Universidade Federal do Rio de Janeiro

Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

Programa de Engenharia de Sistemas e Computação - Engenharia de Dados e Conhecimento Turma de Mestrado 2016.1, Disciplina: Busca e Recuperação da Informação

Professor: Geraldo Xexéo

1 de Agosto 2016, Rio de Janeiro - Brasil

Geração de Tag Cloud\* com Knime®: Impeachment de Dilma Rousseff

Dáve Liaoa, Leniel Macaferib

a Pesquisador acadêmico, mestrando, COPPE - Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil

b Pesquisador acadêmico, mestrando, COPPE - Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil

Abstract

This paper was developed to assess the words most used in the speeches of Brazilian Deputies during the voting process of Dilma Rousseff’s impeachment on April 17th 2016 at the Brazilian National Congress. After examining the PDF transcription file provided by the The Chamber of Deputies which contains all the speeches, we extracted all the relevant data to further process them. After that we used Knime® analytics tool to generate a Tag Cloud to better visualize the words that were most mentioned by the deputies.

This paper is part of the final job for the Information Search and Retrieval discipline.

*Keywords: Information Search Retrieval; Tag Cloud; Knime; Impeachment; Dilma Rousseff*

# 1 Introdução

O presente trabalho foi desenvolvido com o objetivo de comparar as falas dos deputados do Congresso Nacional, por ocasião da votação da admissibilidade do processo de Impeachment da presidenta Dilma Rousseff em sessão plenária do dia 17/04/2016. A comparação foi realizada através da geração de uma tag cloud, usando a ferramenta de analytics Knime®.

Para a execução deste trabalho, executamos as seguintes tarefas:

1. Acessar o site da Câmara dos Deputados do Congresso Nacional.
2. Obter o arquivo PDF que contém a transcrição de todas as falas dos deputados da sessão do dia 17/04/2016.
3. Separar as falas de cada deputado
4. Analisar visualmente as palavras através de uma tag cloud

# 2 Revisão bibliográfica

Neste trabalho utilizamos as principais técnicas empregadas na área de Busca e Recuperação da Informação (BRI). Estas técnicas serão abordadas e discutidas para explicar os passos seguidos para a geração do resultado final deste trabalho.

## 2.1 Knime

Knime® é uma empresa Alemã que constrói software para acesso rápido, fácil e intuitivo, visando a ciência de dados avançada e ajudando as organizações a impulsionar a inovação. A Plataforma de Analytics (descoberta, interpretação e comunicação de padrões significativos nos dados) Knime é a principal solução aberta para a inovação orientada por dados, projetada para descobrir o potencial escondido nos dados, na mineração de ideias novas ou para prever novos futuros. As organizações podem elevar sua colaboração, produtividade e desempenho com uma robusta gama de extensões comerciais criada para a plataforma de código aberto.

Por mais de uma década, uma próspera comunidade de cientistas de dados em mais de 60 países tem trabalhado com a plataforma Knime em todos os tipos de dados: de números a imagens, moléculas a humanos, sinais a redes complexas e estatísticas simples até análises de big data (grandes volumes de dados).

A característica Knime[1] de processamento de texto foi projetada e desenvolvida para ler e processar dados em forma de texto e transformá-los em dados numéricos (documentos e vetores de termos) de forma a aplicar “nós” de mineração regular de dados Knime, como por exemplo, para agrupamento e classificação dos dados. Esta característica permite o *parsing\** de textos disponíveis em vários formatos (XML, MS-Word ou PDF). Os documentos podem ser filtrados (por *stop words* ou filtros de entidades nomeadas), lematizados por stemmers para várias línguas e pre-processados de muitas outras formas. A frequência das palavras pode ser computada, palavras-chave podem ser extraídas e documentos podem ser visualizados (por exemplo, através de tag clouds).

O procedimento adotado no Knime utiliza o fluxo de trabalho exemplificado na Figura 1 a seguir. Os dados processados servem de entrada para outros processos. Os resultado desses processos podem servir de entrada para outros processos e assim sucessivamente até a obtenção da saída desejada.

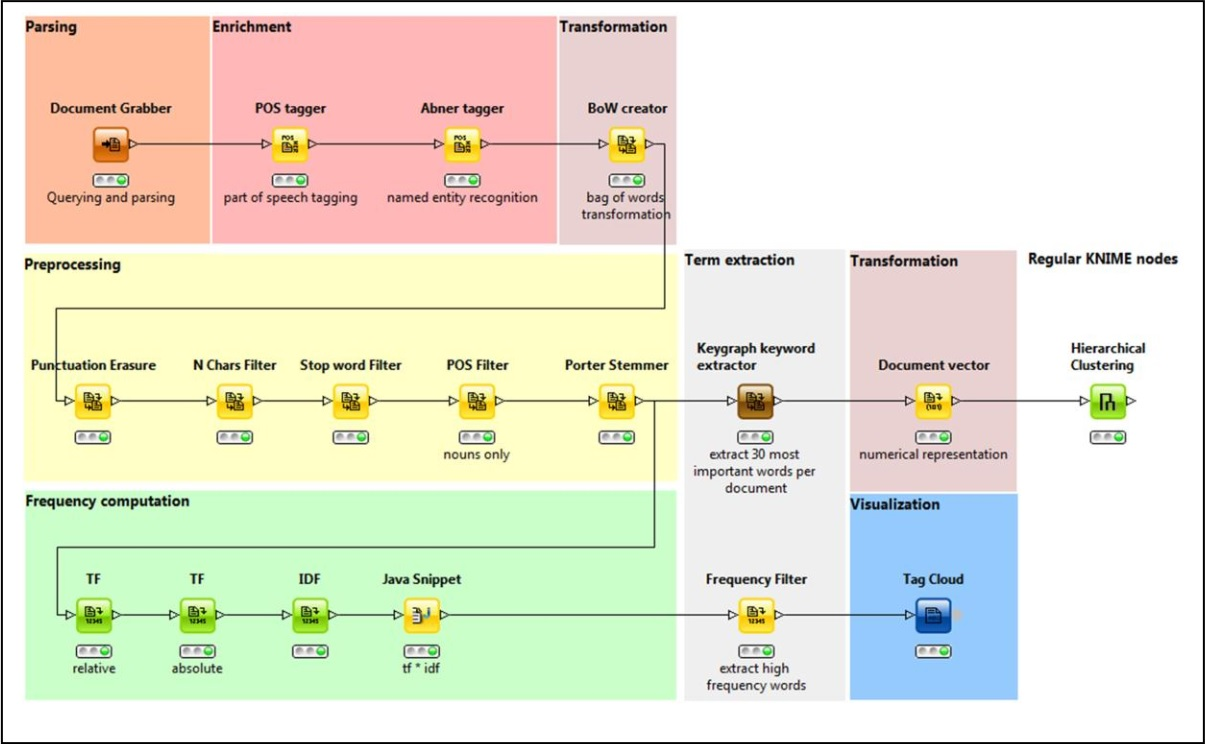


Figura 1 - Fluxo de trabalho ilustrando a ordem de processamento dos nós

# 3 Desenvolvimento

Para o desenvolvimento utilizamos a seguinte estratégia: criação de um console App para processar o arquivo PDF que contém a transcrição das falas de todos os deputados. O programa gera como resultado uma planilha Excel que contém os dados de cada voto dos parlamentares do Congresso Nacional. Optamos por não utilizar a funcionalidade de leitura de PDF do Knime.

## 3.1 Tecnologias utilizadas

Foram utilizadas as seguintes tecnologias:

* Linguagem de programação: C#
* Ambiente de desenvolvimento: Microsoft Visual Studio Community 2015 (free)
* Bibliotecas: iTextSharp (utilizada para manipulação do arquivo PDF) e NPOI utilizado para criação e manipulação do arquivo Excel.

## 3.2 Desenvolvimento do parser

Primeiramente o parser lê o arquivo PDF na função Main do programa desenvolvido:

static void Main(string[] args)

{

   var pdfText = ExtractTextFromPdf(@"Y:\Dev\InformationSearchRetrieval\Paper\EV1704161400.pdf");

   var votes = ParseVotes(pdfText);

   WriteVotesToExcel(votes);

}

A função ExtractTextFromPdf utiliza a biblioteca iTextSharp para a leitura de todo o conteúdo do arquivo PDF. Ao final da execução desta função, a variável pdfText conterá todo o texto do intervalo de páginas 121 até 323. Nestas páginas estão os dados relevantes para nosso trabalho.

/// <summary>

/// Extracts the PDF full text.

/// </summary>

/// <param name="path">Path to PDF file</param>

/// <returns></returns>

private static string ExtractTextFromPdf(string path)

{

   using(PdfReader reader = new PdfReader(path))

   {

       StringBuilder text = new StringBuilder();

       // Voting transcription starts at page 121 and ends at page 323

       for(int i = 121; i <= 323; i++)

       {

           text.Append(PdfTextExtractor.GetTextFromPage(reader, i));

       }

       return text.ToString();

   }

}

Em seguida o texto é passado para a função ParseVotes que é a responsável pela lógica que extrai os votos de cada deputado do Congresso Nacional.

A lógica faz a manipulação do texto dividindo sentenças, removendo palavras, filtrando palavras, etc. de maneira que o texto se enquadre em um formato adequado ao processamento posterior. Para melhor descrever as propriedades de cada voto, foi criada uma classe chamada Vote no seguinte formato:

class Vote

{

public string Deputy { get; set; }

public string Party { get; set; }

public string State { get; set; }

public string Speech { get; set; }

}

Onde Deputy contém o nome do deputado; Party o partido do mesmo; State o estado do parlamentar e Speech que contém a transcrição da fala do deputado no momento do voto.

A seguir é mostrado o corpo da função ParseVotes. Ao final de sua execução uma lista de Votos é retornada.

private static List<Vote> ParseVotes(string text)

{

   var votes = new List<Vote>();

   var sentences = text.Split(new[] {"O SR.", "A SRA."}, StringSplitOptions.None).AsEnumerable();

   // Removing white space...

   sentences = sentences.Select(s => s.TrimStart());

   // Filtering to keep only the Deputies' votes...

   var filtered = sentences.Where(s => !s.StartsWith("PRESIDENTE") && !s.StartsWith("BETO MANSUR") && !s.StartsWith("FELIPE BORNIER") && !s.StartsWith("ALEX CANZIANI") && !s.StartsWith("CÂMARA DOS DEPUTADOS"));

   // Removes page number followed by header text present in every page with session number, etc.

   // Single line changes how the .dot operator works: http://stackoverflow.com/a/1780037/114029

   var regex = new Regex(@"\d+(.\*)4176", RegexOptions.Singleline);

   foreach(var s in filtered)

   {

       var parts = s.Split(new[] { " - " }, StringSplitOptions.None);

       if(parts.Length > 2)

       {

           for(int i = 2; i < parts.Length; i++)

           {

               parts[1] += parts[i];

           }

       }

       var deputy = parts.ElementAt(0);

       var deputyParts = deputy.Split(new[] { '(', ')' }, StringSplitOptions.RemoveEmptyEntries);

       deputy = CultureInfo.CurrentCulture.TextInfo.ToTitleCase(deputyParts[0].ToLower()).TrimEnd();

       var party = deputyParts.Length > 1 ? deputyParts[1].TrimEnd('.') : string.Empty;

       var partyParts = party.Split('-');

       // Removing the word "Bloco/"

       party = partyParts[0].Replace("Bloco/", string.Empty);

       var state = partyParts.Length > 1 ? partyParts[1].Substring(0, 2) : string.Empty;

       var speech = parts.ElementAt(1);

       speech = regex.Replace(speech, string.Empty);

       var vote = new Vote { Deputy = deputy, Party = party, State = state, Speech = speech };

       TreatExceptionInData(vote);

       if(!votes.Any(v => v.Deputy == vote.Deputy))

       {

           votes.Add(vote);

       }

       else

       {

           var deputyVote = votes.Single(v => v.Deputy == vote.Deputy);

           if(deputyVote.State == string.Empty)

           {

               deputyVote.State = vote.State;

               deputyVote.Party = vote.Party;

           }

           // Append the speech continuation for that deputy...

           deputyVote.Speech += vote.Speech;

       }

   }

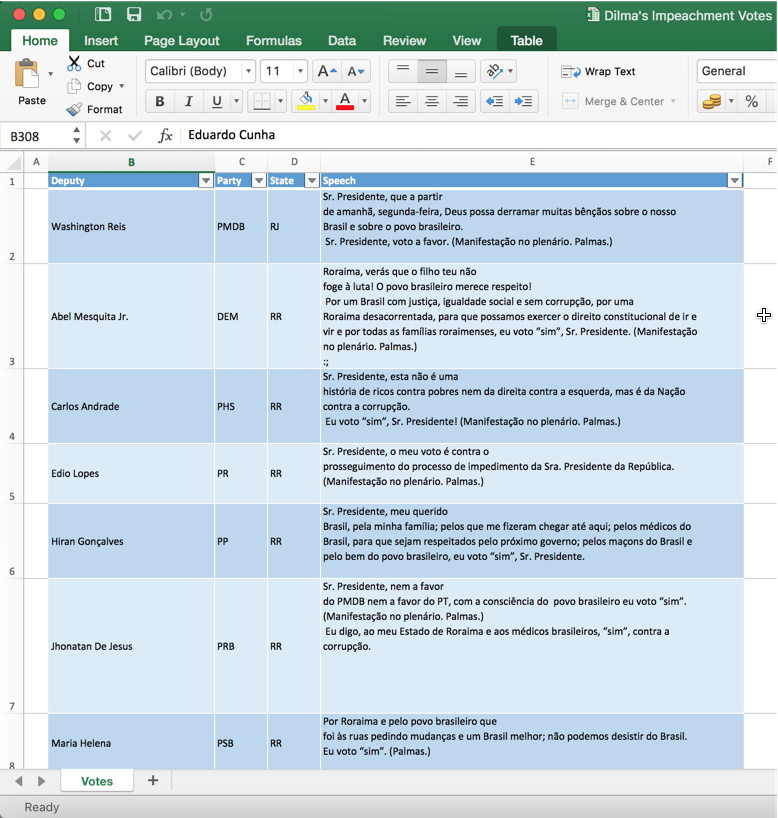
   return votes;

}

## 3.3 Geração da planilha Excel

A função WriteVotesToExcel recebe a lista de votos processada pelo parser descrito na seção anterior. Esta função faz uso da biblioteca NPOI que dentre outras funcionalidades pode criar e manipular arquivos Excel.

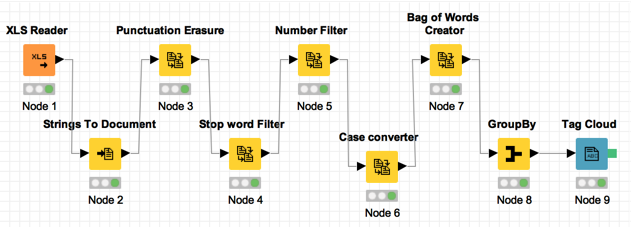
No nosso caso, usamos NPOI para gerar uma planilha Excel que contém uma folha de dados (worksheet) chamada Votes. Nesta worksheet os votos dos deputados são escritos; uma linha para cada voto como mostrado na Figura 2 a seguir:

  
Figura 2 - Planilha Excel contendo os dados de cada deputado: Nome, Partido, Estado e Fala

## 3.4 Tag cloud

Para verificarmos quais foram as palavras mais mencionadas durante as falas de todos os deputados, usamos a ferramenta de analytics Knime.

Foi criado um Workflow no Knime como mostrado na Figura 3:

  
Figura 3 - Workflow do Knime para produzir a tag cloud

O Workflow é composto por 8 nós que são descritos a seguir:

* Nó 1: lê a planilha Excel;
* Nó 2: transforma o texto das falas em documentos que podem ser interpretados pelo módulo de Text Processing (Processamento de Texto) do Knime;
* Nó 3: Remove pontuações tais como: . , ; : etc;
* Nó 4: Remove stop words usando o dicionário já existente no Knime para o idioma Português;
* Nó 5: Remove números;
* Nó 6: Converte todas as palavras para letras maiúsculas;
* Nó 7: Cria uma Bag of Words (saco de palavras, numa tradução literal);
* Nó 8: GroupBy agrupa todos os termos e realiza a contagem (soma);
* Nó 9: Gera a imagem da tag cloud.

O módulo de Text Processing do Knime deve ser instalado através do menu File => Install Knime Extensions. Em seguida procure por Knime Labs Extensions => Text Processing.

Para gerar a tag cloud, consideramos todas as 3172 palavras\termos distintos que sobraram após a execução de todos os nó até chegar no Nó 9 onde é gerada a tag cloud. A tag cloud é mostrada a seguir na Figura 4, com zoom na parte central da mesma:

  
Figura 4 - Tag cloud gerada com o Knime mostrando as palavras que mais foram ditas durante a votação do impeachment de Dilma Rousseff no Congresso Nacional

# 4 Resultado

Uma tag cloud é útil em situações em que um breve contato visual seja suficiente para uma análise ou tomada de decisão.

Ao analisarmos a tag cloud gerada, podemos perceber que dentre as palavras mais ditas estão: presidente, Brasil, voto, impeachment, Dilma, estado, sim, golpe, constituição, corrupção, país, Cunha, esperança, família, manifestação, brasileiro, democracia, vida, respeito.

A tag cloud nos permite uma rápida avaliação das palavras mais usadas na transcrição das falas dos deputados. Ela também mostra o status quo da sociedade brasileira nesse momento decisivo para a nação brasileira; status esse que em grande parte orientou a votação da maioria dos deputados.

# 5 Conclusão e Lições aprendidas

O resultado do presente trabalho nos permitiu experimentar na prática, a utilização dos conceitos de Busca e Recuperação da Informação aprendidos em sala de aula. A partir do conhecimento da ferramenta Knime um novo universo de possibilidades nos foi mostrado.

O código fonte gerado pode ser obtido no seguinte repositório do GitHub:

<https://github.com/leniel/InformationSearchRetrieval/tree/master/Paper>

**Referências**

1. Knime. Technical Report The KNIME Text Processing Feature: An Introduction. <https://www.knime.org/white-papers>,

https://www.knime.org/files/knime\_text\_processing\_introduction\_technical\_report\_120515.pdf, obtido em 21/07/2016.