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 de Busca e Recuperação de Informação, Professor Geraldo Xexéo, 23 Julho 2016, Rio de Janeiro, Brasil

Geração de Tag cloud[1]: Impeachment de Dilma Rousseff

Dáve Liao[a], Leniel Macaferi[b]

[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

Resumo

O presente trabalho foi desenvolvido com o objetivo de avaliar a similaridade das 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, através da geração de um Tag Cloud, usando a ferramenta Knime® para esta finalidade. O Método utilizado para esta análise foi escolhido por abordar todos os principais tópicos do processo de recuperação de informações. Como resultado identificamos as palavras "Sim", "Democracia", "povo" e "Brasil" como as mais relevantes e similares no contexto da votação.

Palavras-chave: Similaridade. Tag Cloud. Recuperação de Informação. Impeachment.

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, buscou-se a seguinte ordem de atividades:

1. Acessar o site da Câmara dos Deputados do Congresso Nacional.
2. Obter o arquivo em 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. Análise visual através de tag cloud.

[1] Tag Cloud – Do inglês: Rótulos dispostos em forma de nuvem. Imagem composta por várias palavras de diferentes tamanhos, posições e direções. Cada palavra varia conforme a sua frequência nos documentos analisados. As palavras mais utilizadas aparecem na região central da imagem, na horizontal, com um tamanho de fonte maior, recebendo destaque.

2. Revisão bibliográfica

Neste trabalho, buscou-se utilizar as principais técnicas empregadas no campo da recuperação de informação (IR – Information Retrieval). Estas técnicas, utilizadas no programa Knime, serão abordadas e discutidas para explicar os passos seguidos para a geração do produto final deste trabalho.

Algumas expressões no universo de IR são tratadas por seus nomes em inglês, pois não encontramos uma tradução adequada que represente o sentido encerrado na expressão original. Estas palavras já possuem aceitação e entendimento universal no campo de IR. Neste artigo encontramos algumas delas e optou-se por preservar seus nomes originais.

2.1. POS Tagger (Part-of-Speech)

Segundo Manning e Schütze (1999), o objetivo final da pesquisa sobre processamento de linguagem natural (NLP – Natural Language Processing) é analisar e compreender a linguagem. [...] ainda estamos longe de alcançar este objetivo. Por esta razão, muitas pesquisas em NLP possuem tarefas intermediárias que tratam algumas estruturas inerentes da linguagem sem a necessidade da compreensão completa. Uma destas tarefas é a rotulação de classes gramaticais, ou simplesmente "tagging" que associa a respectiva classe gramatical a cada palavra lida.

2.2. NER (Named Entity Recognition)

Mansouri, Affendey e Mamat (2008) declaram que o reconhecimento de entidades nomeadas é um subproblema de extração de informação e envolve o processamento de documentos estruturados e não estruturados e identifica expressões que se referem a pessoas, lugares, organizações e companhias. O NER é uma tarefa fundamental e é o núcleo do sistema de processamento de linguagem natural (NLP). O NER envolve duas tarefas, as quais são, primeiramente a identificação de nomes próprios no texto, e depois, a classificação destes nomes num conjunto de categorias de interesse pré-definidas, tais como nomes de pessoas, organizações (companhias, organizações governamentais, comitês etc), localizações (cidades, países, rios etc), datas e expressões de tempo.

2.3. Bag of Words

O modelo de saco-de-palavras (Bag of Words) é uma representação simplificada, usada no NLP e IR. Neste modelo, um texto, podendo ser uma sentença ou um documento, é representado como um saco de suas palavras, desconsiderando a gramática e mesmo a ordem das palavras, mas mantendo a multiplicidade. O modelo de saco-de-palavras é comumente

usado em métodos de classificação de documentos, onde a frequência de cada palavra é usada como uma característica para treinar um classificador.

2.4. Stop Words

Stop words são palavras com funções sintáticas de artigo, preposição, advérbio ou outras cuja presença na lista de pesquisa é desnecessária.

Para Manning e Schütze (1999), Stop words são palavras que podem ser ignoradas na recuperação de informaçãao orientada a palavras-chave, sem um efeito significante na precisão da recuperação. Em alguns sistemas IR nem todas as palavras são representadas no índice invertido. Uma lista de stop words apresenta palavras consideradas improváveis de serem usadas na pesquisa. Estas palavras têm função semântica importante na língua, mas elas raramente contribuem se o critério de pesquisa é simplesmente palavra-a-palavra. A lista de stop words tem a vantagem de reduzir o tamanho do índice invertido.

Pode-se observar no resultado final, que na tag cloud gerada não existem artigos, preposições ou advérbios, tendo sido removidos neste processo de stop words.

2.5. Stemmer de Porter

Segundo Frakes e Baeza-Yates, uma técnica de melhoria de desempenho em IR é fornecer aos pesquisadores maneiras de encontrar variantes morfológicas de termos de busca. Se, por exemplo, um perquisador informar um termo lematizado como parte de uma consulta, provavelmente ele também estará interessado nas variantes do termo. Usa-se o termo conjunção no sentido de indicar o ato de fundir ou combinar, como termo geral, para o processo de identificação morfológica dos termos variantes. A conjunção pode ser manual, usando alguns tipos de expressões regulares ou automático, via programas que são chamados de Stemmers. Este processo também é usado em IR para reduzir o tamanho dos arquivos de índice. Levando em consideração que um simples lema tipicamente corresponde a vários termos, seu armazenamento pode gerar um fator de compressão de 50% ou mais.

2.6. TF-IDF (Term Frequency weighted by Invers Document Frequency)

De acordo com Swiftype, é um modelo de relevância de busca usado em IR, que determina quão relevante é um documento em particular para uma dada consulta através da valoração do número de vezes que a consulta aparece dentro to *corpus* de texto pesquisável (TF) pelo número de vezes que a consulta aparece dentro do documento específico (IDF). O TF-IDF serve como uma base para a determinação da relevância em muitos pacotes de

programas de busca de código aberto. A razão porque este modelo é tão predominante deve-se ao fato dele associar uma pontuação de relevância para cada documento cujo resultado possa, então, ser ordenado em oposição ao modelo simplificado de relevância sim/não, o qual determina meramente se um documento é relevante ou não para a consulta e retorna resultados desorganizados sem pontuação de relevância.

2.7. 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 idéias 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 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, por exemplo, para agrupamento e classificação. 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 pré-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).

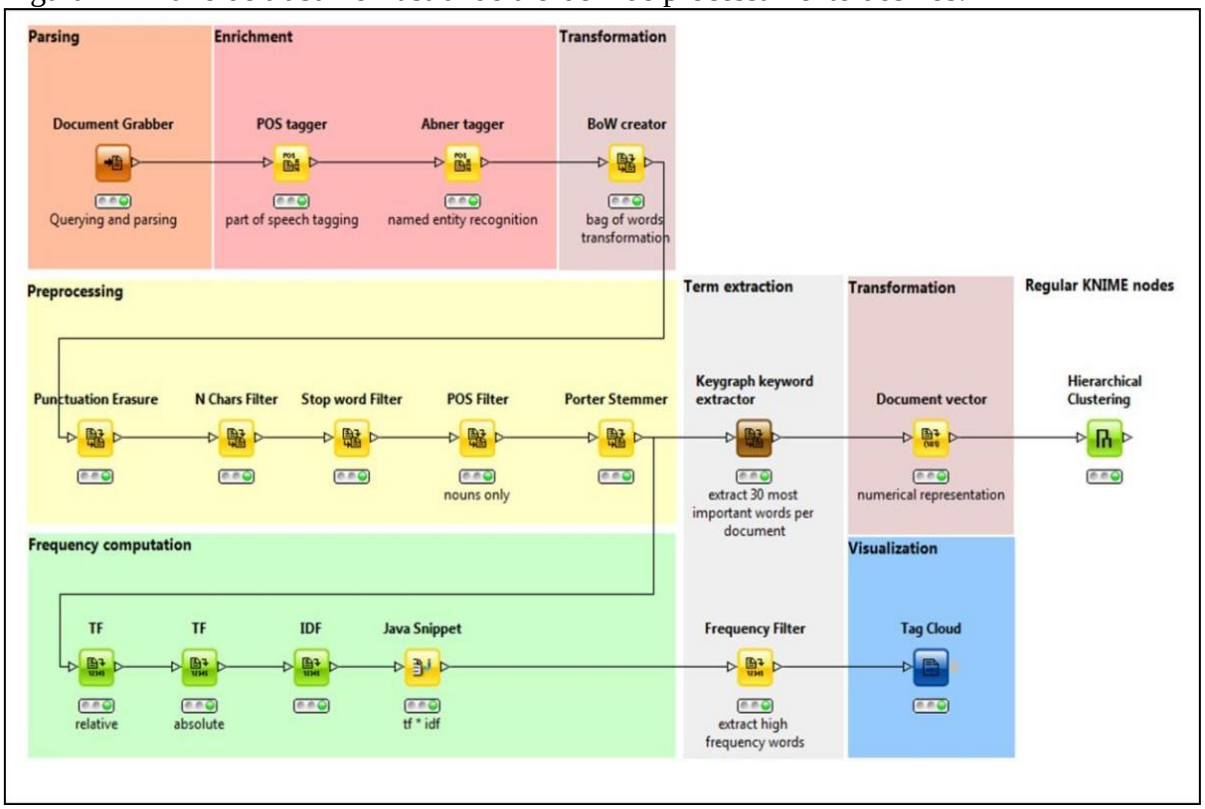
O procedimento adotado no Knime utiliza o fluxo de trabalho exemplificado na Figura 1 a seguir. Os dados processados servem de entrada para outros processos. Os resultado desses processos podem servir de entrada para outros processos e assim sucessivamente até a obtenção da saída desejada.

Para a realização do trabalho, foram realizados os seguintes passos:

1) Captura do documento, fazendo a consulta e a análise da estrutura sintática.

2) Aplicação de um POS (part-of-speech) Tagger, para identificar as classes gramaticais das palavras do documento.
3) Aplicação de um processo de reconhecimento de nomes próprios (NER).
4) Transformação do resultado dos processos anteriores em um *bag of words*.
5) Uma fase de pré-processamento na qual são eliminadas a pontuação em todo o documento, aplicados filtros e um processo de lematização (remoção de sufixos indicativos de gênero e número) (STEMMING).
6) Aplicação da técnica TF (term-frequency) – IDF (Inverse Document Frequency).
7) Extração de palavras com as mais altas frequências.
8) Geração de Tag Cloud.

Figura 1 – Fluxo de trabalho ilustrando a ordem de processamento dos nós.



Fonte: Knime.

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(@"<dir>\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 o transcript da fala do deputado no momento do voto.

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

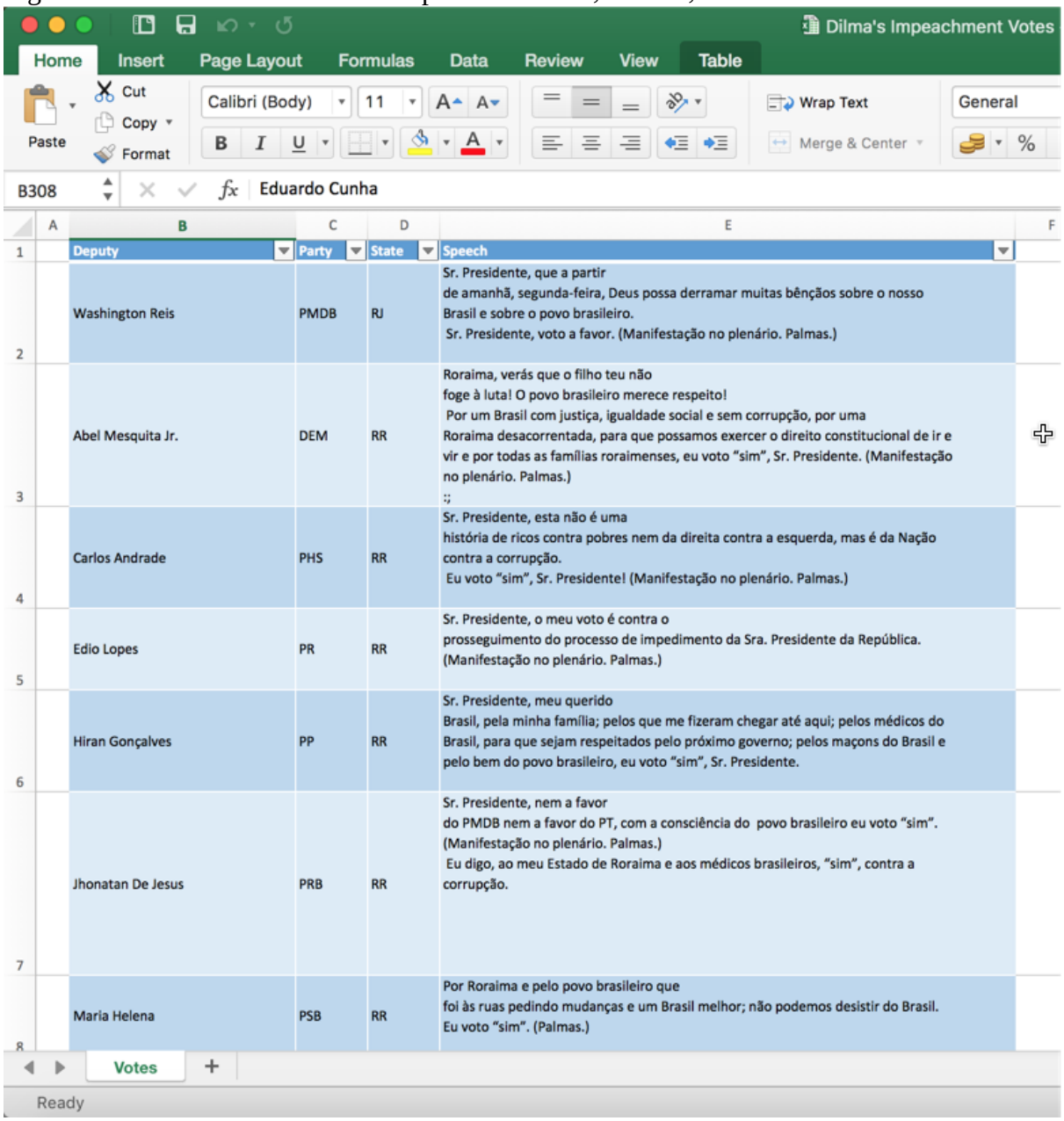
```
private static List<Vote> ParseVotes(string text)
{
    var votes = new List<Vote>();
    var sentences = text.Split(new[] {"O SR.", "A SRA."},
StringSplitOptions.None).AsEnumerable();
    // Removing white space...
    sentences = sentences.Select(s => s.TrimStart());
    // Filtering to keep only 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 # 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 a biblioteca NPOI da Neuzilla 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:

Figura 2 - Planilha com dados dos deputados: Nome, Partido, Estado e Fala durante o voto.



	Deputy	Party	State	Speech
2	Washington Reis	PMDB	RJ	Sr. Presidente, que a partir de amanhã, segunda-feira, Deus possa derramar muitas bênçãos sobre o nosso Brasil e sobre o povo brasileiro. Sr. Presidente, voto a favor. (Manifestação no plenário. Palmas.)
3	Abel Mesquita Jr.	DEM	RR	Roraima, verás que o filho teu não foge à luta! O povo brasileiro merece respeito! Por um Brasil com justiça, igualdade social e sem corrupção, por uma Roraima desacorrentada, para que possamos exercer o direito constitucional de ir e vir e por todas as famílias roraimenses, eu voto "sim", Sr. Presidente. (Manifestação no plenário. Palmas.) ;;
4	Carlos Andrade	PHS	RR	Sr. Presidente, esta não é uma história de ricos contra pobres nem da direita contra a esquerda, mas é da Nação contra a corrupção. Eu voto "sim", Sr. Presidente! (Manifestação no plenário. Palmas.)
5	Edio Lopes	PR	RR	Sr. Presidente, o meu voto é contra o prosseguimento do processo de impedimento da Sra. Presidente da República. (Manifestação no plenário. Palmas.)
6	Hiran Gonçalves	PP	RR	Sr. Presidente, meu querido Brasil, pela minha família; pelos que me fizeram chegar até aqui; pelos médicos do Brasil, para que sejam respeitados pelo próximo governo; pelos maçons do Brasil e pelo bem do povo brasileiro, eu voto "sim", Sr. Presidente.
7	Jhonatan De Jesus	PRB	RR	Sr. Presidente, nem a favor do PMDB nem a favor do PT, com a consciência do povo brasileiro eu voto "sim". (Manifestação no plenário. Palmas.) Eu digo, ao meu Estado de Roraima e aos médicos brasileiros, "sim", contra a corrupção.
8	Maria Helena	PSB	RR	Por Roraima e pelo povo brasileiro que foi às ruas pedindo mudanças e um Brasil melhor; não podemos desistir do Brasil. Eu voto "sim". (Palmas.)

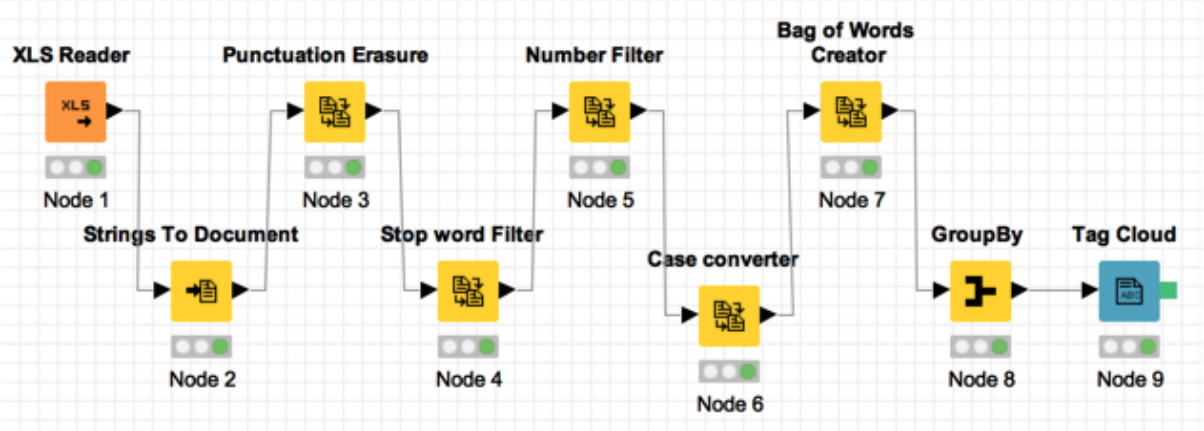
Fonte: Própria.

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.



Fonte: Knime.

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: Produz uma imagem com a 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.

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.

Esta imagem fornece elementos que nos permitem 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; status esse que em grande parte orientou a votação da maioria dos deputados.

Figura 4: Tag cloud gerada, mostrando palavras mais ditas na votação do impeachment.

Fonte: Knime.

5. Conclusão e Lições aprendidas

O resultado do presente trabalho permitiu que fosse experimentada na prática, a utilização dos conceitos de recuperação de informação aprendidos em sala de aula. Ainda que alguns processos estivessem presentes na ferramenta, não tendo sido desenvolvidos pelos autores, pode-se observar, por exemplo, que na nuvem gerada, como resultado final do processamento, não se verificaram as stop words, as quais foram removidas a seu tempo.

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

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

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 processes. 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 Retrieval. Search. Tag Cloud. Knime. Impeachment. Dilma Rousseff.

6. Referências

FRAKES, William B.; BAEZA-YATES, R. **Information Retrieval: Data Structures & Algorithms**. [S.l.: s.n.], [19--?]. Disponível em <http://hornad.fei.tuke.sk/~genci/Vyucba/OOBDS/Archiv/Podklady/TExtoveIS&IR/Information%20Retrieval%20Data%20Structures%20&%20Algorithms.PDF>. Acesso em: 20 ago. 2016.

MANNING, Christopher D.; SCHÜTZE, H. **Foundations of Statistical Natural Language Processing**. London: The MIT Press, 1999. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.2604&rep=rep1&type=pdf>. Acesso em: 20 ago. 2016.

MANSOURI, A.; AFFENDEY, L. S.; MAMAT, A. **Named Entity Recognition Approaches.** Serdong: IJCSNS, Vol. 8 No. 2, 2008. Disponível em <http://paper.ijcsns.org/07_book/ 200802/20080246.pdf>. Acesso em: 20 ago. 2016.

SWIFTYPE. **TF-IDF Search Relevance Model**. Disponível em <https://swiftype.com/search-concepts/tf-idf>. Acesso em 21 ago. 2016.

THIEL, K.; BERTHOLD, M. **The KNIME Text Processing Feature: An Introduction**. [S.l.: s.n.], 2012. Disponível em <https://www.knime.org/files/knime_text_proces sing_introduction_technical_report_120515.pdf>. Acesso em: 21 jul. 2016.