Ovídio José Francisco

# Aplicação de técnicas de Recuperação de Informação para Organização e Extração de Históricos de Decisões de Documentos de Reuniões

**Sorocaba, SP**

**25 de outubro de 2017**

# Sumário

# 1 Introdução

A popularização dos computadores possibilitou o armazenamento cada vez maior de conteúdos digitais, entre esses, o formato textual como em livros, documentos, e-mails, redes sociais e páginas web. A produção de textos gera fontes de informações em volumes crescentes que podem superar a capacidade humana de analisá-los manualmente. Essa dificuldade incentiva a pesquisa de ferramentas automáticas para manipulação de dados não estruturados. Assim, os processos de extração automática de conhecimento em coleções textuais são essenciais, e ao mesmo tempo, constituem um desafio, devido às características de documentos textuais como o formato não estruturado e trechos com diferentes níveis e importância, desde informações essenciais até textos pouco informativos e em alguns casos até irrelevantes.

Além dos tipos de informações mais comuns que são armazenados no formato textual, como e-mails, relatórios, artigos e postagens em redes sociais, têm-se também o armazenamento das atas de reuniões, as quais permitem às organizações a documentação oficial de reuniões em arquivos digitais, facilitando a confecção, compartilhamento e consulta às decisões tomadas. Reuniões são tarefas presentes em atividades corporativas, ambientes de gestão e organizações de um modo geral, onde discute-se problemas, soluções, propostas, planos, questionamentos, alterações de projetos e frequentemente são tomadas decisões importantes. A comunicação entre os membros da reunião é feita de forma majoritariamente verbal. Para que seu conteúdo possa ser registrado e externalizado, adota-se a prática de registrar seu conteúdo em documentos, os quais chamamos de atas de reunião.

Porém, armazenar e recuperar atas em formato textual oferece desafios. As atas de reunião possuem características particulares, frequentemente apresentam um texto com poucas quebras de parágrafo e sem marcações de estrutura, como capítulos, seções ou quaisquer indicações sobre o tema do texto. Devido a fatores como a não estruturação e volume dos textos, a localização de um assunto em uma coleção de atas é uma tarefa custosa, especialmente considerando o seu crescimento em uma instituição. As organizações costumam manter seus documentos eletrônicos organizados em pastas e nomeá-los com informações básicas sobre a reunião a que se refere como a data e alguma referência cronológica, por exemplo "37ª Reunião Ordinária do Conselho ...". Essa forma de organização facilita a localização dos arquivos com ferramentas que fazem buscas pelo nome dos arquivos e pastas. Contudo, essa prática costuma ser insuficiente, pois uma busca pelo conteúdo dos textos usa-se ferramentas computacionais baseadas em localização de palavras-chave que além de encontrar ocorrências das palavras podem oferecer recursos como operadores *and*

e *not* ou ainda suporte às expressões regulares. Esse recurso, conhecido como *grepping*[1], traz resultados satisfatórios em muitos casos. Por outro lado, o *grepping* traz algumas desvantagens como: 1) transfere certa complexidade da tarefa ao usuário. 2) buscas em grandes coleções de documentos tornam-se mais lentas. 3) não há suporte a padrões mais flexíveis como a proximidade entre as palavras ou palavras que estejam na mesma sentença. 4) o retorno ao usuário são os documento integrais, o que pode exigir uma segunda busca dentro de um documento para encontrar o trecho desejado.

Para superar essas limitações tem sido utilizadas técnicas de aprendizado de máquina por meio de diversas abordagens. Por exemplo, elas vêm sendo empregadas na organização, gerenciamento, recuperação de informação e extração de conhecimento, como a extração de tópicos e a categorização de automática de documentos. Essas técnicas permitem melhorar a busca por informações em atas de reunião.

Nas reuniões do conselho de um programa de pós-graduação de uma universidade, são decididos, por exemplo, quais são os critérios para credenciamento e permanência de docentes no programa. Ao longo do tempo, esse tema pode ser discutido e mencionado diversas vezes, podendo os critérios inclusive passar por significativas alterações, devido a diversos fatores. O coordenador do programa pode desejar recuperar qual foi a decisão mais recente, para poder aplicar os critérios a um potencial novo membro do programa, ou os membros do conselho podem desejar rever o histórico de tudo o que já foi discutido/decidido sobre o tema, para poder propor alterações nas regras, de forma mais adequada.

Uma vez que a ata registra a sucessão de assuntos discutidos na reunião, um sistema de recuperação de informação deve retornar ao usuário apenas o trecho que trate do assunto pesquisado ao invés do documento inteiro. Assim, cada trecho com um assunto predominate pode ser considerado um subdocumento. Portanto, em primeiro lugar, há a necessidade de descobrir onde há mudanças de assunto no texto, que pode ser atendida com técnicas de segmentação automática de textos.

A tarefa de segmentação automática de textos, ou segmentação textual consiste em dividir um texto em partes que contenham um significado relativamente independente. Em outras palavras, é identificar as posições nas quais há uma mudança significativa de assunto. É útil em aplicações que trabalham com textos sem indicações de quebras de assunto, ou seja, não apresentam seções ou capítulos, como transcrições automáticas de áudio, vídeos e grandes documentos que contêm vários assuntos como atas de reunião e notícias. Pode ser usada para melhorar o acesso a informação solicitada por meio de uma consulta, onde é possível oferecer porções menores de texto mais relevantes ao invés de exibir um documento grande que pode conter informações menos pertinentes. A navegação pelo documento pode ser aprimorada, em especial na utilização por usuários com deficiência visual, os quais utilizam sintetizadores de texto como ferramenta de acessibilidade. Além disso, encontrar

[1] O nome *grepping* é uma referência ao comando `grep` do Unix

pontos onde o texto muda de assunto, pode ser útil como etapa de pré-processamento em aplicações voltadas ao entendimento do texto, principalmente em textos longos (CHOI, 2000).

Assim, nesse contexto, este trabalho propõe a investigação do uso de mineração de texto e as técnicas que constituem o estado da arte na área para o desenvolvimento de uma ferramenta para extração automática de históricos de decisão em atas de reuniões.

# 2 Conceituação Teórica

A popularidade dos computares permite a criação e compartilhamento de textos onde a quantidade de informação facilmente extrapola a capacidade de humana de leitura e análise de coleções de documentos, estejam eles disponíveis na Internet ou em computadores pessoais. A necessidade de simplificar e organizar grandes coleções de documentos criou uma demanda por modelos de aprendizado de máquina para extração de conhecimento em bases textuais. Para esse fim, foram desenvolvidas técnicas para descobrir, extrair e agrupar textos de grandes coleções, entre essas, a modelagem de tópicos (BLEI, 2012).

## 2.1 Segmentação Textual

A tarefa de segmentação textual consiste em dividir um texto em partes ou segmentos que contenham um significado relativamente independente. Em outras palavras, é identificar as posições nas quais há uma mudança significativa de assuntos. As técnicas de segmentação textual consideram um texto como uma sequência linear de unidades de informação que podem ser, por exemplo, cada termo presente no texto, os parágrafos ou as sentenças. Cada unidade de informação é um elemento do texto que não será dividido no processo de segmentação e cada ponto entre duas unidades é considerado um candidato a limite entre segmentos. Nesse sentido, um segmento pode ser visto como uma sucessão de unidades de informação que compartilham o mesmo assunto.

Para encontrar essas posições, trabalhos anteriores se apoiam na ideia de que a mudança de assunto em um texto é acompanhada de uma proporcional mudança de vocabulário. Essa ideia, chamada de coesão léxica, sugere que a distribuição das palavras é um forte indicador da estrutura do texto. A partir disso, vários algoritmos foram propostos baseados na ideia de que um segmento pode ser identificado pela análise das palavras que o compõe (CHEN et al., 2017; FERRET, 2009; SAKAHARA; OKADA; NITTA, 2014).

Uma vez que a coesão léxica é pressuposto básico da maioria dos algoritmos, o cálculo da similaridade entre unidades de informação é fundamental. Uma medida de similidade frequentemente utilizada é o cosseno, apresentada na Equação 2.1, onde dadas duas unidades de informação, $x$ e $y$, $f_{x,j}$ é a frequência do termo $j$ em $x$ e $f_{y,j}$ é a frequência do termo $j$ em $y$.

$$Sim(x, y) = \frac{\Sigma_j f_{x,j} \times f_{y,j}}{\sqrt{\Sigma_j f_{x,j}^2 \times \Sigma f_{y,j}^2}} \tag{2.1}$$

### 2.1.1 Algoritmos

Entre os trabalhos tradicionais da literatura podemos citar o *TextTiling* (HEARST, 1994) e o *C99* (CHOI, 2000). O *TextTiling* é um algoritmo baseado em janelas deslizantes, em que, para cada candidato a limite, analisa-se o texto circundante. Um limite ou quebra entre segmentos é identificado sempre que há uma queda abaixo de um limiar na similaridade entre as unidades que antecedem e precedem o ponto candidato cai abaixo de um limiar. O *TextTiling* recebe uma lista de candidatos a limite, usualmente finais de parágrafo ou finais de sentenças. Para cada posição candidata são construídos 2 blocos, um contendo sentenças que a precedem e outro com as que a sucedem. O tamanho desses blocos é um parâmetro a ser fornecido ao algoritmo e determina o tamanho mínimo de um segmento. Em seguida, os blocos de texto são representados por vetores que contém as frequências de suas palavras. Então, usa-se cosseno (Equação 2.1) para calcular a similaridade entre os blocos adjacentes a cada candidato e identifica-se uma transição entre tópicos pelos vales na curva de dissimilaridade.

O *TextTiling* possui baixa complexidade computacional. Por outro lado, algoritmos mais complexos, como os baseados em matrizes de similaridade, apresentam acurácia relativamente superior como apresentado em (CHOI, 2000; KERN; GRANITZER, 2009; MISRA et al., 2009).

Outro algoritmo frequentemente referenciado na literatura é o C99 o qual é baseado em *ranking*. Embora muitos trabalhos utilizem matrizes de similaridades para pequenos segmentos, o cálculo de suas similaridades não é confiável, pois uma ocorrência adicional de uma palavra pode causar certo impacto e alterar o cálculo da similaridade (CHOI, 2000). Além disso, o estilo da escrita normalmente não ser constante em todo o texto. Por exemplo, textos iniciais dedicados a introdução costumam apresentar menor coesão do que trechos dedicados a um tópico específico. Portanto, comparar a similaridade entre trechos de diferentes regiões não é apropriado. Devido a isso, as similaridades não podem ser comparadas em valores absolutos. Então, contorna-se esse problema fazendo uso de *rankings* de similaridade para encontrar os segmentos de texto. Para isso, o C99 constrói uma matriz que contém as similaridades de todas as unidades de informação (normalmente sentenças ou parágrafos). Em seguida, cada valor na matriz de similaridade é substituído por seu *ranking local*. Para cada elemento da matriz, seu *ranking* é o número de elementos vizinhos com valor de similaridade menor que o seu. Então, cada elemento e comparado com seus vizinhos dentro de uma região denominada máscara.

Na Figura 2.1.1 é destacado um quadro 3 x 3 de uma matriz em que cada elemento é a similaridade entre duas unidades de informação. Tomando como exemplo o elemento com valor $0,5$, a mesma posição na matriz de *rankings* terá o valor 4, pois esse é o número de vizinhos com valores inferiores a $0,5$ dentro do quadro analisado na matriz de similaridades. Da mesma forma, na Figura 2.1.1 para o valor $0,2$ a matriz de *rankings* conterá o valor 1

na mesma posição.

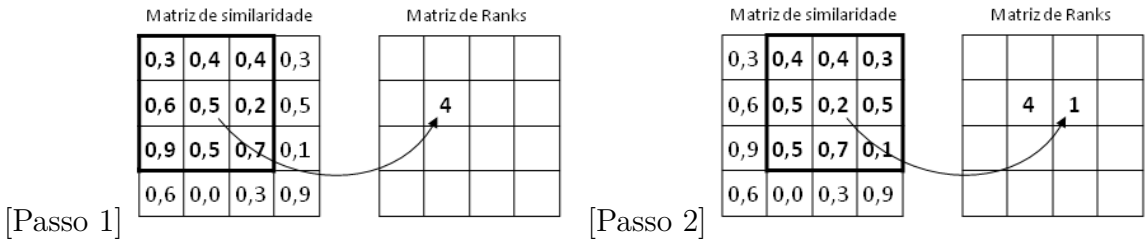


Figura 1 – Exemplo de construção de uma matriz de rankings.

Finalmente, com base na matriz de *ranking*, o C99 utiliza um método de *clustering* baseado no algoritmo de maximização de Reynar para identificar os limites entre os segmentos.

### 2.1.2 Medidas de Avaliação

As medidas de avaliação tradicionais, como acurácia, precisão e revocação, podem não ser confiáveis, por não considerarem a distância entre os limites, mas penalizam o algoritmo sempre que um limite que não coincide perfeitamente com a referência. Essas medidas podem ser mais adequadas quando necessita-se de segmentações com maior exatidão. Em outras palavras, computam apenas os erros do algoritmo quando se detecta falsos positivos ou falsos negativos, o que nesse contexto de segmentação textual pode não ser suficiente, dado a subjetividade da tarefa. Além dessas medidas, que consideram apenas se um segmento foi perfeitamente definido conforme uma referência, pode-se também considerar a distância entre o segmento extraído automaticamente e o segmento de referência (KERN; GRANITZER, 2009). Chama-se *near misses* o caso em que um limite identificado automaticamente não coincide exatamente com a referência, mas é necessário considerar a proximidade entre eles.

Na Figura 2 é apresentado um exemplo com duas segmentações extraídas automaticamente e uma referência. Em ambos os casos não há nenhum verdadeiro positivo, o que implica em zero para os valores de precisão, acurácia, e revocação, embora o resultado do algoritmo A possa ser considerado superior ao primeiro se levado em conta a proximidade dos limites.

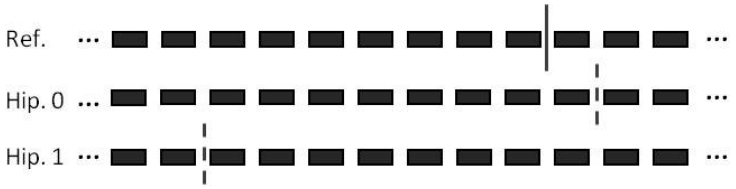


Figura 2 – Exemplos de *near missing* e falso positivo puro. Os blocos indicam uma unidade de informação e as linha verticais representam uma transição de assunto.

Considerando o conceito de *near misses*, algumas soluções foram propostas. As medidas de avaliação mais utilizadas são a $P_k$ e *WindowDiff*.

Proposta por (BEEFERMAN; BERGER; LAFFERTY, 1999), $P_k$, atribui valores parciais a *near misses*, ou seja, limites sempre receberão um peso proporcional à sua proximidade, desde que dentro de um janela de tamanho $k$. Para isso, esse método move uma janela de tamanho $k$ ao longo do texto. A cada passo verifica, na referência e na hipótese, se o início e o final da janela estão ou não dentro do mesmo segmento, então, penaliza o algoritmo caso não concorde com a referência. Ou seja, dado duas palavras de distância $k$, o algoritmo é penalizado quando não concordar com a segmentação de referência se as palavras estão ou não no mesmo segmento. O valor de $k$ é calculado como a metade da média dos comprimentos dos segmentos reais. Como resultado, é retornado a contagem de discrepâncias divida pelo quantidade de segmentações analisadas. $P_k$ é uma medida de dissimilaridade entre as segmentações e pode ser interpretada como a probabilidade de duas sentenças extraídas aleatoriamente pertencerem ao mesmo segmento.

*WindowDiff* é uma medida alternativa à $P_k$. De maneira semelhante, move uma janela pelo texto e penaliza o algoritmo sempre que o número de limites proposto pelo algoritmo não coincidir com o número de limites esperados para aquela janela. Ou seja, o algoritmo é penalizado quando não concordar com a segmentação de referência quanto ao número de segmentos na janela. Assim, consegue manter a sensibilidade a *near misses* e além disso, considerar o tamanho das janelas. A fim de melhor equilibrar o peso dos falsos positivos em relação a *near misses*, dobra-se a penalidade para falsos positivos, evitando-se a supervalorização dessa medida.

As medidas *WindowDiff* e $P_k$, consideram a quantidade e proximidade entre os limites, sendo mais tolerantes a pequenas imprecisões. Essa é uma característica desejável, visto que as segmentações de referência possuem diferenças consideráveis. *WindowDiff* equilibra melhor os falsos positivos em relação a *near misses*, ao passo que $P_k$ os penaliza com peso maior. Isso significa que segmentadores melhores avaliados em $P_k$ ajudam a selecionar as configurações que erram menos ao separar trechos de texto com o mesmo assunto, enquanto *WindowDiff* é mais tolerante nesse aspecto. De maneira geral, observa-se melhores resultados de *WindowDiff* quando os algoritmos aproximam a quantidade de segmentos automáticos da quantidade de segmentos da referência. Por outro lado, $P_K$ avalia melhor as configurações que retornam menos segmentos. Contudo, não é possível definir um valor adequado, uma vez que os segmentadores humanos frequentemente apontam segmentações diferentes.

Ao final do processo de segmentação, são produzidos fragmentos de documentos, aqui chamados de subdocumentos. Esses subdocumentos contém um texto, assim como no documento original, em um estágio de processamento inicial, pois ainda não estão estruturados. Ocorre que as técnicas de aprendizado de máquina exigem uma representação

estruturada dos textos conforme será visto na Seção 2.2.

## 2.2 Representação de Textos

Uma das formas mais comuns para que a grande maioria dos algoritmos de aprendizado de máquina possa extrair padrões das coleções de textos é a representação no formato matricial conhecido como Modelo Espaço Vetorial ( *Vectorial Space Model* - VSM) (**??**), onde os documentos são representados como vetores em um espaço Euclidiano $T$-dimensional em que cada termo extraído da coleção é representado por uma dimensão. Assim, cada componente de um vetor expressa a relação entre os documentos e as palavras. Essa estrutura é conhecida como *document-term matrix* ou matriz documento-termo. Uma das formas mais populares para representação de textos é conhecida como *Bag Of Words* a qual é detalhada a seguir.

### 2.2.1 *Bag Of Words*

Após a etapa de pré-processamento, onde as *stop words* foram removidas e as palavras reduzidas ao seus radicais (*stemming*), tem-se uma versão reduzida, com menos atributos, dos dados originais. Essa versão pode ser facilmente convertida em uma tabela ou matriz documento-termo. Essa representação, conhecida como *bag-of-words*, onde cada termo é transformado em um atributo (*feature*) (**??**). Essa representação é mostrada pela Tabela 1.

| | $t_1$ | $t_2$ | $t_j$ | ... | $t_n$ |
|---|---|---|---|---|---|
| $d_1$ | $a_{11}$ | $a_{12}$ | $a_{1j}$ | ... | $a_{1n}$ |
| $d_2$ | $a_{21}$ | $a_{22}$ | $a_{2j}$ | ... | $a_{2n}$ |
| $d_i$ | $a_{i1}$ | $a_{i2}$ | $a_{ij}$ | ... | $a_{in}$ |
| ... | ... | ... | ... | ... | ... |
| $d_m$ | $a_{m1}$ | $a_{m2}$ | $a_{mj}$ | ... | $a_{mn}$ |

Tabela 1 – Coleção de documentos na representação *bag-of-words*

Essa forma de representação sintetiza a base de documentos em um contêiner de palavras, ignorando a ordem em que ocorrem, bem como pontuações e outros detalhes, preservando apenas o peso de determinada palavra nos documentos. É uma simplificação de toda diversidade de informações contidas na base de documentos sem o propósito de ser uma representação fiel do documento, mas oferecer a relação entre as palavras e os documentos a qual é suficiente para a maioria dos métodos de aprendizado de máquina (**??**).

Nessa representação, $a_{ij}$ é o peso do termo $j$ no documento $i$ e indica a sua relevância dentro da base de documentos. As medidas mais tradicionais para o cálculo desses pesos são a binária, onde o termo recebe o valor 1 se ocorre em determinado documento ou 0 caso

contrário; *document frequency*, que é o número de documentos no qual um termo ocorre; *term frequency - tf*, atribui-se ao peso a frequência do termo dentro de um determinado documento; *term frequency-inverse document frequency, tf-idf*, pondera a frequência do termo pelo inverso do número de documentos da coleção em que o termo ocorre.

## 2.3 Modelos de Extração de Tópicos

Os modelos de extração de tópicos são abordagens não-supervisionadas que visam descobrir padrões latentes nas relações entre os documentos e seus termos. Baseiam-se na premissa de que um documento é produzido a partir de tópicos previamente definidos que determinam os termos a serem utilizados em um documento. Nesse contexto, um documento é uma mistura de tópicos onde cada termo presente no documento pode ser associado a um tópico. Um tópico por sua vez, é uma estrutura com valor semântico que representada por um conjunto de termos e seus pesos que indicam o quão significante esses termos são para um assunto e pode ser útil para o entendimento do tema ao qual o tópico trata.

Para descobrir esses tópicos, algumas técnicas foram propostas. Em termos de metodologia, a maioria dos trabalho enquadram-se em duas principais categorias, os modelos não-probabilísticos e os modelos probabilísticos.

Os modelos não-probabilísticas baseiam-se em técnicas de fatoração de matrizes, onde a matrix documento-termo é projetada em um espaço com menor dimensionalidade chamado *Latent Semantic Space*. Seja $d \in D = \{d_1, \cdots, d_n\}$ o vetor que representa a coleção de documentos, $t \in T = \{t_1, \cdots, t_m\}$ seus termos distintos e $z \in Z = \{z_1, \cdots, z_k\}$ seus tópicos. Esses métodos aprendem decompondo a matriz documento-termo $W$, em duas matrizes $Z$ e $A$, tal que a resultante de $ZA$ seja uma aproximação da matriz $W$ original. Mais formalmente tem-se:

$$Z \cdot A = \hat{W} \approx W \tag{2.2}$$

A matriz $A$ corresponde a matriz documento-tópico e possui dimensão $k \times n$. $Z$ corresponde a matriz termo-tópico e possui dimensão $m \times k$ onde $n$ é o número de termos, $m$ é o número de documentos da coleção e $k$ é a quantidade de tópicos a serem extraídos. Uma vez que $k \ll n, m$, então $A$ e $Z$ são menores que a matriz de entrada, o que resulta em uma versão comprimida da matriz original, pois $k \cdot n + m \cdot k \ll n \cdot m$. Ao final, obtém-se uma representação documento-tópico que atribui um peso para cada tópico em cada documento da coleção e uma representação termo-tópico que representa a probabilidade de ocorrência de um termo em um documento dado que o tópico está presente no documento.

Nesse sentido, o *Latente Semantic Indexing* (LSI) usa a técnica chamada *Singular*

*Value Decomposition* (SVD) para encontrar padrões no relacionamento entre assuntos e termos em uma coleção de texto não estruturada. Entretanto, esse método não fornece uma interpretação para elementos com valores negativos (DEERWESTER et al., 1990) (CHENG et al., 2013).

Outro modelo popular é o *Non-Negative Matrix Factorization* (NMF). Diferente do LSI, no processo de fatoração apenas operações aditivas são permitidas, o que garante que as matrizes resultantes não possuem elementos negativos, permitindo uma interpretação mais intuitiva de seus valores. Além disso, o processo de fatoração proporciona a propriedade de *clustering*, ou seja, agrupar as colunas da matriz $W$, e dessa forma, oferece a característica interessante de agrupar os documentos da coleção.

Os modelos probabilísticos consideram os documentos como uma mistura de tópicos e um tópico como uma distribuição probabilística sobre os termos. O processo de elaboração do documento a partir desses tópicos é chamado de processo generativo ou modelo generativo, o qual é desconhecido porém pode ser estimado com base nos termos presentes no documento, também chamados de variáveis observáveis. Assim, o processo de extração de tópicos consiste em estimar o modelo generativo que deu origem ao documento. O PLSA foi um dos primeiros a estender o modelo LSA e formalizar a extração de tópicos probabilísticos. De maneira similar ao LSA, o esse modelo decompõe uma matriz esparsa a fim de reduzir a dimensionalidade. O PLSA cria um modelo estatístico chamado *aspect model* que associa os tópicos as variáveis observáveis atribuindo probabilidades às ligações entre os tópicos e os documentos e entre as palavras e os tópicos. Assim, cada documento pode ser representado como a probabilidade de um tópico estar presente, $P(z|d)$. E a probabilidade de um termo ocorrer dado que um tópico esta presente, $P(t|z)$. Em comparação ao LSA, é considerado uma método mais robusto por proporcionar uma interpretação probabilística. Por outro lado, esse modelo apresenta desvantagens como o número de parâmetros do modelo que cresce linearmente com o número de documentos da coleção que pode ocasionar *overfitting*.

A fim de contornar esses problemas, o LDA estende o modelo PLSA incorporando um modelo generativo onde os cada tópico obedece à distribuição multivariada de *Dirichlet* o que o torna menos propenso ao *overfitting* e capaz de inferir tópicos a documentos ainda não observados. É referenciado na literatura como estado da arte sobre modelos probabilísticos de extração de tópicos e influencia uma grande quantidade de trabalhos, tornando-se base para novos modelos. No modelo LDA, o processo de geração de palavras se dá em duas etapas:

1. Atribui-se uma distribuição aleatória sobre os tópicos.
2. Para cada termo no documento:
    a) Atribui-se aleatoriamente a um tópico da distribuição obtida na etapa 1;

b) Seleciona-se aleatoriamente uma palavra do tópico correspondente.

Assim cada documento é associado a múltiplos tópicos com proporções distintas (etapa 1). Cada palavra do documento é obtida de um tópico específico (etapa 2.b) que foi anteriormente obtido a partir da distribuição de tópicos do documento (etapa 2.a). Isso permite ao modelo LDA atribuir, para cada documento, múltiplos tópicos com proporções distintas (BLEI, 2012).

Os modelos de extração de tópicos foram inicialmente propostos para utilização em mineração texto onde são empregados na redução de dimensionalidade, extração de informações em textos, bem como na organização e recuperação de documentos, sendo utilizados para mensurar a relevância de um termo ou conjunto de termos para determinado assunto ou documento. Visto a popularidade nessas tarefas e flexibilidade dos modelos, logo notou-se sua utilidade em outros tipos de dados com atributos discretos como imagens, grafos e genética.

## 2.4 Trabalhos Relacionados

# 3 Proposta

Essa seção apresenta as etapas de desenvolvimento do sistema, bem como o seu funcionamento geral, desde a preparação dos documentos até a entrega dos históricos de ocorrência ao usuário. Inicialmente serão descritos a seleção e pré-processamento. Em seguida, será relatado como as técnicas de mineração de texto e resgate de informação são utilizadas nesse trabalho.

O objetivo do sistema é permitir ao usuário consultar uma coleção de documentos de reuniões a fim de obter todo o histórico de ocorrências de um determinado tema pesquisado, podendo identificar nos documentos onde o tema foi mencionado como informe ou onde houve uma decisão sobre o tema. Para isso, o sistema é divido em dois módulos principais: Módulo de preparação e manutenção e Módulo de consulta.

O módulo de preparação e manutenção recebe uma coleção de documentos e produz uma estrutura de dados interna, que é utilizada pelo módulo de consulta, que por sua vez, é responsável por receber a intensão usuário, proceder a busca na estrutura de dados interna e retornar os trechos associados com a intensão do usuário, tanto quanto ao assunto como no tipo de ocorrência. A Figura 3 mostra a visão geral do sistema com suas principais entradas e saídas.

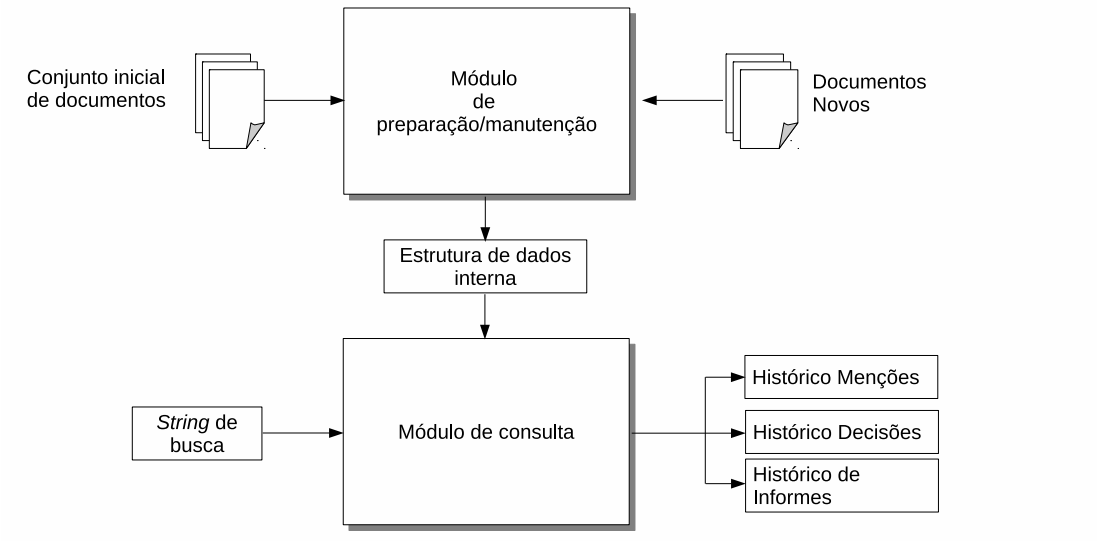


Figura 3 – Visão geral do sistema

## 3.1 Módulo de preparação e manutenção

O módulo de preparação e manutenção tem como funções principais dividir cada ata em em segmentos de texto que contêm um assunto predominante, e descrevê-los por

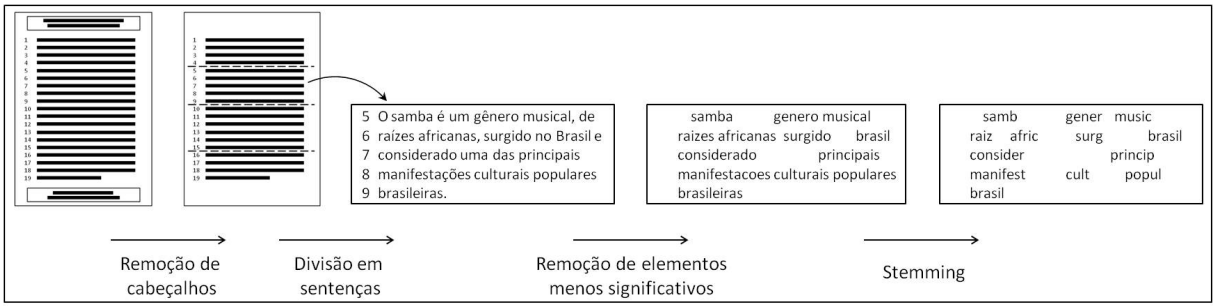


Figura 4 – Exemplo de pré-processamento.

meio de técnicas de extração tópicos e classificação. Além disso, produz uma estrutura de dados que registra quais assuntos foram tratados na reunião, bem como o trecho do documento onde é discutido.

### 3.1.1 Preparação dos documentos

As atas são normalmente armazenadas em arquivos do tipo *pdf*, *doc*, *docx* ou *odt* que normalmente possuem formato binário. O texto deve ser preparado para os métodos de MT e RI. Inicialmente, o texto puro é extraído e passa por processos de transformação conforme apresentados a seguir.

1. Remoção de cabeçalhos e rodapés: as atas contém trechos que podem ser considerados pouco informativos e descartados durante o pré-processamento, como cabeçalhos e rodapés que se misturam aos tópicos tratados na reunião, podendo ser inseridos no meio de um tópico prejudicando tanto os algoritmos de MT e RI, quanto a leitura do texto pelo usuário.

2. Identificação de finais sentenças: devido ao estilo de pontuação desses documentos, como encerrar sentenças usando um *";"* e inserção de linhas extras, foram usadas as regras especiais para identificação de finais de sentença. Cada final de sentença é identificado e marcado com uma *string* especial.

3. Redução de termos: eliminou-se as *stop words* por meio de uma lista de 438 palavras. Além disso, eliminou-se a acentuação, sinais de pontuação, numerais e todos os *tokens* menores que três caracteres.

4. *Stemming*: extraiu-se o radical de cada palavra. Para isso, as letras foram convertidas em caixa baixa e aplicou-se o algoritmo *Orengo*para remoção de sufixos.

A Figura 4 mostra a etapa de preparação de um documento em português.

#### 3.1.1.1 Segmentação

Como já mencionado, uma ata registra a sucessão de assuntos discutidos em uma reunião, porém apresenta-se com poucas quebras de parágrafo e sem marcações de estrutura, como capítulos, seções ou quaisquer indicações sobre o assunto do texto. Portanto, faz-se necessário descobrir quando há uma mudança de assunto no texto da ata. Para essa tarefa, as técnicas de segmentação de texto recebem uma lista de sentenças, da qual considera cada ponto entre duas sentenças como candidato a limite, ou seja, um ponto onde há transição entre assuntos.

Para esse trabalho, os algoritmos *TextTiling* (HEARST, 1994) e *C99* (CHOI, 2000) foram avaliados na tarefa de segmentação dos textos extraídos das atas conforme apresentado a seguir.

#### 3.1.1.2 Avaliação dos Segmentadores

Para que se possa avaliar um segmentador automático de textos é preciso uma referência, isto é, um texto com os limites entre os segmentos conhecidos. Essa referência, deve ser confiável, sendo uma segmentação legítima que é capaz de dividir o texto em porções relativamente independentes, ou seja, uma segmentação ideal.

#### 3.1.1.3 Segmentação de Referência

A fim de obter um conjunto de documentos segmentados que possam servir como referência na avaliação, os documentos coletados foram segmentados manualmente por profissionais que participam de reuniões. Para isso, utilizou-se um *software*, desenvolvido com esse objetivo especifico, que permitiu aos voluntários visualizar um documento, e indicar livremente as divisões entre segmentos. Com o uso desse *software* foram coletados os dados de seis atas segmentadas por dois participantes das reuniões, os quais serviram como referência para a avaliação dos algoritmos. O *software* desenvolvido para segmentação manual está disponível para utilização e consulta em

Os arquivos gerados foram tratados para que os segmentos sempre terminem em uma sentença reconhecida pelo algoritmo, uma vez que as sentenças são a unidade mínima de informação nesse trabalho.

A Tabela 2 contém, para cada ata, a quantidade de sentenças e a quantidade de segmentos identificadas pelos participantes.

### 3.1.2 Configuração experimental

O *TextTiling* permite ajustarmos dois parâmetros, sendo o tamanho da janela e o passo. Por meio de testes empíricos escolheu-se os valores os valores 20, 40 e 60 para o tamanho da janela e 3, 6, 9 e 12 para o passo. Gerando ao final 20 configurações.

| **Ata** | **Sentenças** | **Participante 1** | **Participante 2** |
|---|---|---|---|
| Ata 1 | 18 | 7 | 15 |
| Ata 2 | 26 | 9 | 20 |
| Ata 3 | 24 | 7 | 15 |
| Ata 4 | 32 | 9 | 17 |
| Ata 5 | 25 | 11 | 17 |
| Ata 6 | 10 | 4 | 9 |

Tabela 2 – Quantidade de sentenças e segmentos de referência por ata.

O *C99* permite o ajuste de três parâmetros, sendo, o primeiro a quantidade segmentos desejados, uma vez que, não se conhece o número ideal de segmentos e os documentos não apresentam muitos candidatos, calculou-se uma proporção dos candidatos a limite. Para isso atribuiu-se os valores 0,2; 0,4; 0,6; 0,8. O segundo parâmetro, o tamanho do quadro utilizado para gerar a matriz de ranking, atribuiu-se os valores 9 e 11, sendo 11 o valor padrão da apresentado pelo autor. O algoritmo permite ainda indicar se as sentenças serão representados por vetores contendo a frequência ou o peso de cada termo. Ambas as representações foram utilizadas. Considerando todos os parâmetros, foram geradas 16 configurações para o algoritmo *C99*.

#### 3.1.2.1 Critérios de avaliação

Para fins de avaliação desse trabalho, um bom método de segmentação é aquele cujo resultado melhor se aproxima de uma segmentação manual, sem a obrigatoriedade de estar perfeitamente alinhado com tal. Ou seja, visto o contexto das atas de reunião, e a subjetividade da tarefa, não é necessário que os limites entre os segmentos (real e hipótese) sejam idênticos, mas que se assemelhem em localização e quantidade.

Os algoritmos foram comparados com a segmentação fornecida pelos participantes das reuniões. Calculou-se as medidas mais aplicadas à segmentação textual, $P_k$ e *WindowDiff*. Além dessas, computou-se também as medidas tradicionais acurácia, precisão, revocação e $F^1$ para comparação com outros trabalhos que as utilizam.

Inicialmente, calculou-se as medidas configurando cada algoritmo conforme mostrado na Subseção 3.1.2, sem aplicar o pré-processamento. O teste de Friedman com pós-teste de Nemenyi foi utilizado para gerar um ranking das melhores configurações para cada medida calculada. Com isso, foi possível descobrir quais valores otimizam um algoritmo para uma medida, desconsiderando o pré-processamento.

A fim de conhecer o impacto do pré-processamento, repetiu-se os testes com o texto pré-processado. Com isso, descobriu-se quais valores otimizam os algoritmos para cada medida, considerando essa etapa.

Com os testes anteriores obteve-se, para cada medida, 4 configurações, levando

em conta ambos os algoritmos e a presença ou ausência do pré-processamento. Novamente utilizou-se o teste de Friedman e Nemenyi e descobriu-se, para cada medida, qual configuração a otimiza. Os resultados completos estão disponíveis para consulta em .

#### 3.1.2.2 Resultados

Obteve-se, por meio dos testes estatísticos apresentados, as melhores configurações para as principais medidas de avaliação de segmentadores. Com essas configurações calculou-se a média de cada medida considerando o conjunto de documentos. Na Tabela 3 são apresentadas, as médias obtidas com o *TextTiling* bem como as configurações utilizadas, onde **J** é o tamanho da janela e **P** é o passo.

| | Sem Pré-processamento | | | Com Pré-processamento | | |
|---|---|---|---|---|---|---|
| **Medida** | **J** | **P** | **Média** | **J** | **P** | **Média** |
| $P_k$ | 50 | 9 | 0,142 | 50 | 9 | 0,144 |
| *WindowDiff* | 50 | 6 | 0,387 | 40 | 9 | 0,396 |
| Acurácia | 50 | 6 | 0,612 | 40 | 9 | 0,603 |
| Precisão | 40 | 9 | 0,611 | 50 | 12 | 0,613 |
| Revocação | 20 | 3 | 0,886 | 20 | 3 | 0,917 |
| $F^1$ | 30 | 6 | 0,605 | 40 | 3 | 0,648 |

Tabela 3 – Resultados obtidos com o *TextTiling*

Na Tabela 4 são apresentadas, as médias obtidas com o *C99* bem como as configurações utilizadas, onde **S** é a proporção de segmentos em relação a quantidade de candidatos, **M** é o tamanho do quadro utilizado para criar a matriz de *rankings* e **W** indica se os segmentos são representados por vetores contendo a frequência ou um peso das palavras.

| | Sem Pré-processamento | | | | Com Pré-processamento | | | |
|---|---|---|---|---|---|---|---|---|
| **Medida** | **S** | **M** | **W** | **Média** | **S** | **M** | **W** | **Média** |
| $P_k$ | 20 | 9 | Sim | 0,134 | 20 | 11 | False | 0,116 |
| *WindowDiff* | 60 | 9 | Sim | 0,411 | 60 | 9 | Sim | 0,390 |
| Acurácia | 60 | 9 | Sim | 0,588 | 60 | 9 | Sim | 0,609 |
| Precisão | 40 | 9 | Sim | 0,645 | 20 | 11 | False | 0,720 |
| Revocação | 80 | 9 | Sim | 0,869 | 80 | 11 | Sim | 0,897 |
| $F^1$ | 80 | 9 | Sim | 0,638 | 80 | 11 | Sim | 0,655 |

Tabela 4 – Resultados obtidos com o *C99*

De acordo com os últimos testes, o algoritmo *C99* obteve melhor desempenho em acurácia, precisão, $F^1$, $P_k$ e *WindowDiff*, enquanto o *TextTiling* obteve o melhor desempenho em revocação como pode ser visto na Tabela 5.

O algoritmo *C99* obteve melhor desempenho em acurácia, precisão, $F^1$, $P_k$ e *WindowDiff*, enquanto o *TextTiling* obteve o melhor desempenho em revocação como pode ser visto na Tabela 5.

| **Algoritmo** | **Medida** | **Média** |
|---|---|---|
| *C99* | $P_k$ | 0,116 |
| *C99* | *WindowDiff* | 0,390 |
| *C99* | Acurácia | 0,609 |
| *C99* | Precisão | 0,720 |
| *C99* | $F^1$ | 0,655 |
| *TextTiling* | Revocação | 0,917 |

Tabela 5 – Melhores resultados obtidos.

Verificou-se que, de maneira geral, o algoritmo *C99* apresenta melhores resultados em relação ao *TextTiling*, contudo, testes estatísticos realizados indicaram que não houve diferença significativa entre os métodos. Nesse trabalho, escolheu-se o algoritmo *C99* por apresentar resultados satisfatórios e ligeira superioridade em relação ao *TextTiling*.

Após a identificação dos segmentos, o algoritmo retorna uma lista onde cada elemento é um texto com um assunto predominante e será a partir de disso considerado um documento.

### 3.1.3 Representação Computacional

As etapas anteriores produzem fragmentos de documentos onde o texto esta em um estágio de processamento inicial, com menos atributos que as versões originais, onde cada fragmento está associado a um tema, porém, ainda não estruturado. Ocorre que as técnicas de mineração de texto exigem uma representação estruturada dos textos.

Uma das formas mais comuns é a representação no formato matricial conhecida como Modelo Espaço Vetorial (*Vectorial Space Model* - VSM) (**??**), onde os documentos são representados como vetores em um espaço Euclidiano $t$-dimensional em que cada termo extraído da coleção é representado por um dimensão. Assim, cada componente de um vetor expressa a relação entre os documentos e as palavras. Essa estrutura é conhecida como *document-term matrix* ou matriz documento-termo.

A forma adotada nesse trabalho é a *Bag Of Words* a qual sintetiza a base de documentos em um contêiner de palavras, ignorando a ordem em que ocorrem, bem como pontuações e outros detalhes, preservando apenas o peso de determinada palavra nos documentos. É uma simplificação de toda diversidade de informações contidas na base de documentos sem o propósito de ser uma representação fiel do documento, mas oferecer a relação entre as palavras e os documentos a qual é suficiente para a maioria dos métodos de aprendizado de máquina.

### 3.1.4 Extração de Tópicos / Classificação de Textos

## 3.2 Módulo Consulta

Uma vez que a estrutura de dados interna contem os assuntos abordados na coleção de documentos, o tipo de ocorrência para cada assunto e o trecho onde se encontram, caberá ao módulo de consulta receber a *string* de consulta do usuário, resgatar os dados desejados e apresentá-los em ordem cronológica, dando condições para o usuário acessar os segmentos encontrados bem como os documentos originais.

### 3.2.1 Seleção dos tópicos

### 3.2.2 Visualização

O usuário final precisa de uma interface adequada para visualizar os resultados da busca considerando-se a relevância dos tópicos selecionados e a sequência cronológica.

Uma boa apresentação deve permitir ao usuário identificar a relevância os resultados e ser relativante independente para compreensão do conteúdo, evitando a leitura do texto completo. Ou seja, o texto de cada tópico apresentado deve ser suficiente para compreensão do assunto mencionado, sem necessidade de visualizar o documento original.

As informações apresentadas, incluem dados obtidos do documento como o nome do arquivo e data do original e o texto onde o assunto é mencionado. Além disso, apresenta-se as informação extraídas pelas técnicas de mineração de texto como os descritores e rótulos.

Para cada busca, é retornada uma lista de resultados ordenados pela relevância com a *string* de entrada, sendo cada item referente a uma menção a um assunto. Um tópico é abordado em diferentes momentos e registrado em atas distintas, onde cada menção é um resultado a ser apresentado.

Como parte da proposta, o sistema apresenta cada resultado dentro de um histórico de menções. Para isso, abaixo do texto é exibida uma linha com links para os resultados que compartilham o mesmo tópico ordenados por data. Os links, ao ser acionado, direciona para o resultado que aponta, além disso, quando o cursor do mouse está sobre o link, é apresentado um pre-visualização do texto. Dessa forma o usuário tem acesso uma interface que lhe fornece uma visão temporal das menções.

## 3.3 Estudo de caso

## 3.4 Avaliação

# Referências

BEEFERMAN, D.; BERGER, A.; LAFFERTY, J. Statistical models for text segmentation. *Machine Learning*, v. 34, n. 1, p. 177–210, 1999. ISSN 1573-0565. Disponível em: <http://dx.doi.org/10.1023/A:1007506220214>. Citado na página 10.

BLEI, D. M. Probabilistic topic models. *Commun. ACM*, ACM, New York, NY, USA, v. 55, n. 4, p. 77–84, abr. 2012. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2133806.2133826>. Citado 2 vezes nas páginas 7 e 14.

CHEN, H. et al. Modeling latent topics and temporal distance for story segmentation of broadcast news. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, v. 25, n. 1, p. 112–123, Jan 2017. ISSN 2329-9290. Citado na página 7.

CHENG, X. et al. Learning topics in short texts by non-negative matrix factorization on term correlation matrix. In: *SDM*. SIAM, 2013. p. 749–757. ISBN 978-1-61197-283-2. Disponível em: <http://dblp.uni-trier.de/db/conf/sdm/sdm2013.html#ChengGLWY13>. Citado na página 13.

CHOI, F. Y. Y. Advances in domain independent linear text segmentation. In: *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000. (NAACL 2000), p. 26–33. Disponível em: <http://dl.acm.org/citation.cfm?id=974305.974309>. Citado 3 vezes nas páginas 5, 8 e 17.

DEERWESTER, S. et al. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, v. 41, n. 6, p. 391–407, 1990. Citado na página 13.

FERRET, O. Improving text segmentation by combining endogenous and exogenous methods. In: *International Conference Recent Advances in Natural Language Processing, RANLP*. [S.l.: s.n.], 2009. p. 88–93. Citado na página 7.

HEARST, M. A. Multi-paragraph segmentation of expository text. In: *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994. (ACL '94), p. 9–16. Disponível em: <http://dx.doi.org/10.3115/981732.981734>. Citado 2 vezes nas páginas 8 e 17.

KERN, R.; GRANITZER, M. Efficient linear text segmentation based on information retrieval techniques. *Proceedings of the International Conference on Management of Emergent Digital EcoSystems, MEDES '09*, p. 167–171, 2009. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-74549147972&doi=10.1145%2f1643823.1643854&partnerID=40&md5=1c6f73bc0e07446fcc178440e48bbc40>. Citado 2 vezes nas páginas 8 e 9.

MISRA, H. et al. Text segmentation via topic modeling: An analytical study. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2009. (CIKM '09), p. 1553–1556. ISBN 978-1-60558-512-3. Disponível em: <http://doi.acm.org/10.1145/1645953.1646170>. Citado na página 8.

SAKAHARA, M.; OKADA, S.; NITTA, K. Domain-independent unsupervised text segmentation for data management. In: *2014 IEEE International Conference on Data Mining Workshop*. [S.l.: s.n.], 2014. p. 481–487. ISSN 2375-9232. Citado na página 7.