. Em primeiro lugar, seriam feitas diversas operações com strings, e a biblioteca `java.lang.String` é extensa e tem suporte a expressões regulares, de uma maneira familiar ao autor. Em segundo lugar, a linguagem é organizada e permite fácil entendimento a outros leitores do código. Naturalmente, as principais convenções da linguagem (e portanto de Orientação a Objetos) foram usadas para este fim: classes, *getters*, *setters*, construtores, sobrescrita, etc.

O programa realiza um pré-processamento do texto de cada acórdão. Em primeiro lugar, são identificadas as linhas onde há o título de uma seção, para cada uma, é construído um objeto `Secao`, que tem um título (atribuído pelo seu construtor na hora da sua criação) e uma lista de linhas. Em seguida, cada linha que não é um título de seção é adicionada à lista de linhas da sua seção (a última seção instanciada). Para efeito de padronização, foi criado um título fictício à primeira seção, pois ela não possui um e contém informações importantes. O nome dado foi, naturalmente, “Título”.

Deste modo, tem-se uma classe que possui diversas seções, e cada seção tem diversas linhas. Fica, então, a cargo de cada seção procurar e retornar as informações pedidas. Por exemplo, para obter o identificador de cada processo, basta acessar a primeira linha da seção “Título”, e recortar a palavra que antecede a primeira barra, removendo espaços adicionais. Para obter a seção título e pedir a ela o identificador, basta executar:

```
secoes.get("Titulo").getID();
```

e a seção correspondente executará a seguinte linha:

```
lines.get(0).split("/")[0].trim();
```

que é concisa e simples de entender. A extração de outras informações pontuais (estado, relator, data) é feita de maneira similar.

Para obter a lista de palavras-chave (ou "rótulos"), o processo é análogo, porém uma camada a mais de processamento foi necessária. Antes de tudo, todas as linhas da seção "Indexação" foram concatenadas, para remover quebras de linha errôneas, inseridas pelo gerador de HTML usado no site do STF. Em seguida, foram usadas expressões regulares complexas para remover trechos que não eram rótulos, e para tentar corrigir inconsistências na digitação dos termos (falta ou excesso de pontuação, diferentes padrões de separação, etc.). Em seguida, os rótulos foram recortados da *string* principal e adicionados a um vetor.

A única informação que não foi obtida através deste método foi a lista de acórdãos citados. Observou-se que no conjunto de todos os documentos, não havia um padrão de organização comum, ou um algoritmo que pudesse ser aplicado sistematicamente para encontrar todos os identificadores. Por exemplo, os identificadores não seguem um padrão de nomenclatura, então não há uma expressão regular razoável que pode ser buscada no texto. Adicionalmente, a lista encontra-se em seções diferentes para acórdãos diferentes. No meio da lista há informações adicionais, irrelevantes para a análise, difíceis de eliminar. Há muitos erros de digitação, gerando identificadores que não pertencem a nenhum acórdão. Outro empecilho importante é o fato que os acórdãos pertencem a uma faixa temporal, e os textos mais antigos fazem referência a acórdãos que estão fora desta faixa. Por este motivo, a identificação dos acórdãos citados foi feita posteriormente, usando uma metodologia diferente.

3.2 Identificação de citações

Claramente, encontrar todos (e somente) os identificadores de acórdãos nos seus textos é uma tarefa difícil. Foi então concebida uma maneira diferente de obter os mesmos resultados, que além de gerar resultados satisfatórios e completamente confiáveis, é mais simples de entender e adaptar.

Após a análise descrita na seção anterior, os dados foram persistidos localmente. O processo usado e os resultados obtidos serão detalhados nas próximas seções, mas após este processo foi possível obter, para cada acórdão, uma lista de todas as propriedades extraídas com sucesso. Sendo uma delas o identificador de cada acórdão, foi possível gerar rapidamente uma lista de todos os cinquenta mil

identificadores. O processo de busca dos acórdãos citados torna-se, então, surpreendentemente simples.

Analisa-se o texto de cada acórdão individualmente, e realiza-se uma simples busca de padrão textual. Esta busca resume-se em dividir o texto em palavras (*strings* delimitadas por espaços) e verificar se uma das palavras é igual ao padrão buscado. Busca-se, então, o primeiro identificador da lista no texto do primeiro acórdão. Naturalmente este padrão será encontrado, pois o primeiro identificador da lista é justamente o identificador do primeiro acórdão, mas este caso é facilmente detectável e é então descartado. Em seguida, busca-se o segundo identificador no texto do primeiro acórdão. Caso ele seja encontrado, tem-se a certeza de que o primeiro acórdão cita o segundo. Prossegue-se desta maneira para todos os identificadores da lista, e obtém-se assim uma lista de todos os acórdãos citados pelo primeiro. É importante notar que trata-se apenas das citações "internas", ou seja, citações a acórdãos que também estão no conjunto. As citações "externas" (a acórdãos mais antigos) não são detectadas.

Realiza-se então esta mesma sequência de operações para todos os acórdãos da lista para obter, de maneira confiável, todas as citações internas feitas por todos os acórdãos. Contudo, este método é pouco eficiente. A busca de padrão no texto tem complexidade relativamente baixa ($O(m+n)$ no caso do algoritmo KMP[7], onde m é o tamanho do texto e n é o tamanho do padrão). Contudo, ela deve ser realizada K vezes para cada um dos K acórdãos, tornando a complexidade do programa $O(K^2(m+n))$. Para valores de K tão grandes quanto 50000, esta complexidade torna o algoritmo impraticável.

Como modo de diminuir a complexidade do algoritmo, foi usada uma estrutura de dados capaz de realizar todas as K buscas simultaneamente. Isto foi possível através do armazenamento em memória de todos os identificadores, de forma compacta. Esta solução pode não ser prática para todas as aplicações, mas foi escolhida para este projeto por se adequar bem às condições de execução: dispunha-se de muita memória e armazenamento local, e era desejável diminuir a complexidade do algoritmo (e portanto o tempo de execução) através de um aumento de uso de memória.

A estrutura utilizada foi uma árvore de prefixos, inicialmente contendo somente uma raiz e uma única folha, que serve unicamente para identificar um nó terminal de maneira mais simples. Ao ler um novo identificador da lista gerada anteriormente, coloca-se todos os seus caracteres na árvore, de modo que o $(n+1)$-ésimo caractere seja filho do n-ésimo. Caso não haja nenhum outro identificador na árvore

cujo prefixo seja também prefixo do identificador a ser adicionado, adiciona-se este à raiz. Caso contrário, descarta-se o seu prefixo e adiciona-se os caracteres restantes como sub-árvore do último nó do prefixo já existente. Em ambos os casos, une-se o último nó colocado com a folha da árvore, indicando o fim do identificador.

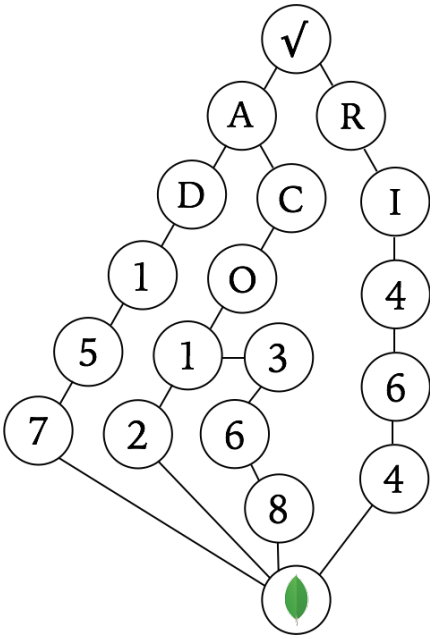


Imagem: árvore de prefixos com 4 identificadores

Após a construção desta árvore, basta usá-la como se fosse um único padrão no texto de todos os *K* acórdãos. Ao encontrar uma nova palavra, verifica-se se o seu primeiro caractere é igual ao de um dos filhos da raiz da árvore. Em caso afirmativo, compara-se a segunda letra com um dos filhos deste nó, e assim por diante até que a palavra tenha terminado e o último nó analisado tenha a folha da árvore como filho. Neste caso, a palavra é o identificador de um outro acórdão da lista. Caso qualquer uma destas condições seja falsa, passa-se para a próxima palavra do texto.

A complexidade deste algoritmo naturalmente é maior do que a do KMP[7], mas os resultados obtidos foram muito satisfatórios. Isto deve-se primariamente ao baixo grau de cada nó da árvore de prefixos (normalmente os identificadores começam com um prefixo comum, como "ACO" ou "AI").

Deste modo, foi possível buscar todas citações internas entre os acórdãos e realizar as análises propostas. A seguir, detalhar-se-á o método de persistência dos dados.

4. Persistência

4.1 Paradigma

A estrutura dos acórdãos, considerando os dados que deles foram extraídos, é bem simples. Trata-se de documentos com alguns atributos qualitativos ou quantitativos, e ponteiros para outros documentos do mesmo tipo. Por isso, o tradicional paradigma de persistência conhecido como Entidade-Relacionamento não é adequado e introduz complexidades desnecessárias, tornando-se um *overkill*. Por este motivo, utilizou-se um paradigma diferente: orientação a documentos.

O principal sistema gerenciador de bancos de dados ("SGBD") orientado a documentos é o MongoDB[8], uma implementação aberta que pertence ao conjunto dos SGBDs "noSQL", ou não-relacionais. Ele foi utilizado para armazenar as informações dos acórdãos, pois estes são um perfeito exemplo do tipo de dado que deve ser armazenado em um banco de dados deste tipo: um documento. Além disso, os bancos de dados não-relacionais não possuem esquemas, lidando mais facilmente com documentos que não são padronizados, como é o caso dos acórdãos do STF.

Cada documento no MongoDB é exibido na forma de dicionário, com pares "chave-valor", e sintaxe JSON. Internamente, o banco de dados é armazenado em formato BSON[9], ou *Binary JSON*.

4.2 Modelagem

Os documentos estão contidos em coleções, análogas às tabelas de um SGBD relacional. Para este projeto, foram criadas duas coleções principais.

A primeira, chamada `acordaos`, contém documentos que descrevem as características de um acórdão. Além das chaves mostradas a seguir, há algumas outras de uso interno. Esta coleção serviu para gerar a lista de identificadores usada na busca de padrões descrita anteriormente, e portanto não contém uma lista de acórdãos citados. Ela também foi utilizada para obter estatísticas que não dependem das citações.

```
{
    "id" : "Rcl 9688 AgR",
    "uf" : "SP",
    "relator" : "CÁRMEN LÚCIA",
    "date" : ISODate("2012-12-19"),
    "file" : "P0001J02",
    "tags" : [
        "VOTO",
        "NECESSIDADE",
        "INCLUSÃO",
        "PROCESSO",
        "PAUTA DEJULGAMENTO",
    ],
}
```

Imagem: exemplo de documento da coleção `acordaos`

A segunda coleção chama-se `links`, e contém justamente as citações entre processos. Os seus documentos têm as chaves necessárias às análises que dependem das citações.

```
{
    id : "RE 199142",
    data : ISODate("2012-07-19"),
    file : "P0025J04",
    quotes : [
        {
            id : "ACO 1234",
            uf : "RS",
            relator : "Dias Toffoli"
        },
        {
            id : "RE 9182",
            uf : "RJ",
            relator : "Ricardo Lewandowski"
        }
    ]
}
```

Imagem: exemplo de documento da coleção `links`

Como pode ser observado, dentro do documento há um vetor ou *array* de documentos citados, e cada elemento deste array é um documento dito e próprio. Não há necessidade de criar relacionamentos (e portanto tabelas adicionais), pois o banco não tem esquemas e cada documento pode ter um formato diferente. Além disso, espaço em disco (ou em memória) não é um fator limitante, então não há problemas em armazenar dados redundantes de modo a tornar as *queries* mais eficientes.

4.3 Queries

Um banco de dados do MongoDB pode ser acessado através de um interpretador, ou através de *drivers*[10] disponíveis para a maioria das linguagens de programação. Foi utilizado o *driver* para Java para popular e manipular o banco após o *parsing*, e também para executar algoritmos mais complicados, como a busca de padrões. Nos outros casos, as *queries* foram escritas em um arquivo dedicado e executadas no interpretador. A seguir, são mostrados alguns exemplos de *queries* na sintaxe do interpretador. Para os *drivers*, as *queries* funcionam exatamente do mesmo jeito, com uma sintaxe similar, adequada à linguagem correspondente.

Em primeiro lugar, deve-se abrir um servidor rodando o comando

```
mongod
```

Em seguida, pode-se abrir qualquer número de clientes, rodando o comando

```
mongo
```

para cada um. Cada cliente é um *prompt* onde pode-se gerenciar o banco e executar *queries*. Por exemplo, para conectar-se a um banco de dados específico e mostrar todas as suas coleções, basta executar:

```
use stf
show collections
```

Para encontrar um documento na coleção acordaos, a sintaxe é:

```
db.acordaos.find({ [chave : valor, chave : valor, ...] })
```

Por exemplo, o comando

```
db.acordaos.find()
```

encontra todos os acórdãos. O comando a seguir, em vez, encontrará todos os acórdãos de São Paulo redigidos por Joaquim Barbosa:

```
db.acordaos.find({relator: "Joaquim Barbosa", uf: "SP"})
```

Em seguida, pode-se executar qualquer tipo de função javascript sobre os documentos encontrados, do seguinte modo:

```
db.collection.find().forEach( function(documento) {
      var x = documento.chave + documento.outraChave;
      print(x);
});
```

A primeira busca feita no banco foi por duplicatas: acórdãos com o mesmo identificador. Foram encontrados somente 8, e supõe-se que tenham sido erros de digitação. O algoritmo utilizado consiste em encontrar todos os acórdãos e ordená-los por identificador. Para cada um, aplica-se uma função que analisa o atual e o anterior, e verifica se os identificadores são iguais. Caso sejam, os 2 documentos são removidos da coleção e adicionados a uma nova coleção chamada `duplicates`, para inspeção manual.

Em seguida, foi elaborado um arquivo com *queries* de pesquisa, que imprimem na tela uma série de estatísticas interessantes sobre os acórdãos. Para realizar estar *queries*, foram utilizados alguns operadores especiais do MongoDB[11], sobre os quais vale a pena discorrer rapidamente.

Em primeiro lugar há o operador `count`, análogo ao `COUNT` do SQL, que age sobre um conjunto de acórdãos e conta-os. Por exemplo,

```
db.collection.find().count()
```

Em seguida, pode-se mencionar o comando `sort`, análogo ao `ORDER BY` do SQL, que age sobre um conjunto de acórdãos e ordena-os pelo valor da chave passada como argumento, em ordem crescente ou decrescente. Por exemplo, pode-se usar

```
db.collection.find().sort({ myKey : 1 })
```

para ordem crescente e

```
db.collection.find().sort({ myKey : -1 })
```

para ordem decrescente. Finalmente, e talvez mais importantemente, há o comando de agregação, `aggregate`, para o qual é passado um *array* de funções que agem sobre os dados uma depois da outra, fornecendo o resultado desejado no final. Um exemplo de função muito utilizada é `$group`, análoga ao `GROUP BY` do SQL, que agrupa os dados pela chave passada como parâmetro. Portanto, para obter uma lista dos acórdãos por estado, pode-se executar:

```
db.acordaos.aggregate( [
      { $group : { _id : “$uf", count : { $sum : 1} } },
] )
```

Outra importante função utilizada dentro de uma agregação é `unwind`: esta função age sobre um documento que possui um *array*, e para cada elemento deste cria um novo documento que, no lugar do *array*, possui somente aquele elemento. Por exemplo, um documento

```
{ tags: [ "a", "b" ] }
```

seria transformado em dois documentos:

```
{ tags : "a" }
```

e

```
{ tags : "b" }
```

Esta função foi utilizada, justamente, para contar quais eram as tags mais frequentes na coleção `acordaos`.

Capítulo 3: Análise dos Dados

1. Page Rank

1.1 Origem

O algoritmo Page Rank[12] foi desenvolvido por Larry Page e Sergey Brin enquanto estudantes de Stanford, em 1998. Ele forneceu ao Google a sua funcionalidade principal e foi responsável pela superioridade do mecanismo de busca em relação aos outros existentes na época.

Em poucas palavras, é um método de mensurar a importância ou relevância de páginas da *web*, de modo a poder ordená-las após uma busca.[13][14] Na época, a quantidade de páginas existentes (cerca de 26 milhões) já tornava impraticável um cálculo da importância de todas elas por meio de um sistema de equações determinístico. Por isso, foi desenvolvido um método iterativo de atualização do *rank* de cada página, permitindo obter os resultados corretos em apenas algumas horas, considerando a capacidade computacional existente.

1.2 Definição

Como mencionado anteriormente, o objetivo do Page Rank é mensurar a importância de páginas da *web*. Pode-se, porém, generalizar o conceito de página e considerar nós de um grafo dirigido, e pensar nas suas arestas como os *hyperlinks* que ligam uma página à outra. Deste modo, pode-se aplicar o algoritmo aos acórdãos do STF exatamente da mesma maneira e com o mesmo intuito: obter um critério de ordenação para poder reconhecer os mais importantes.

Como poder-se-á ver a seguir, o conceito de importância depende da qualidade e da quantidade de links que chegam em um nó. No caso dos acórdãos, pode-se dizer que um valor de *page rank* alto indica que ele tem muitas citações, ou que é citado por acórdãos cujo *page rank* é alto. Fica então claro que o cálculo do *page rank* de um nó depende do *rank* dos outros, motivo pelo qual ele deve ser feito iterativamente. Para melhor entender este conceito, a seguir é exibido o pseudo-código do algoritmo, onde PR(X) é o valor do *page rank* atual do nó X, e nPR(X) é o novo valor calculado.

```
1    N = número de nós da rede
2    para cada nó X da rede:
3        PR(X) = 1/N
4
5    enquanto a condição de parada não for verdadeira:
6    |   para cada nó X da rede:
7    |   |   soma = 0
8    |   |   para cada nó Y tal que existe aresta X -> Y:
9    |   |       termo = PR(Y) / número de arestas saindo de Y
10   |   |       soma = soma + termo
11   |   |   nPR(X) = soma
12   |
13   |   P = [ PR(A),  PR(B),  PR(C), ... ]
14   |   Q = [nPR(A), nPR(B), nPR(C), ... ]
15   |   se distanciaEuclidiana(P, Q) < epsilon
16   |       condição de parada = verdadeira
17   |
18   |   para cada nó X da rede:
19   |       PR(X) = nPR(X)
```

Como pode-se observar, o algoritmo antes calcula um valor temporário de *page rank* para cada nó, e só depois de ter calculado todos, atribui o novo valor. Isto é feito até que se atinja uma condição de parada: normalmente, quando um certo tempo se passou, ou quando os novos valores são muito similares aos anteriores. Neste projeto, a segunda condição foi utilizada. Após calcular os novos valores de *page rank* para todos os nós, compara-se o vetor formado por todos os valores antigos com o vetor formado por todos os novos valores. Caso a distância euclidiana entre os dois seja pequena o suficiente, o algoritmo atinge a condição de parada.

Caso os nós representem páginas da web, os valores de *page rank* são uma distribuição de probabilidade, que informa qual é a chance de um navegador chegar até uma certa página clicando em *hyperlinks* aleatoriamente. Contudo, por definição, este valor é 0 para páginas do meio da rede e 1 para páginas da borda (páginas sem links para outras páginas). Por este motivo, introduziu-se o conceito de *damping factor*, indicado pela letra *d*: a probabilidade de continuar navegando. Deste modo, um navegador que age aleatoriamente encerrará a navegação

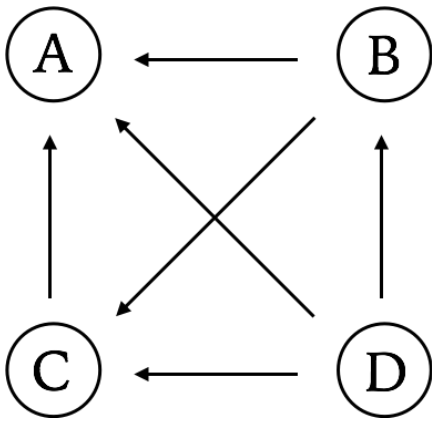
com probabilidade *d*. A expressão para o cálculo do *page rank* de um nó torna-se então:

```
nPR(X) = (1-d)/N + d * soma
```

A seguir será feito um exemplo de execução do Page Rank em uma pequena rede.

1.3 Exemplo

Suponha que exista um grafo dirigido com $N = 4$ nós: A, B, C e D. As arestas ligam os nós do seguinte modo:



Inicialmente, cada nó recebe o valor de 1/N = 0.25. Vamos exibir o cálculo do *page rank* do nó A, feito através da expressão

$$PR(A) = \frac{1\text{-}d}{N} + d\left(\sum_{x \in R(A)} \frac{PR(x)}{L(x)}\right)$$

onde R(A) é o conjunto de nós dos quais saem arestas que terminam em A, e L(x) é o número de arestas que saem de cada um deles. Normalmente, usa-se o valor 0.85 para a constante *d*.

O nó B possui 2 arestas de saída, portanto tem-se que L(B) = 2 e PR(B)/L(B) = 0.125.

Da mesma maneira, PR(C)/L(C) = 0.25 e PR(D)/L(D) = 0.083. Portanto,

$$PR(A) = (1 - 0.85)/4 + 0.85 * (0.125 + 0.25 + 0.083) = 0.4268$$

Da mesma maneira, calcula-se PR(B), PR(C) e PR(D), sempre usando o valor antigo de *page rank* dos nós (0.25) para o cálculo dos novos. Ao término da iteração, atualiza-se permanentemente o valor do *page rank*, e verifica-se se a mudança dos valores foi suficientemente grande. Caso não tenha sido, o algoritmo para.

1.4 Implementação

O algoritmo foi implementado em Java, de modo que a manipulação do banco de dados fosse idêntica ao que já tinha sido feito anteriormente. Antes de calcular o *page rank*, cada acórdão foi mapeado para uma classe homônima, de modo que se pudesse armazenar em memória uma informação importante e que seria acessada muito frequentemente: a lista de acórdãos que citam o acórdão atual. Acessar esta informação a partir do banco através de uma *query* seria muito ineficiente.

Após o cálculo, os valores de *page rank* de cada acórdão (juntamente com algumas informações de identificação) foram persistidos em uma coleção `pageranks`, para fácil inspeção.

1.5 Significado

Como descrito anteriormente, o *page rank* de um nó tem dois significados principais. O primeiro é a probabilidade deste nó ser visitado por um navegador aleatório, que muda de nó para nó através das arestas, e para de navegar com probabilidade *d*. Pode-se, portanto, imaginar o navegador como um leitor, que analisa o texto de um acórdão e decide ler mais sobre o assunto escolhendo um dos acórdãos citados.

O segundo significado é a importância do acórdão, que é diretamente proporcional ao *page rank* dos acórdãos que o citam, e inversamente proporcional à quantidade de citações feitas por estes acórdãos. Em outras palavras, se um acórdão é citado por muitos

outros, ele é importante. Se, contudo, todas essas citações foram feitas por acórdãos que citam muitos outros, elas não são tão significativas, e o acórdão em questão não é muito importante.

Os acórdãos com maior *page rank* são:

AI 541696 AgR	(1.01287×10^{-3})
AI 372358 AgR	(7.98241×10^{-4})
ADI 3289	(7.55900×10^{-4})
ADI 1721	(7.11430×10^{-4})
Inq 2206 QO	(6.45409×10^{-4})

2. Resultados

Após ter construído todas as coleções necessárias para persistir os dados, foi possível executar *queries* para extrair dados estatísticos sobre os acórdãos. Procurou-se, principalmente, produzir resultados que caracterizassem todo o conjunto de acórdãos, ou que fornecessem comparações entre um acórdão e os demais. Em outras palavras, buscou-se evidenciar características que não seriam possíveis ao analisar cada acórdão singularmente. A seguir, elas serão apresentadas em mais detalhe.

2.1 Estatísticas básicas

Em primeiro lugar, agrupou-se os acórdãos por ano, usando o *framework* de agregação mencionado anteriormente. Como pode-se observar a seguir, o número de acórdãos publicados tende a crescer com o passar dos anos, muito provavelmente influenciado pelo aumento da população, além do aumento da eficiência do STF ao julgar os seus processos.

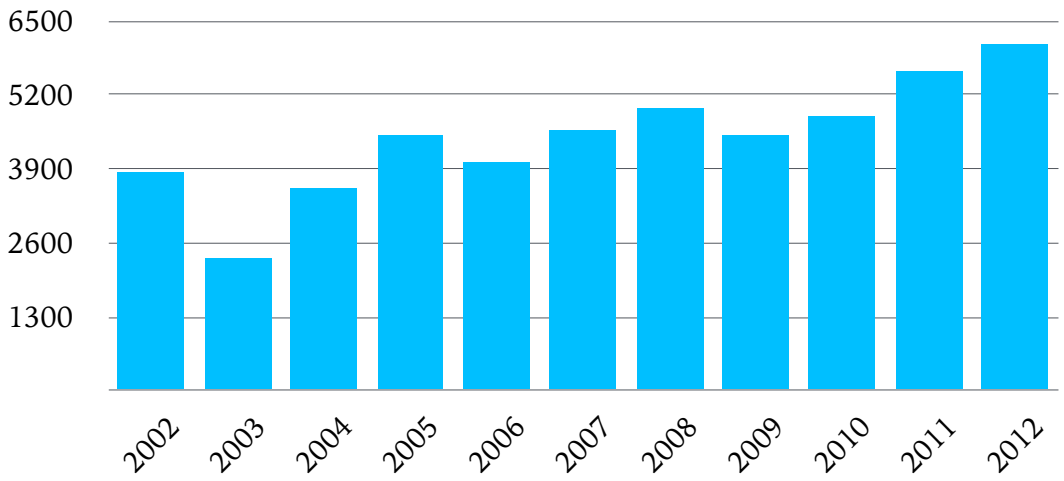


Gráfico: número de acórdãos por ano

A seguir apresenta-se o número de acórdãos por estado, calculado similarmente:

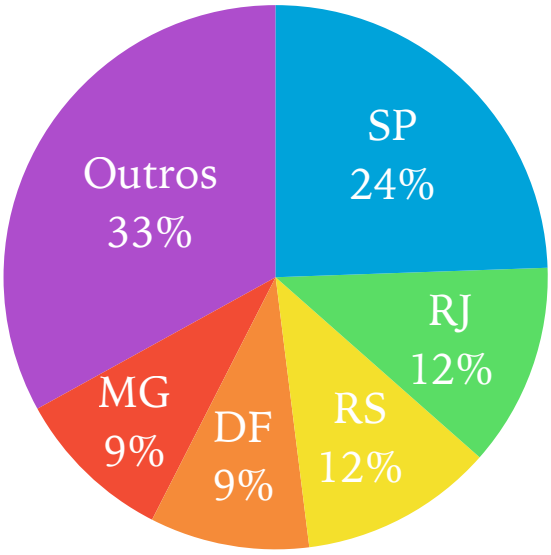


Gráfico: número de acórdãos por estado

Nota-se claramente que os estados mais populosos, com exceção do Distrito Federal, são os que geram processos que dão origem a mais acórdãos. A correlação destes números com os índices de população destes estados é quase linear:

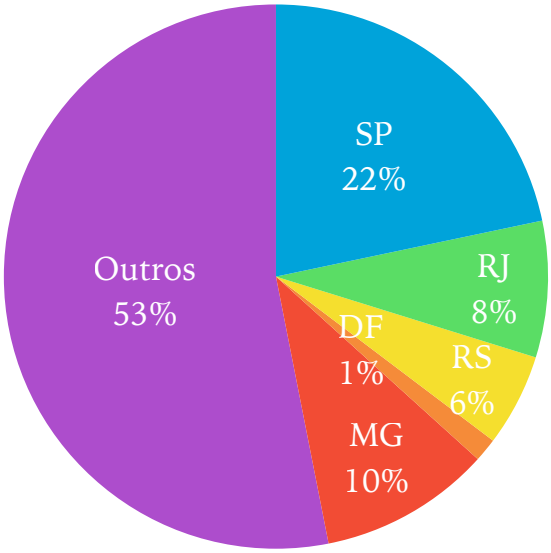


Gráfico: população por estado

Em seguida, calculou-se o número de acórdãos por relator, de maneira análoga aos cálculos feitos anteriormente:

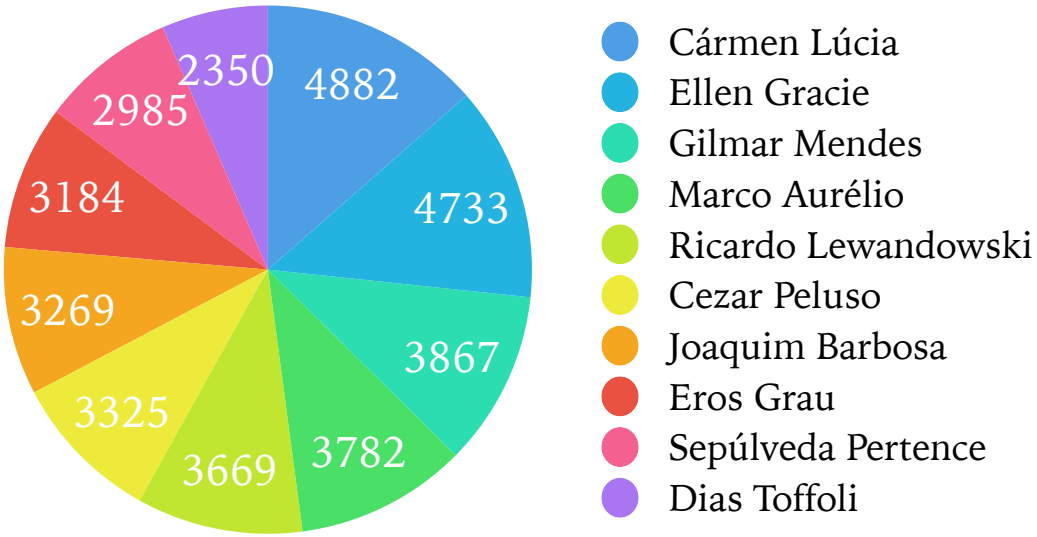


Gráfico: acórdãos por relator

A distribuição é quase uniforme, como era de se esperar, pois a disponibilidade dos ministros do Supremo é similar, e o tempo dedicado à redação de um acórdão tem variabilidade muito baixa.

2.2 Citações

Seja *A* o conjunto de todos os acórdãos da coleção. Destes, alguns não fazem nenhuma citação. Esta informação pode ser verificada pela ausência dos termos "acórdãos citados" e "acórdãos no mesmo sentido" (com as devidas variabilidades léxicas). Os acórdãos que não pertencem a este conjunto, consequentemente, citam pelo menos um outro acórdão. Seja AC este conjunto. Dentro dele, há ainda os acórdãos cujas citações incluem uma citação interna, ou seja, uma citação a um acórdão existente na coleção. Seja AI o conjunto destes acórdãos.

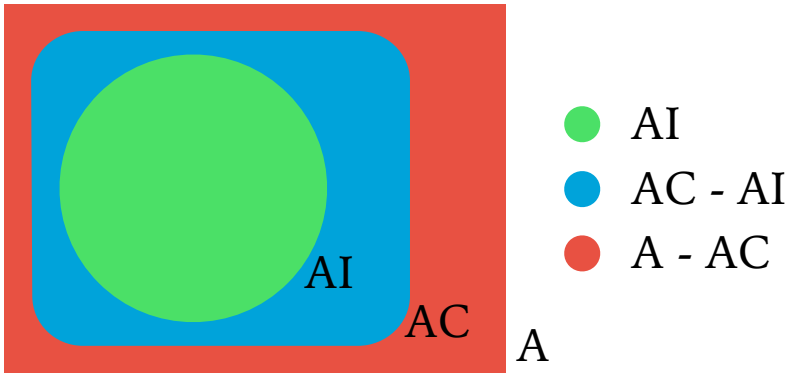


Diagrama: conjuntos de acórdãos com base em citação

Estas quantidades se relacionam deste modo:

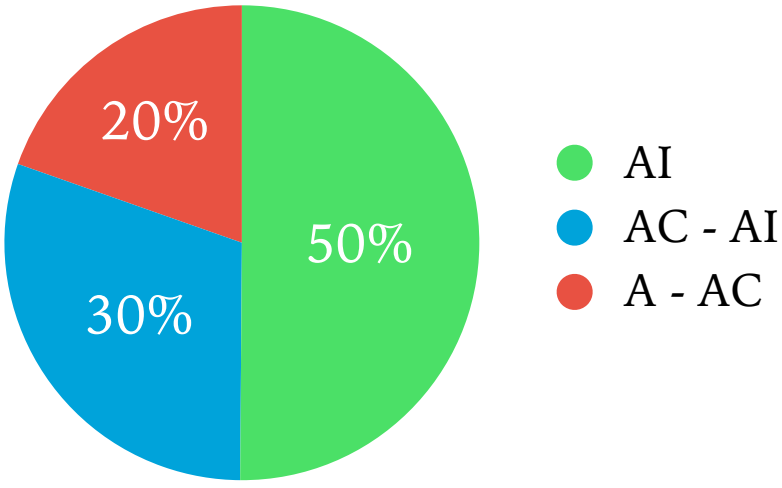


Gráfico: proporções entre acórdãos com base em citação

Como pode ser observado, 80% dos acórdãos fazem citações, revelando uma rede mais densa do que se imaginava. Além disso, metade dos acórdãos fazem citações internas, que, de fato, constituem as arestas do grafo, tornando possível realizar cálculos com grande confiabilidade.

De posse destes dados, é simples calcular a média de citações por acórdão, para que se tenha uma ideia do grau de saída de cada nó da rede. Considerando-se o número total de citações, M, e as três quantidades de acórdãos A, AC e AI mencionadas acima, calcula-se:

Média de citações por acórdão: M / A = **1.42**
Média de citações por acórdão que cita: M / AC = **1.77**
Média de citações por acórdão que cita internamente: M / AI = **2.84**

Apesar da grande proporção de acórdãos que fazem citações, as médias demonstraram-se baixas. Contudo, há acórdãos que realizam muitas citações. O processo que mais cita, com 176 citações internas. é o AP 470[15], também conhecido como "mensalão". Trata-se de um documento enorme, com um nível de detalhe muito maior do que a maioria. Isto provavelmente deve-se à grande visibilidade que o processo teve na mídia.

Os acórdãos mais citados, com 292, 270 e 266 citações respectivamente, são: AI 330970, AgR AI 372358 AgR e AI 481531 AgR. Não se nota uma característica especial no tema destes acórdãos, e supõe-se que o motivo pelo qual eles são amplamente citados resuma-se ao tema sobre o qual tratam: possivelmente é um tema de base, comum a muitos outros processos.

2.3. Rótulos

Como última medida interessante sobre os acórdãos, procurou-se elencar quais eram os rótulos mais frequentes. Isto foi feito usando uma operação conhecida como *map-reduce*, uma sequência de duas operações sequenciais que extraem todos os rótulos de um acórdão, e geram documentos temporários que contam as suas ocorrências. Estes documentos são persistidos na coleção `tags`, permitindo uma inspeção de todos os rótulos presentes. A coleção foi ordenada por frequência, e os dez primeiros termos são exibidos a seguir:

ausência
descabimento
recurso extraordinário
voto vencido
impossibilidade
decorrência
existência
necessidade
ocorrência
inexistência

Para atribuir significado a estes termos, é necessário conhecimento sobre o que eles representam dentro do contexto no qual estão inseridos. Contudo, pode-se ter uma ideia básica sobre as situações que foram descritas nos textos destes processos.

Capítulo 4: Lições Aprendidas

1. Desafios

O primeiro desafio encontrado, perceptível somente no final do ano, é relacionado à dimensão do trabalho. A realização do TCC é comparável a uma maratona: deve-se ter um bom planejamento, grande resistência e é essencial manter sempre o mesmo ritmo. Durante a execução das tarefas, percebi que o ritmo foi muito variável. No começo, era estimulante aprender sobre coisas novas e planejar tudo o que seria feito de forma megalomaníaca. Com o decorrer do ano, porém, foi necessário dedicar atenção a muitas outras tarefas e o ritmo caiu – assim como o escopo do projeto – voltando a crescer nos últimos meses, nutrido pela linha de chegada que se aproximava.

Também constituiu um grande desafio a dependência de tecnologias que eu não conhecia e que ninguém mais parecia conhecer. Foi comum encontrar algoritmos, bancos de dados, *drivers* e *frameworks* dos quais eu nunca tinha ouvido falar. Ficou claro que o problema da atualidade é o excesso de informação, e não mais a falta. Este trabalho inteiro é prova disso.

Além disso, encontrei bastante dificuldade em transformar vários conhecimentos obtidos durante a graduação em um único trabalho de forma concreta. Estes conhecimentos haviam sido esquecidos, não praticados há muito tempo, ou mesmo adquiridos sem querer durante o tempo livre. Foi necessário reler muitos textos e códigos, e ponderar sobre o funcionamento conjunto de muito do que aprendi desde o primeiro ano.

2. Frustrações

As maiores frustrações sofridas durante o desenvolvimento se deram por conta da imprevisibilidade de alguns resultados. O maior exemplo ocorreu no primeiro semestre, enquanto eu procurava estudar a ferramenta Scrapy[4] para poder fazer *web crawling* no site do STF e obter assim os dados de maneira organizada. Passei semanas lendo a documentação[5] e escrevendo alguns scripts menores para me certificar de que o sistema funcionava. Depois disso, escrevi o código da *spider* que deveria iterar pela lista de acórdãos, e este apresentou um problema após algumas centenas de arquivos. Em pelo menos um

deles, a ordem das seções não era a mesma. A ferramenta não estava preparada para lidar com isso, e a técnica teve que ser abandonada.

3. Disciplinas relevantes

Algumas disciplinas foram especialmente relevantes para o desenvolvimento do trabalho, por terem apresentado um conteúdo ou ferramenta útil ou por terem introduzido um paradigma diferente, sem o qual o trabalho não poderia ter sido bem sucedido. Contudo, antes de listá-las, é necessário dizer que todas as disciplinas, sem exceção, contribuíram para a minha formação acadêmica e foram essenciais para o meu amadurecimento intelectual. Foi necessário estudar assuntos difíceis, às vezes considerados irrelevantes, e lidar com a frustração de não obter o aproveitamento desejado em todos eles. Isto, de certa forma, antecipou alguns dos problemas que vieram a ocorrer durante a elaboração deste trabalho.

A seguir, então, listo as disciplinas cujo conteúdo me ajudou a criar e desenvolver o TCC, e inspirou as decisões tomadas e soluções encontradas.

MAE0121 - Introdução à Probabilidade e à Estatística I e MAE0212 - Introdução à Probabilidade e à Estatística II

Estas disciplinas introduziram os conceitos fundamentais usados na análise dos acórdãos, permitindo-me entender como condensar listas enormes de números em significados concretos. Através dos conhecimentos adquiridos, pude exibir os resultados com confiança e incluindo dados adicionais que fornecessem escala e significado.

MAC0328 - Algoritmos em Grafos

Esta disciplina foi fundamental para que eu pudesse não só compreender o conceito de grafo, como ser capaz de modelar um conjunto de dados como tal. Para alcançar os resultados obtidos, foram feitas buscas e aplicados algoritmos iterativos cuja compreensão não teria sido possível sem os conhecimentos ensinados nesta disciplina.

MAC0425 - Inteligência Artificial

Através desta disciplina, pude aprimorar os meus conhecimentos sobre buscas em árvores (um caso particular de grafos dirigidos) e também tive contato com reconhecimento de texto, que foi usado no

começo do desenvolvimento para procurar obter informações dos acórdãos a partir do seu texto bruto.

MAC0439 - Laboratório de Bancos de Dados

Nesta disciplina aprendi a modelar devidamente um banco de dados relacional e a realizar *queries* complexas para consultar os dados nele contidos. A partir destes conhecimentos, fui capaz de modelar o banco de dados utilizado para armazenar os dados dos acórdãos e a consultá-lo de maneira eficiente para extrair as informações desejadas. O paradigma de orientação a documentos não teria sido escolhido se eu não tivesse um conhecimento prévio sobre bancos de dados relacionais.

4. Próximos passos

Um dos intuitos deste trabalho, como citado algumas vezes no decorrer deste texto, é fornecer um conjunto de dados a partir das quais podem ser geradas novas informações. Desde o começo, o objetivo foi produzir um trabalho sobre o qual pudessem ser aplicados, justamente, próximos passos.

Por exemplo, pode-se futuramente atualizar a lista de acórdãos incluindo textos mais recentes, e assim verificar se as estatísticas obtidas sobre os dados atuais são diferentes daquelas obtidas sobre os dados futuros, e por que motivos. Outra modificação importante refere-se ao tipo de informações obtidas a partir dos dados: pode-se mudar os parâmetros das buscas, ou aplicar novos algoritmos para obter novas informações sobre o conjunto de processos.

Pode-se também adicionar novas fontes de informação. Um exemplo é o Superior Tribunal de Justiça, que também disponibiliza online os seus processos da mesma maneira que o STF. Naturalmente, deve-se escrever outra *spider* para fazer o download dos dados, e re-implementar inteiramente o Parser para adequar-se a uma estrutura de texto diferente.

Em relação à implementação do trabalho, certamente pode-se trabalhar mais para tornar as pesquisas ainda mais eficientes. Pode-se também encontrar ou criar um *website* para divulgar tais resultados, permitir consultas customizadas e ouvir sugestões e críticas da população que os consultar.

5. Agradecimentos

Em primeiro lugar, agradeço ao orientador deste projeto, Prof. Dr. Marcelo Finger, pela orientação e sabedoria, não só em termos computacionais. Agradeço também aos professores do IME que transmitiram seus conhecimentos ao longo dos anos, especialmente os que disponibilizaram seu tempo e conhecimento para ajudar na elaboração deste e de outros projetos.

Agradeço também à gentil comunidade do StackOverflow, certo de que compartilho este sentimento com todos os leitores deste texto.

Aos colegas de curso, que desde o primeiro ano compartilharam tudo e mais um pouco, deixo um especial agradecimento e desejo de boa sorte. Dos EPs aos BIFEs, foi tudo muito muito legal.

Finalmente, agradeço muito pelo apoio e incentivo à família, aos amigos e à Paula, que sempre estiveram ao meu lado.

Espero poder retribuir à altura.

Referências

[1] **Supremo Tribunal Federal**. Acórdão: do julgamento pelo STF até a publicação no Diário da Justiça, 2013. Acessado em 27/11/2014. Disponível em http://www.stf.jus.br/portal/cms/verNoticiaDetalhe.asp?idConteudo=229547

[2] **Supremo Tribunal Federal**. Institucional, 2011. Acessado em 27/11/2014. Disponível em http://www.stf.jus.br/portal/cms/verTexto.asp?servico=sobreStfConhecaStfInstitucional

[3] **Supremo Tribunal Federal**. Pesquisa de Jurisprudência. Acessado em 27/11/2014. Disponível em http://www.stf.jus.br/portal/jurisprudencia/pesquisarJurisprudencia.asp

[4] **Scrapy**. Homepage. Acessado em 27/11/2014. Disponível em http://scrapy.org

[5] **Scrapy**. Scrapy Documentation, 2014. Acessado em 27/11/2014. Disponível em http://doc.scrapy.org

[6] **W3C**. XML Path Language, 1999. Acessado em 27/11/2014. Disponível em http://www.w3.org/TR/xpath/

[7] **KNUTH D.E., MORRIS J.H., PRATT V.R.** Fast pattern matching in strings, SIAM Journal on Computing, 1977.

[8] **MongoDB Inc**. MongoDB Homepage, 2013. Acessado em 27/11/2014. Disponível em http://www.mongodb.org

[9] **MongoDB Inc**. BSON Specification. Acessado em 27/11/2014. Disponível em http://bsonspec.org

[10] **MongoDB Inc**. MongoDB Drivers, 2011. Acessado em 27/11/2014. Disponível em http://docs.mongodb.org/ecosystem/drivers/

[11] **MongoDB Inc**. MongoDB CRUD Introduction, 2011. Acessado em 27/11/2014. Disponível em http://docs.mongodb.org/manual/core/crud-introduction/

[12] **Brin S., Page L.**, The Anatomy of a Large-Scale Hypertextual Web Search Engine, 1998. Acessado em 27/11/2014. Disponível em http://infolab.stanford.edu/~backrub/google.html

[13] **Sobek M.**, A Survey of Google's PageRank, 2003. Acessado em 27/11/2014. Disponível em http://pr.efactory.de

[14] **Jaster A.**, The PageRank algorithm. Acessado em 27/11/2014. Disponível em http://pagerank.suchmaschinen-doktor.de

[15] **Grupo Bandeirantes de Comunicação**, STF libera acórdão completo do caso mensalão, 2013. Acessado em 27/11/2014. Disponível em http://noticias.band.uol.com.br/brasil/noticia/100000592558/STF-disponibiliza-acórdão-completo-do-caso-mensalão.html