Ovídio José Francisco

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

Sorocaba, SP

20 de fevereiro de 2018

Lista de símbolos

$\bar{\oplus}$	operador NXOR, recebe dois argumentos lógicos e retorna verdadeiro se e somente se os argumentos forem iguais

Sumário

1 Introdução

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 (HOFMANN, 1999; DEERWESTER et al., 1990; LEE; SEUNG, 1999; BLEI, 2012).

2.1 Conceitos Básicos

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) (REZENDE, 2003), 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*

Nessa representação, cada termo é transformado em um atributo (*feature*) (REZENDE, 2003), em que 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. Essa representação é mostrada pela Tabela 1.

Essa forma de representação sintetiza a base de documentos em um contêiner de

	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*

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 (REZENDE, 2003).

2.2.2 Medidas de Proximidade

No modelo espaço vetorial, a similidade entre um documentos x e y é calculada pela correlação entre os vetores $\vec{x}$ e $\vec{y}$, a qual pode ser medida pelo cosseno do ângulo entre esses vetores. Dados dois documentos $x = (x_1, x_1, \ldots, x_t)$ e $y = (y_1, y_1, \ldots, y_t)$, calcula-se:

$$cosseno(x, y) = \frac{\vec{x} \bullet \vec{y}}{|\vec{x}| \times |\vec{y}|} = \frac{\sum_{i=1}^{t} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{t} x_i^2} \times \sqrt{\sum_{i=1}^{t} y_i^2}} \tag{2.1}$$

Valores de cosseno próximos a 0 indicam um ângulo próximo a 90º entre $\vec{x}$ e $\vec{y}$, ou seja, o documento x compartilha poucos termos com a consulta y, enquanto valores próximos a 1 indicam um ângulo próximo a 0º, ou seja, x e y compartilham termos e são similares (TAN; STEINBACH; KUMAR, 2005; FELDMAN; SANGER, 2006).

2.3 Recuperação de Informação

Devido à popularização dos computadores e à grande disponibilidade de documentos em formato digital, em especial na Web a área da Recuperação de Informação (RI) tem recebido atenção de pesquisadores nas últimas décadas. Recuperação de informação é área da computação que envolve a aplicação de métodos computacionais no tratamento e busca de informação em bases de dados não estruturados, usualmente grandes coleções de documentos textuais armazenados em dispositivos eletrônicos. De fato, não há dados completamente não estruturados ao se considerar a estrutura linguística latente em documentos textuais. O termo 'não estruturado' se refere a dados que oferecem uma estrutura claramente estruturada para sistemas computadorizados, a exemplo de documentos textuais (MANNING; RAGHAVAN; SCHüTZE, 2008).

A tarefa central da recuperação de informação é encontrar informações de interesse dos usuários e exibi-las. A principal ferramenta empregada nesse problema é o desenvolvimento de sistemas de recuperação de informação (SRI). Nesses sistemas o usuário expressa sua necessidade por meio da formulação de uma consulta, usualmente composta por um conjunto de palavras-chave. Então, o sistema apresenta os resultados da busca, frequentemente documentos, em ordem de relevância com a consulta.

2.3.1 Modelos de Recuperação de Informação

Um modelo de recuperação de informação deve criar representações de documentos e consultas a fim de predizer a necessidade expressa nos termos da consulta. Com base na entrada do usuário esses modelos buscam por documentos similares aos termos da consulta. Segue abaixo a descrição dos três modelos clássicos para recuperação de informação.

2.3.1.1 Modelo Booleano

O modelo booleano ou modelo lógico foi um dos primeiros modelos aplicados a recuperação informação sendo utilizado a partir de 1960. Nesse modelo uma consulta é considerada uma sequencia de termos conectador por operadores lógicos como AND, OR e NOT. Como resultado, classifica cada documento como relevante ou não relevante à consulta, sem gradação de relevância. Esse operadores lógicos poder ser manipulados por usuários com algum conhecimento em álgebra booleana para aumentar a quantidade de resultados ou restringi-la.

Esse modelo apresenta como principal desvantagem a impossibilidade de ordenação dos resultados por relevância, uma vez que para muitos sistemas de RI o *ranking* dos resultados é uma característica essencial, principalmente em grades bases de dados.

As vantagens desse modelo são a facilidade de implementação e a possibilidade de usuários experientes usarem os operadores lógicos como uma forma de controle sobre os resultados da busca. Por outro lado, para usuários inexperientes isso pode ser considerado uma desvantagem, uma vez que o uso expressões lógicas não é intuitivo. Apesar dos problemas apresentados, visto sua simplicidade, esse modelo foi largamente utilizado em sistemas comerciais.

2.3.1.2 Modelo Vetorial

Uma das formas mais comuns para representação textual é conhecida como Modelo Espaço Vetorial (*Vectorial Space Model* - VSM) (REZENDE, 2003), onde os documentos e consultas são representados como vetores em um espaço Euclidiano t-dimensional em que cada termo extraído da coleção é representado por uma dimensão. Considera-se que um documento pode ser representado pelo seu conjunto de termos, onde cada termo k_i de

um documento d_j associa-se um peso $w_{ij} \geq 0$ que indica a importância desse termo no documento. De forma similar, para uma consulta q, associa-se um peso $w_{i,q}$ ao par termo consulta que representa a similaridade entre a necessidade do usuário e o termo k_i. Assim o vetor associado ao documento d_j é dado por $\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j})$. De forma similar, o vetor associado a consulta q é dado por $\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q})$.

No modelo vetorial, a similidade entre um documento d_j e uma consulta q é calculada pela correlação entre os vetores $\vec{d_j}$ e $\vec{q}$, a qual pode ser medida pelo cosseno (Equação 2.1) do ângulo entre esses vetores, conforme já mostrado na Seção 2.2.2.

Avaliar a relevância de um documento sob uma consulta é fundamental para os modelos de RI. Para isso pode-se utilizar medidas estatísticas simples como a frequência do termo, conhecida como TF (do inglês *Term Frequency*) e a frequência de documentos, conhecida como DF (do inglês *Document Frequency*). A frequência do termo indica o número de vezes que um termo ocorre na coleção de documentos. A frequência de documentos, indica o número de documentos que contém ao menos uma ocorrência de um determinado termo. Considera-se que os termos que ocorrem frequentemente em muitos documentos, em geral, não trazem informações úteis para discriminar a relevância dos documentos, então, a fim de diminuir o peso de termos altamente frequentes, usa-se o fator IDF (*Inverted Document Frequency*), que é o inverso da número de documentos que contem um termo. O IDF é a medida de informação que um termo fornece com base em quão raro ou comum esse termo é para a coleção. Seja N o número de documentos de uma coleção e n_i o número de documentos onde o termo k_i ocorre, o cálculo de IDF é dado por:

$$IDF(k_i) = log\frac{N}{n_i} \tag{2.2}$$

Entre a medidas mais populares para ranqueamento de buscas está a TF-IDF (*Term Frequency-Inverted Document Frequency*) que pondera a frequência de um termo em um documento com sua frequência na coleção total de documentos. Assim, a relevância de um termo para um documento é dada por:

$$w_{i,j} = freq_{i,j} \cdot IDF(k_i) \tag{2.3}$$

Onde $freq_{i,j}$ é a frequência do termo k_i no documento d_j. A medida TF-IDF atribui valores altos para termos que ocorrem frequentemente em um documentos, e valores menores para termos que ocorrem poucas vezes em um documento ou em muitos documentos da coleção. A ideia da medida tf.idf e quantificar a importância de um termo em um documento com base em sua frequência no próprio documento e sua distribuição ao longo da coleção de documentos (CROFT; METZLER; STROHMAN, 2009; SALTON; BUCKLEY, 1988; SHAMSINEJADBABKI; SARAEE, 2012; SALTON; ALLAN, 1994).

Uma vez que o sistema calcula os graus de similaridade entre os documentos e a busca por meio da equação 2.1, é possível ranquear os resultados por ordem de relevância. Além disso, sua relativa simplicidade e flexibilidade, favorecem a aplicação desse modelo em sistemas de recuperação de informação (TAN; STEINBACH; KUMAR, 2005; CROFT; METZLER; STROHMAN, 2009; MANNING; RAGHAVAN; SCHüTZE, 2008).

2.3.1.3 Modelo Probabilístico

O modelo probabilístico é baseado no princípio da ordenação probabilística (*Probability Ranking Principle*) onde dada um consulta q e um documento d_j relevante a q, o modelo tenta estimar a probabilidade do usuário encontrar o documento d_j. O modelo assumente que para uma consulta q há um conjunto de documentos R_q que contém exatamente os documentos relevantes e nenhum outro, sendo este um conjunto resposta ideal que maximiza a probabilidade do usuário encontrar um documento d_j relevante a q.

Seja $\overline{R_q}$ o complemento de R de forma que $\overline{R_q}$ contém todos os documentos não relevantes à consulta q. Seja $P(R_q|d_j)$ a probabilidade do documento d_j ser relevante à consulta q e $P(\overline{R_q}|d_j)$ a probabilidade de d_j não ser relevante à q. A similaridade entre um documento d_j e uma consulta q é definida por:

$$sim(d_j, q) = \frac{P(R_q|dj)}{P(\overline{R_q}|dj)} \tag{2.4}$$

A fim de obter-se uma estimativa numéricas das probabilidades, o modelo assume o documento como uma combinação de palavras e seus pesos aos quais atribui-se valores binários que indicam a presença ou ausência de um termo, isto é, $w_{ij} \in \{0, 1\}$ e $w_{iq} \in \{0, 1\}$. Seja $p_i = P(k_i|R_q)$ a probabilidade do termo k_i ocorrer em um documento relevante à consulta q, e $s_i = P(k_i|\overline{R_q})$ a probabilidade do termo k_i estar presente em um documento não relevante. Então, pode-se calcular:

$$sim(d_j, q) = \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot \prod_{i:d_i=0} \frac{1 - p_i}{1 - s_i} \tag{2.5}$$

, onde $\prod_{i:d_i=1}$ significa o produto dos termos com valor 1.

O modelo também supõe que os termos ocorrem independentemente no documento, ou seja, a ocorrência de um termo não influencia a ocorrência de outro. Partindo dessas suposições, a Equação 2.5 passa por transformações que incluem aplicação da regra de Bayes e simplificações matemáticas, e chega-se a Equação 2.6 conhecida como equação de Robertson-Spark Jones a qual é considerada a expressão clássica para ranqueamento no modelo probabilístico. Detalhes da dedução dessa equação podem ser encontradas

em (CROFT; METZLER; STROHMAN, 2009; MANNING; RAGHAVAN; SCHüTZE, 2008; RIJSBERGEN, 1979).

$$sim(d_j, q) = \sum_{i=1}^{t} w_{i,j} \cdot w_{i,q} \cdot \sigma_{i/R} \tag{2.6}$$

, onde t é o número total de termos da coleção e

$$\sigma_{i/R} = \log \frac{p_i}{1 - p_i} + \log \frac{1 - s_i}{s_i} \tag{2.7}$$

Esse modelo tem com principal desvantagem a necessidade de estimar a separação inicial entre entre R_q e $\overline{R_q}$, pois o não se conhece inicialmente o conjunto dos documentos relevantes a uma consulta, o qual deve ser aprimorado por meio de interações com o usuário. Além disso, o modelo não leva em consideração a frequência dos termos na indexação do documento. Apresenta como vantagem a característica de atribuir probabilidades as similaridades entre documentos e consultas, o que permite ranquear dos resultados por ordem de relevância.

2.4 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.

Os primeiros trabalhos dessa área 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, e demonstrou-se que há uma estreita correlação entre quedas na coesão léxica em janelas de texto e a transição de assuntos (KOZIMA, 1993). Em seu trabalho, Kozima calculou a coesão léxica de uma janela de palavras usando *spreading activation* em uma rede semântica especialmente elaborada para o idioma Inglês. Contudo, a implementação de um algoritmo para outros domínios depende da construção de uma rede adequada.

O conceito de coesão léxica permite a aplicação da técnica de janelas deslizantes para encontrar os segmentos de um texto, em que se verifica a frequência dos termos em um fragmento do documento. Inicialmente, estabelece-se a partir do início do texto, um intervalo de t termos, chamado janela que em seguida é deslocada em passos de k termos adiante até o final do texto. A cada passo, analisa-se os termos contidos na janela.

O conceito de coesão léxica motivou a elaboração dos primeiros algoritmos para segmentação textual, entre eles o *TextTiling*. O *TextTiling* baseia-se na ideia de que um segmento pode ser identificado pela análise dos termos que o compõe. Inicialmente, o *TextTiling* recebe uma lista de candidatos a limite entre segmentos, usualmente finais de parágrafo ou finais de sentença. Utilizando a técnica de janelas deslizantes, para cada posição candidata são construídos 2 blocos, um contendo as 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. Esse processo é ilustrado na Figura 1.

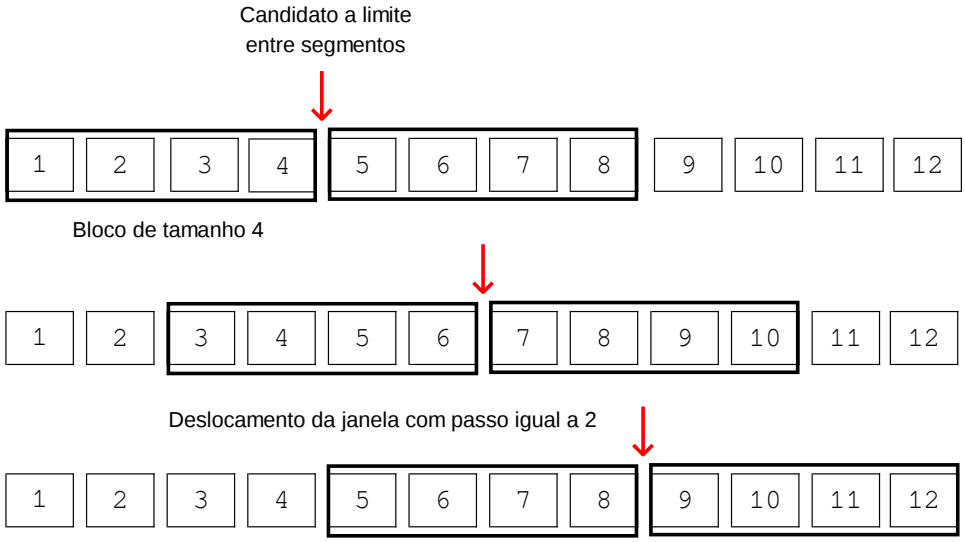

Figura 1 – Processo de deslocamento da janela deslizante. Os quadrados numerados representam as sentenças e os retângulos representam os blocos de texto a serem comparados. O deslocamento movimenta o candidato a limite e por consequência os blocos que o antecede e sucede.

Em seguida, os blocos de texto são representados por vetores que contém as frequências de suas palavras. Diferente da proposta de Kozima, o *TextTiling* utiliza cosseno (Equação 2.1) como medida para a similaridade entre os blocos adjacentes. Um limite ou transição entre segmentos é identificado sempre que a similaridade entre as unidades que antecedem e precedem o ponto candidato cai abaixo de um limiar, indicando uma diminuição da similaridade entre os blocos adjacentes. Ou seja, identifica-se uma transição entre segmentos pelos vales na curva de dissimilaridades. Para cada final de sentença representada por c_i atribui-se uma profundidade dada por $(c_{i-1} - c_i) + (c_{i+1} - c_i)$ e será um

limite entre segmentos caso a profundidade exceda $\overline{s} - \sigma$, onde $\overline{s}$ é a média da profundidade de todos os vales do documento e σ, o desvio padrão. Na Figura 2 é ilustrado a curva de dissimilaridade entre os blocos adjacentes.

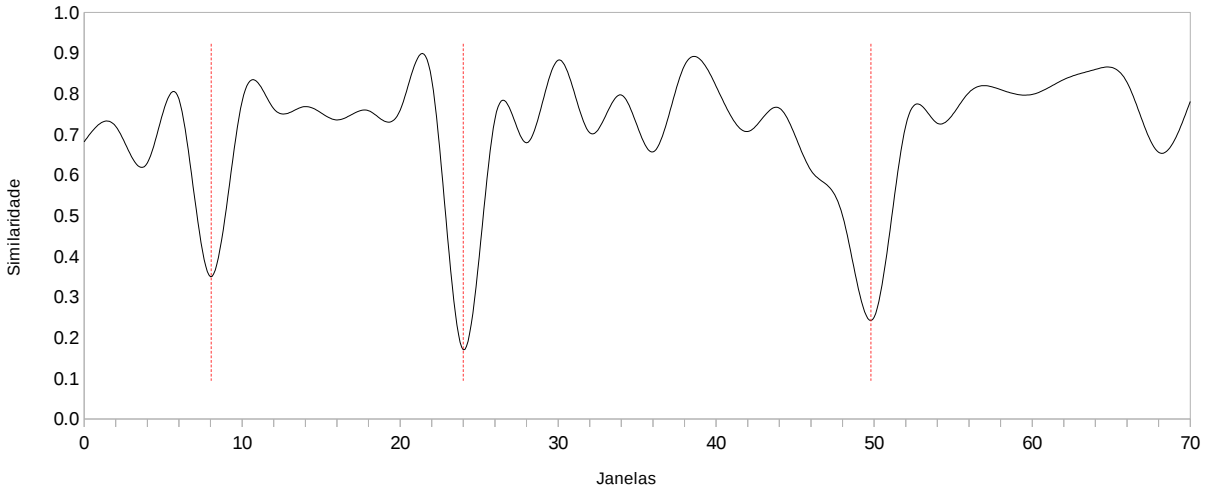

Figura 2 – Curva de dissimilaridades entre blocos de texto adjacentes. As linhas pontilhadas representam diminuições de similaridade que indicam limites entre segmentos.

O TextTiling apresenta como vantagens a facilidade de implementação e baixa complexidade computacional, favorecendo a implementação de trabalhos similares (NAILI; CHAIBI; GHEZALA, 2016; BOKAEI; SAMETI; LIU, 2015; CHAIBI; NAILI; SAMMOUD, 2014; KERN; GRANITZER, 2009; GALLEY et al., 2003), e sua utilização como base line em outros trabalhos (CARDOSO; PARDO; TABOADA, 2017; DIAS; ALVES; LOPES, 2007). 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 (CHOI, 2000) o qual é baseado em uma matriz de *ranking* das similaridades. A utilização de da coesão léxica pode não ser confiável para segmentos pequenos nessa abordagem, pois a ocorrência adicional de uma palavra pode causar certo impacto e alterar o cálculo da similaridade. Além disso, o estilo da escrita normalmente não é 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. Contorna-se esse problema fazendo uso de matrizes 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).

Na Figura 3 é mostrado um exemplo de uma matriz de similaridade onde a intensidade do ponto(i, j) representa a similaridade entre as sentenças i e j. Observa-se

que a matriz é simétrica, assim cada ponto na linha diagonal representa a similaridade quando $i = j$ (ou seja, com a mesma sentença) e revela quadrados com maior concentração de pontos ao longo da diagonal. A concentração de pontos ao longo da diagonal indica porções de texto com maior coesão léxica.

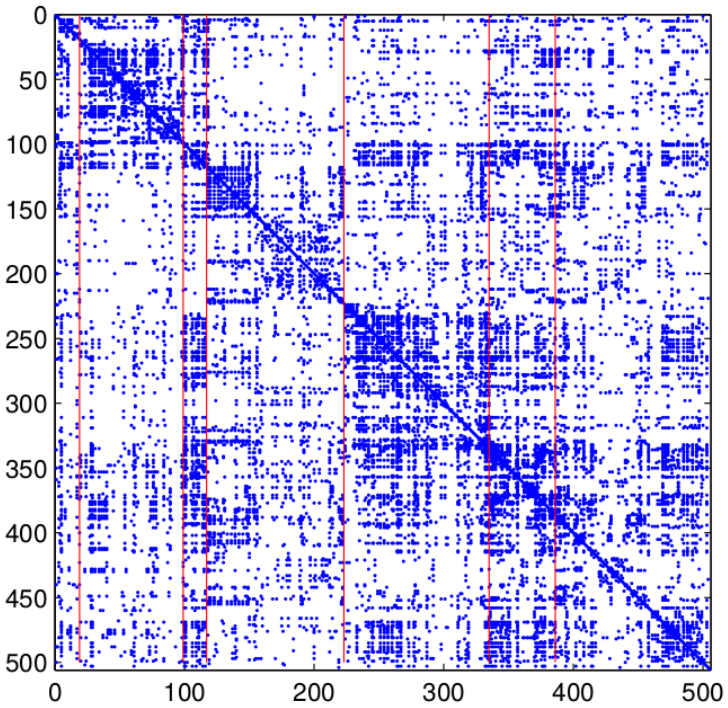

Figura 3 – *DotPlot* da similaridade entre sentenças onde as linha verticais representam segmentos reais (EISENSTEIN; BARZILAY, 2008).

Em seguida, cada valor na matriz de similaridade é substituído por seu *ranking* local. Para cada elemento da matriz, seu *ranking* será o número de elementos vizinhos com valor de similaridade menor que o seu. Assim, para cada elemento determina-se uma região quadrada de tamanho n em que o elemento em questão será comparado com $n \times n - 1$ elementos vizinhos. Na Figura 2.4 é destacado um quadro 3 x 3 de uma matriz. 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 2.4 para o valor $0,2$ a matriz de *rankings* conterá o valor 1 na mesma posição. Após a construção da matriz de ranking obtêm-se um maior contraste entre os pontos facilitando a detecção de limites quando a queda de similaridade entre sentenças é mais sutil.

Finalmente, com base na matriz de *ranking*, o C99 utiliza um método de *clustering* baseado no algoritmo *DotPloting* (REYNAR, 1998) que usa regiões com maior densidade em uma matriz de similaridades para determinar como os segmentos estão distribuídos. Um segmento é definido por duas sentenças i e j que representam uma região quadrada ao longo da diagonal da matriz. Calcula-se a densidade dessa região como mostrado na

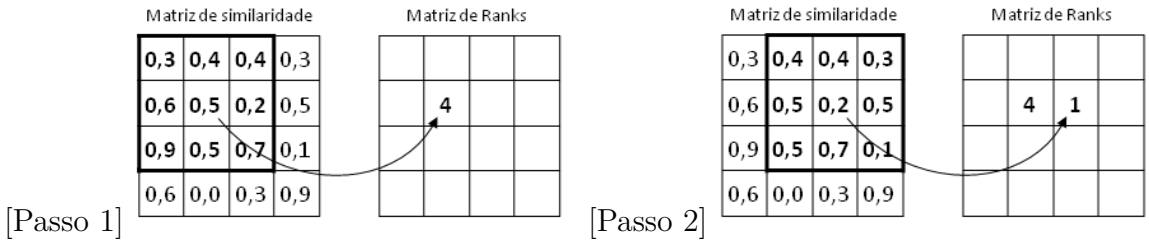

Figura 4 – Exemplo de construção de uma matriz de rankings.

Equação 2.8. Seja $s_{i,j}$ a somatória dos *rakings* de um segmento e $a_{i,j}$ sua área interior. Seja $B = \{b1, ..., b_m\}$ a lista de m segmentos e s_k e a_k são a somatória dos valores dos rankings e a área de um segmento k em B. Então, a densidade é computada por:

$$D = \frac{\sum_{k=1}^{m} s_k}{\sum_{k=1}^{m} a_k} \tag{2.8}$$

O processo inicia com um único segmento formado por todas as sentenças do documento e o divide recursivamente em m segmentos. Cada passo divide um dos segmentos em B no ponto (i, j) que maximiza D (Equação 2.8). O processo se repte até atingir o número de segmentos desejados ou um limiar de similaridade.

Desenvolveu-se também abordagens probabilísticas para segmentação textual, por exemplo, o método proposto por (UTIYAMA; ISAHARA, 2001) encontra a segmentação por meio de um modelo estatístico. Dado um texto representado por um conjunto de palavras $W = \{w_1, w_2, \ldots, w_n\}$ e um conjunto de segmentos $S = \{s_1, s_2, \ldots, s_m\}$ que segmenta W, a probabilidade da segmentação S é dada por:

$$P(S|W) = \frac{P(W|S)P(S)}{P(W)} \tag{2.9}$$

Com isso, é possível encontrar a sequência de segmentos mais provável $\hat{S} = argmax_S\ P(W|S)P(S)$. Nesse trabalho assume-se que os segmentos são estaticamente independentes entre si e as palavras nos segmentos são independentes dado o segmento que as contém. Essa simplificação permite decompor o termo $P(W|S)$ em um produtório de ocorrência de das palavras dado um segmento.

$$P(W|S) = \prod_{i=1}^{m} \prod_{j=1}^{n} P(w_j^i|S_i) \tag{2.10}$$

Onde $P(w_j^i|S_i)$ é a probabilidade da j-ésima palavra ocorrer no segmento S_i a qual é definida na Equação 2.11. Seja $f_i(w_j)$ a frequência da j-ésima palavra no i-ésimo segmento,

n_i é o número de palavras em S_i e k é o número de palavas diferentes em W. Calcula-se:

$$P(w_j^i|S_i) = \frac{f_i(w_j) + 1}{n_i + k} \tag{2.11}$$

A suposição de independência entre segmentos e as palavras neles contidas, são é verificada no mundo real. Para segmentos muito pequenos a estimativa das probabilidades das palavras pode ser afetada, além disso, o modelo não leva em conta a importância relativa das palavras (MALIOUTOV; BARZILAY, 2006).

Os métodos baseados em coesão léxica que utilizam métricas como cosseno quantificam a similaridade entre sentenças baseando-se apenas na frequência das palavras, Essa abordagem, ignora certas características do texto que podem dar pistas sobre a estrutura do texto. Por exemplo, frases como "Prosseguindo", "Dando continuidade", "Ao final da reunião"podem dar ajudar a detectar o inicio ou final de segmento. A fim de aproveitar esses indicadores, pode-se usar um framework bayesiano que permite incorporar fontes externas ao modelo. O método *BayesSeg* (EISENSTEIN; BARZILAY, 2008) aborda a coesão léxica em um contexto bayesiano onde as palavas de um segmento surgem de um modelo de linguagem multinomial o qual é associado a um assunto.

Essa abordagem é similar à métodos probabilísticos de extração de tópicos como o Latent Dirichlet Allocation (LDA) (BLEI; NG; JORDAN, 2003), com a diferença que ao invés de atribuir tópicos ocultos a cada palavra, esses são usados para segmentar o documento. Nesse sentido, detecta-se um limite entre sentenças quando a distribuição de tópicos entre elas for diferente. O *BayesSeg* baseia-se na ideia que alguns termos são usados em tópicos específicos enquanto outros são neutros em relação aos tópicos do documento e são usados para expressar uma estrutura do documento, ou seja, as "frases-pista"vem de um único modelo generativo. A fim de refletir essa ideia, o modelo é adaptado para influenciar a probabilidade da sentença de ser uma final ou início de segmento conforme a presença de "frases pista".

O *MinCutSeg* (MALIOUTOV; BARZILAY, 2006) aborda a segmentação textual como um problema de particionamento de grafo, em que cada nó representa um sentença e os pesos das arestas representam a similaridade entre duas sentenças (Figura 5). Nessa abordagem, a segmentação textual corresponde ao particionamento do grafo que representa o texto.

Essa abordagem é inspirada no trabalho de (SHI; MALIK, 2000) que propõe um critério para particionamento de grafos chamado *normalized-cut criterion* inicialmente desenvolvido para segmentação de imagens estáticas a qual foi aproveita a restrição de linearidade dos textos para segmentação textual.

Seja $G = V, E$ um grafo ponderado, unidimensional em que V é o conjunto de vértices que correspondem às sentenças e E é o conjunto de arestas que correspondem

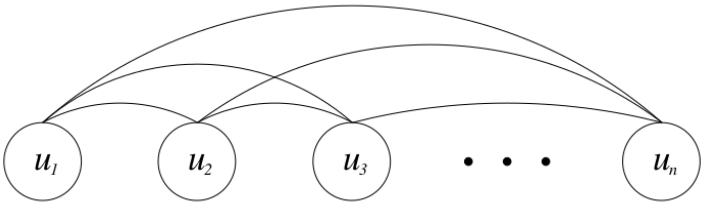

Figura 5 – Representação de texto baseada em grafo (MALIOUTOV; BARZILAY, 2006)

as similaridades entre as sentenças. Seja $w(u, v)$ o valor de similaridade entre o par de vértices u e v. O *MinCutSeg* visa particionar G em dois grafos disjuntos A e B de modo a minimizar o corte definido pela somatória das arestas que ligam u à v(Equação 2.12):

$$corte(A, B) = \sum_{u \in A, v \in B} w(u, v) \tag{2.12}$$

Além de maximizar a diferença entre as partições A e B, é necessário que essas seja homogenias em relação a similaridade de suas sentenças, conforme requerimento definido por (SHI; MALIK, 2000) em que o valor do corte deve ser normalizado pelo volume das partições dado por:

$$vol(A) = \sum_{u \in A, v \in V} w(u, v) \tag{2.13}$$

Em seguida, define-se o critério de corte normalizado (NCorte) como o resultado da normalização do corte pelo volume, conforme mostrado na Equação 2.14.

$$NCorte(A, B) = \frac{corte(A, B)}{vol(A} + \frac{corte(A, B)}{vol(B)} \tag{2.14}$$

Uma vez que um texto normalmente é dividido em mais que dois segmentos, é necessário extender o modelo para atender a essa necessidade. Seja A_{1k} uma partição e $V - A_k$ a diferença entre o grafo V e a partição k. O critério para múltiplos cortes normalizados é então estendido para:

$$NCorte_k(V) = \frac{corte(A_1, V - A_1)}{vol(A_1)} + \cdots + \frac{corte(A_k, V - A_k)}{vol(A_k)} \tag{2.15}$$

A decomposição do modelo em uma somatória de termos individuais permite empregar técnicas de programação dinâmica para o problema de cortes multidirecionais em grafos. Mais detalhes da formulação dessa solução estão disponíveis em (MALIOUTOV; BARZILAY, 2006).

Embora o problema minimizar cortes normalizados em grafos seja um problema do tipo NP-Completo[1], no contexto de segmentação textual esse problema é restrito a manter a linearidade dos vertices. A segmentação linear em um grafo implica que todos os vértices entre as extremidades esquerda e direitas de uma partição pertencem à essa partição, consequentemente o espaço de soluções possíveis é reduzido o que permite a execução do algoritmo em tempo polinomial.

2.4.1 Medidas de Avaliação em Segmentação Textual

As medidas de avaliação tradicionais como precisão e revocação permitem medir o desempenho de modelos de Recuperação de Informação e Aprendizado de Máquina por meio da comparação dos valores produzidos pelo modelo com os valores observados em uma referência. Usa-se uma tabela, chamada matriz de confusão, para visualizar o desempenho de um algoritmo. Na Tabela 2 é apresentada uma matriz de confusão para duas classes (Positivo e Negativo).

	Predição Positiva	Predição Negativa
Positivo real	VP (Verdadeiro Positivo)	FN (Falso Negativo)
Negativo real	FP (Falso Positivo)	VN (Verdadeiro Negativo)

Tabela 2 – Matriz de confusão.

No contexto de segmentação textual, um falso positivo é um limite identificado pelo algoritmo que não corresponde a nenhum limite na segmentação de referência, ou seja, o algoritmo indicou que em determinado ponto há uma quebra de segmento, mas na segmentação de referência, não há quebra no mesmo ponto. De maneira semelhante, um falso negativo é quando o algoritmo não identifica um limite existente na segmentação de referência, ou seja, em determinado ponto há, na segmentação de referência, um limite entre segmentos, contudo, o algoritmo não o identificou. Um verdadeiro positivo é um ponto no texto indicado pelo algoritmo e pela segmentação de referência como uma quebra de segmentos, ou seja, o algoritmo e a referência concordam que em determinado ponto há uma transição de assunto. Na avaliação de segmentadores, não há o conceito de verdadeiro negativo. Este seria um ponto no texto indicado pelo algoritmo e pela segmentação de referência onde não há uma quebra de segmentos. Uma vez que os algoritmos apenas indicam onde há um limite, essa medida não é necessária.

Nesse sentido, a precisão indica a proporção de limites corretamente identificados pelo algoritmo, ou seja, correspondem a um limite real na segmentação de referência. Porém, não diz nada sobre quantos limites reais existem. É calculada dividindo-se o

[1] NP-Completo configura um tipo de problema para o qual não se conhece uma solução determinística que possa ser computada em tempo polinomial. Papadimitriou provou que o problema de corte mínimo em grafos está incluso nessa categoria.

número de limites identificados automaticamente pelo número de candidatos a limite (Equação 2.16).

$$Precisão = \frac{VP}{VP + FP} \tag{2.16}$$

A revocação, é a proporção de limites verdadeiros que foram identificados pelo algoritmo. Porém não diz nada sobre quantos limites foram identificados incorretamente. É calculada dividindo-se o número de limites identificados automaticamente pelo número limites verdadeiros (Equação 2.17).

$$Revocação = \frac{VP}{VP + FN} \tag{2.17}$$

Existe uma relação inversa entre precisão e revocação. Conforme o algoritmo aponta mais segmentos no texto, este tende a melhorar a revocação e ao mesmo tempo, reduzir a precisão. Esse problema de avaliação pode ser contornado utilizado a medida F^1 que é a média harmônica entre precisão e revocação onde ambas tem o mesmo peso (Equação 2.18).

$$F^1 = \frac{2 \times Precisão \times Revocação}{Precisão + Revocação} \tag{2.18}$$

As medidas de avaliação tradicionais, 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 6 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.

Considerando o conceito de *near misses*, algumas medidas de avaliação foram propostas. Proposta por (BEEFERMAN; BERGER; LAFFERTY, 1999), P_k atribui valores parciais a *near misses*, ou seja, limites sempre receberão um peso proporcional à

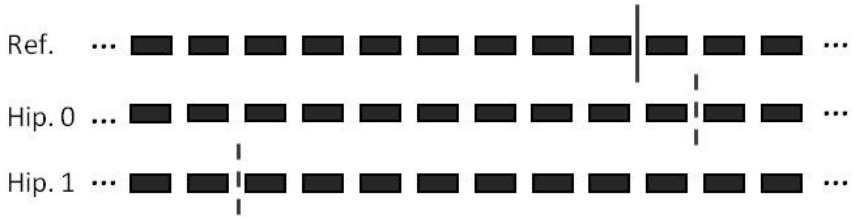

Figura 6 – Exemplos de *near missing* e falso positivo puro. Os blocos indicam uma unidade de informação e as linha verticais representam uma transição de assunto.

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 no algoritmo, se as extremidades (a primeira e última sentença) da janela estão ou não dentro do mesmo segmento, então, penaliza o algoritmo caso este não concorde com a referência. Ou seja, dado dois termos de distância k, P_k verifica se o algoritmo coloca os termos no mesmo segmento ou em segmentos distintos e o penaliza caso não concorde com a referência. Dadas uma segmentação de referência ref e uma segmentação automática hyp, ambas com N sentenças, P_k é computada como:

$$P_k(ref, hyp) = \frac{1}{N-k} \sum_{i=1}^{N-k} (\delta_{ref}(i, i+k) \bar{\oplus} \delta_{hyp}(i, i+k)) \tag{2.19}$$

onde $\delta_S(i, j)$ é a função indicadora que retorna 1 se as sentenças i e j estão no mesmo segmento e 0 caso contrário, $\bar{\oplus}$ é o operador `XNOR` (ambos ou nenhum) que retorna 1 se ambos os argumentos forem diferentes. O valor de k é calculado como a metade da média dos comprimentos dos segmentos reais. Como resultado, é retornada a dissimilaridade entre a segmentação calculada pela contagem de discrepâncias divida pela quantidade de segmentos analisados. Essa medida pode ser interpretada como a probabilidade de duas sentenças extraídas aleatoriamente pertencerem ao mesmo segmento.

WindowDiff (PEVZNER; HEARST, 2002) 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. Mais formalmente, para cada intervalo k, compara o número de segmentos obtidos pela referência r_i com o obtido pelo algoritmo a_i e penaliza o algoritmo se $r_i \neq a_i$. Na Equação 2.20 é mostrada a definição de WindowDiff onde $b(i, i+k)$ representa o número de limites entre as sentenças i e $i+k$ e N, o total de sentenças no texto.

$$WindowDiff(ref, hyp) = \frac{1}{N-k} \sum_{i=1}^{N-k} (|b(ref_i - ref_{i+k}) - b(hyp_i - hyp_{i+k})| > 0) \tag{2.20}$$

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.

3 Sistema Proposto

Essa seção apresenta as etapas de desenvolvimento do sistema de recuperação de atas proposto, 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 das atas. Em seguida, será relatado como as técnicas de mineração de texto e recuperação de informação são utilizadas nesse trabalho.

A Figura 7 mostra a visão geral do sistema proposto cujo objetivo é permitir ao usuário consultar uma coleção de documentos de reuniões a fim de obter todo o histórico de ocorrências de um determinado tema relacionado à pesquisa do usuário, podendo identificar nos documentos onde esse tema foi mencionado, bem como se 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, os quais serão detalhados nas próximas seções.

A Figura 7 mostra a visão geral do sistema com suas principais entradas e saídas.

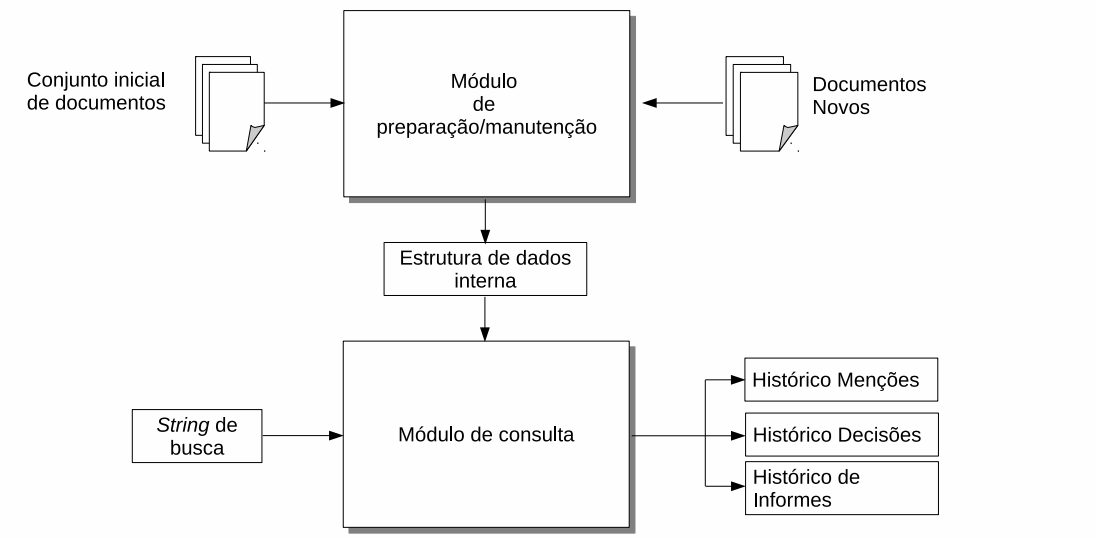

Figura 7 – Visão geral do sistema

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 separá-los em categorias por 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. A seguir são apresentadas as etapas do módulo de

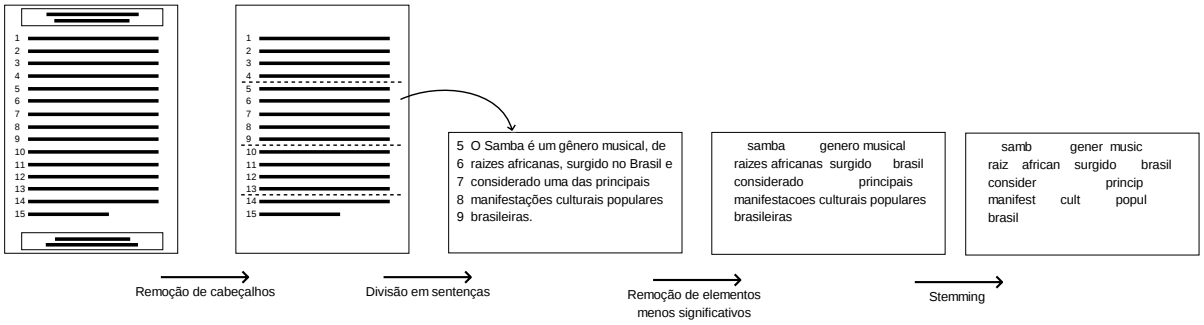

Figura 8 – Etapa de pré-processamento

preparação e manutenção desde a preparação dos documentos até a entrega da estrutura interna ao módulo de consulta.

3.1.1 Preparação dos documentos

As atas são normalmente armazenadas em arquivos binários do tipo *pdf*, *doc*, *docx* ou *odt*. As atas devem ser pré-processadas e estruturadas para que posam ser aplicados métodos de MI e RI. Inicialmente, o texto puro é extraído e passa por processos de transformação conforme apresentados a seguir.

A Figura 8 mostra a etapa de preparação de um documento em português que inclui a remoção de elementos menos significativos e a identificação de sentenças e segmentos.

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. Um cabeçalho é a porção de texto que inicia cada página do documento e, de forma semelhante, um rodapé e a porção que as encerra. Detecta-se os cabeçalhos e os rodapés sempre que há uma repetição das primeiras e últimas palavras do documento.

2. Identificação de finais sentenças: Ao considerar intuitivamente que uma sentença seja uma sequência de palavras entre sinais de pontuação como “.”, “!” e “?”, alguns erros poderiam ocorrer quando esses tiverem outra função dentro do texto como em abreviações, endereços de internet e datas. Outro problema seriam frases curtas com poucas palavras e que não expressam um conceito completo, mas parte dele. 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. No Algoritmo 1 é mostrado como cada *token* é identificado e marcado com final de sentença.

Algoritmo 1: Identificação de finais de sentença.

Entrada : Texto
Saída : Texto com identificações de finais de sentença

para todo *token, marcá-lo como final de sentença se:*
  Terminar com um `!`
  Terminar com um `.` e não for uma abreviação
  Terminar em `.?;` e:
    For seguido de uma quebra de parágrafo ou tabulação
    O próximo *token* iniciar com `({["'`
    O próximo *token* iniciar com letra maiúscula
    O penúltimo caracter for `)}]"'`
fim

3. Redução de termos: Removeu-se do as palavras que não contribuem para a distinção do texto em tópicos ou categorias, as quais são chamadas de *stop words*. Palavras como artigos, preposições, pronomes, verbos de estado[1]. Trata-se também como *stop words* as palavras de uso muito frequente dentro de um determinado domínio as quais não são capazes de discriminar documentos, portanto também não devem fazer parte dos atributos (REZENDE, 2003). Para removê-las, as letras foram convertidas em caixa baixa e usou-se uma lista de 438 palavras para identificá-las. 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, aplicou-se o algoritmo *Orengo* para remoção de sufixos (ALVARES; GARCIA; FERRAZ, 2005).

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 (BOKAEI; SAMETI; LIU, 2015; BOKAEI; SAMETI; LIU, 2016; MISRA et al., 2009; SAKAHARA; OKADA; NITTA, 2014).

Entre os principais trabalhos da literatura podemos citar o *TextTiling* (HEARST, 1994) e o *C99* (CHOI, 2000) são considerados um dos primeiros mais influentes sendo

[1] Apresentam uma situação inativa, onde o verbo não expressa uma alteração, mas apenas uma propriedade ou condição dos envolvidos.

utilizados com base lines em trabalhos recentes (CHAIBI; NAILI; SAMMOUD, 2014; NAILI; CHAIBI; GHEZALA, 2016; CARDOSO; PARDO; TABOADA, 2017)

3.1.1.2 Segmentação de Referência

A avaliação de um segmentador automático de textos exige 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.

Para isso, selecionou-se um conjunto de atas reais coletadas do Departamento de Computação da UFSCar campus Sorocaba. Analisou-se as atas públicas das reuniões do Conselho de Pós-Graduação e Conselho de Graduação desse departamento das quais foram selecionadas seis atas de cada conselho, sendo cinco referentes a reuniões ordinária e uma reunião extraordinária, totalizando doze documentos. Esses documentos foram escolhidos de forma que o conjunto final contenha atas com tamanhos diferentes (entre 1 e 4 páginas), e maior diversidade de conteúdo.

Em seguida, selecionou-se um grupo de anotadores para analisar e coletar dados referentes a segmentação de cada ata. O grupo de anotadores foi formado por profissionais com alguma afinidade com atas de reunião, como profissionais administrativos, professores e coordenadores de curso. Optou-se por utilizar um *software* como ferramenta para a coleta dos dados a fim de facilitar o trabalho de anotação e diminuir eventuais erros, conforme sugerido por (HOVY; LAVID, 2010). Essa ferramenta permitiu aos anotadores visualizar os documentos e indicar livremente as divisões entre segmentos, bem como rotulá-los em classes e indicar palavras que melhor descrevem o assunto central do segmento. Os anotadores receberam informações básicas sobre o objetivo da pesquisa e instruções de como operar o *software*. Contudo, nenhum critério foi estabelecido para o procedimento ficando os anotadores livres para segmentar e rotular as atas orientados apenas pela interface da ferramenta. Na Figura 9 é mostrada a interface da ferramenta utilizada para as anotações.

Após o processo de anotação, os dados coletados foram analisados para gerar as segmentações de referência. A segmentação de referência foi gerada utilizando o critério de maior concordância, como já relatado em outros trabalhos (HEARST, 1997; CARDOSO; PARDO; TABOADA, 2017; KAZANTSEVA; SZPAKOWICZ, 2012; PASSONNEAU; LITMAN, 1997; GALLEY et al., 2003). Considerou-se que ocorre um limite entre segmentos quando a maioria dos anotadores (metade mais um) concordaram que a mesma sentença é um final de segmento. A concordância entre os anotadores é uma medida importante que mostra como os anotadores compreendem os textos analisados e o nível de confiabilidade da segmentação de referência. Na Figura 10 é mostrado um exemplo de criação de uma segmentação de referência por meio da concordância entre anotadores. A primeiras

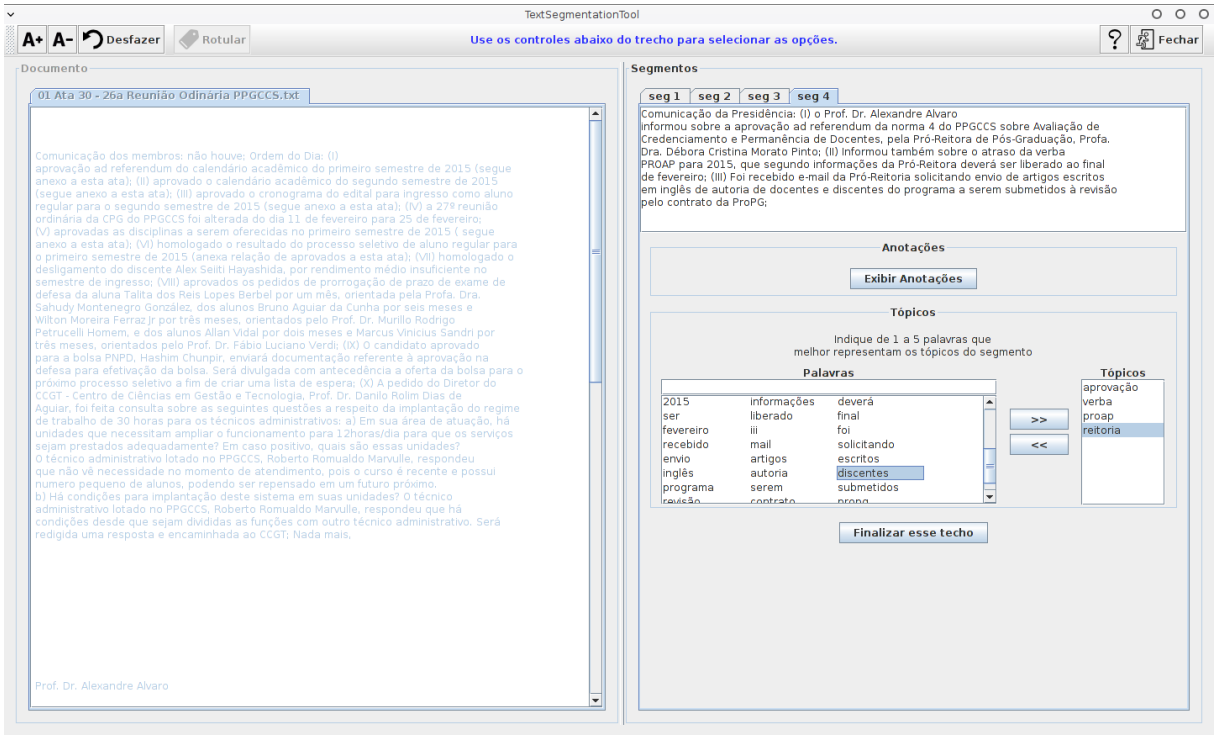

Figura 9 – Interface da ferramenta utilizada para anotações onde o texto a ser segmentado é exibido no painel a esquerda e os controles para anotação estão disponíveis a direita.

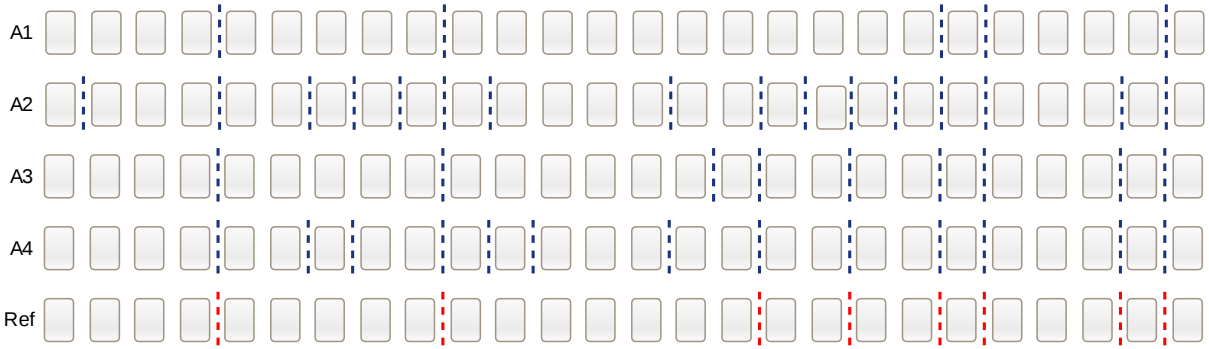

Figura 10 – Exemplo uma segmentação de referência criada a partir da concordância entre segmentações manuais.

linhas representam segmentações fornecidas por anotadores e a última linha representa a segmentação resultante da concordânica entre a maioria dos segmentadores e portanto mais confiável.

Para mensurar a concordância entre anotadores, a medida kappa (k) (CARLETTA, 1996) é frequentemente utilizada (GRUENSTEIN; NIEKRASZ; PURVER, 2007; CARDOSO; PARDO; TABOADA, 2017; HEARST, 1997). Essa medida retorna um valor

no intervalo de 0 até 1, onde 1 significa uma concordância perfeita e 0 que não houve concordância. Embora (CARLETTA, 1996) afirme que valores de k > 0.8 indicam que os dados são confiáveis, visto a subjetividade da tarefa de segmentação textual, medidas menores podem ser aceitaveis, como reportado em (HEARST, 1997) que alcançou k = 0,64 e (CARDOSO; PARDO; TABOADA, 2017), k = 0,56.

A Tabela 3 contém, para cada ata, a quantidade de sentenças e a quantidade de segmentos identificadas pelos participantes.

Ata	**Sentenças**	**A1**	**A2**	**A3**	**A4**
Ata 1	25	7	15	8	16
Ata 2	17	4	14	6	14
Ata 3	26	6	17	10	14
Ata 4	26	5	12	7	12
Ata 5	33	4	17	9	16
Ata 6	11	3	9	4	5
Ata 7	20	3	5	4	0
Ata 8	35	4	5	9	0
Ata 9	24	3	3	9	0
Ata 10	50	4	5	8	0
Ata 11	43	4	5	12	0
Ata 12	56	3	4	11	0

Tabela 3 – Quantidade de sentenças e segmentos de referência por ata.

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.

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 4 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 4 – Resultados obtidos com o *TextTiling*

Uma vez que a coesão léxica é pressuposto de muitas abordagens em segmentação textual, fez-se uma análise desses documentos quanto a similaridade dos termos ao longo do

texto. Verificou-se que a técnica de janelas deslizantes empregada pelo TextTiling encontra os vales que indicam transições entre segmentos, contudo ao comparar esses vales com a segmentação de referência, nota-se que a maioria dos limites coincide ou estão próximos aos vales, porém há casos onde a referência indica limites em trechos com alta coesão léxica e outros onde a queda da coesão, indicada por vales, não coincide com nenhum limite de referência.

Na Figura 11 é apresentado a variação da coesão léxica ao longo de uma ata e a segmentação obtida pelo *TextTiling* usando tamanho de janela igual a 50 e passo 9. A linha horizontal representa a variação da coesão léxica e as linha verticais azuis e vermelhas representam os limites entre segmentos atribuidos pela referência e pelo algoritmo respectivamente.

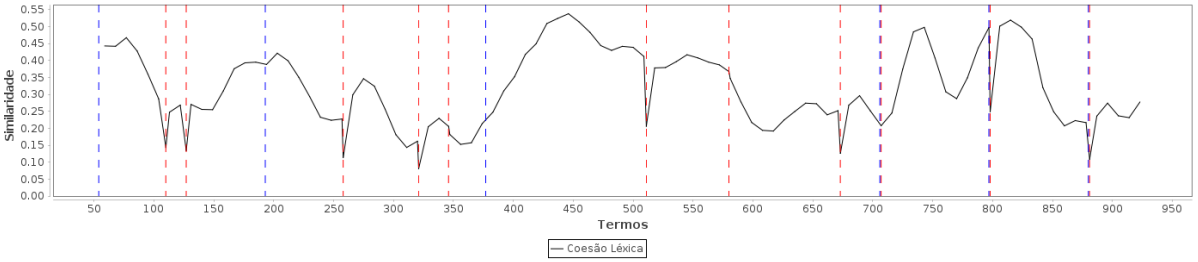

Figura 11 – Variação da coesão léxica ao longo de uma ata junto a uma segmentação automática em contraste com uma segmentação de referência.

Na Tabela 5 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 5 – Resultados obtidos com o *C99*

Verificou-se que o *C99* obteve melhor desempenho em acurácia, precisão, F^1, P_k e *WindowDiff*, em relação ao *TextTiling*, enquanto este obteve o melhor desempenho em revocação. 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.

A avaliação final foi feita pela comparação dos algoritmos usando as medidas *Pk* e *WindowDiff*. É apresentada também, para fins de comparação, as medidas tradicionais acurácia, precisão, revocação e F[1], entretanto, nesse contexto, essas medidas são menos significativa que Pk e WindowDiff, conforme já mencionado na Seção **??**. A Tabela 6 contém as médias com cada algoritmo. Vale lembrar que P_k e *WindowDiff* são medidas de dissimilaridade, ou seja, os valores menores significam melhores resultados.

Método	**Pk**	**WD**	**A**	**P**	**R**	**F1**	**Segmentos**
Sentenças	0.320	0.502	0.498	0.498	**1.000**	**0.642**	22.083
TextTiling	0.275	0.469	0.531	0.514	0.937	0.640	19.583
C99	0.142	0.426	0.574	0.601	0.473	0.506	8.167
BayesSeg	0.148	0.414	0.586	0.599	0.526	0.528	8.750
MinCut	0.226	0.532	0.468	0.464	0.438	0.432	10.333
TextSeg	**0.085**	**0.387**	**0.613**	**0.714**	0.412	0.497	5.167

Tabela 6 – Melhores resultados obtidos.

Na Figura 12 é apresentada a performance dos algoritmos nas medidas tradicionais. Observa-se valores altos de revocação para a segmentação por sentenças, pois é atribuído um limite a todo candidato a final de segmento, o que resulta no valor máximo para revocação. De maneira semelhante, o comportamento do *TextTiling* gera mais segmentos em relação aos demais, e com isso tem-se valores maiores de revocação, o que pode ser contornado configurando o algoritmo com passos maiores, ou ainda, sobre-escrevendo a função que calcula os *depth scores* para reconhecer vales mais largos.

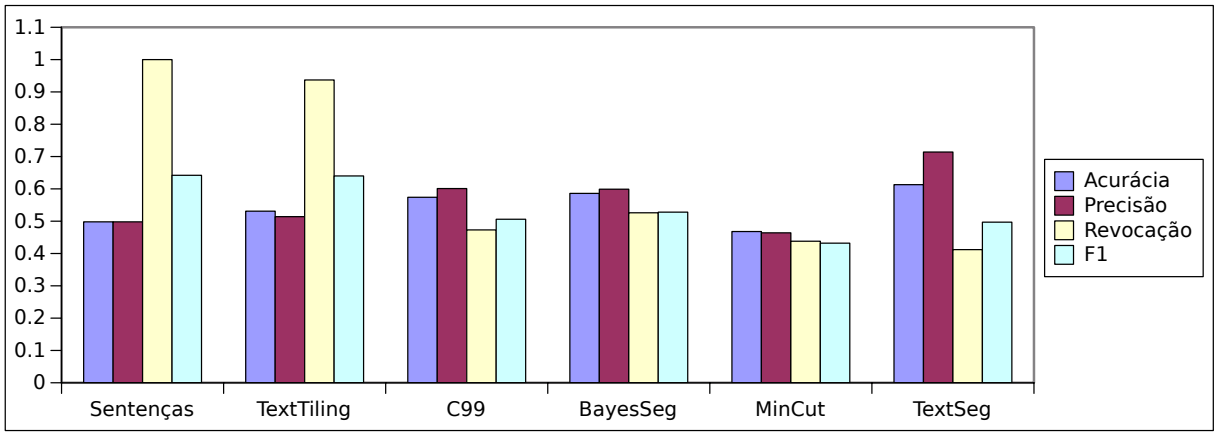

Figura 12 – Performance dos algoritmos de segmentação textual com as medidas tradicionais

Na Figura 13 é apresentada a performance dos algoritmos nas medidas P_k e *WindowDiff*. Verifica-se que *TextSeg* apresenta valores de *WindowDiff* próximas ao *C99* e *BayesSeg* e resultados mais significantes quando medidos por P_k em relação aos demais algoritmos.

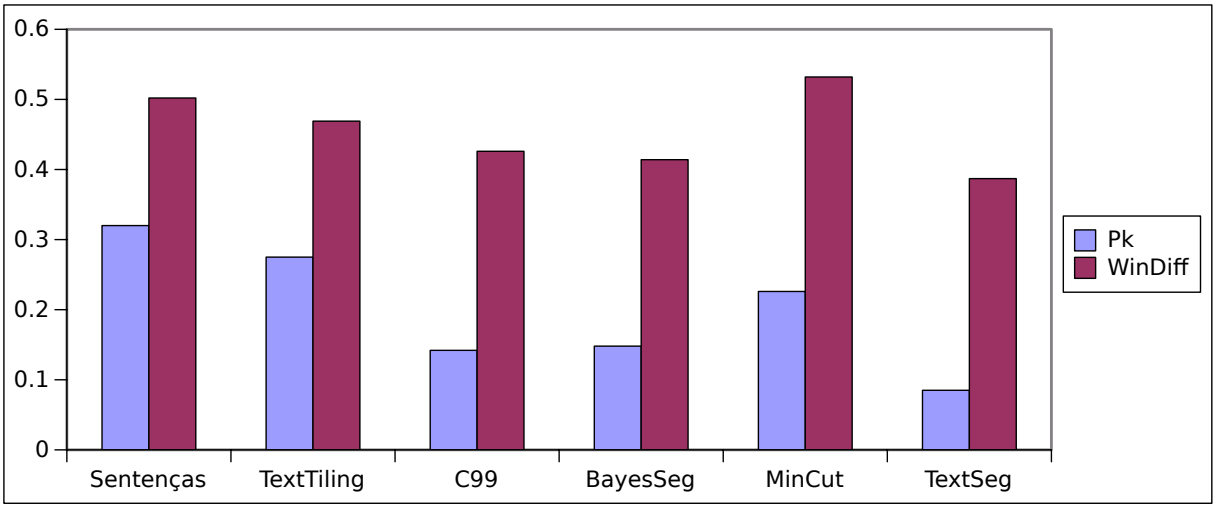

Figura 13 – Performance dos algoritmos de segmentação textual com as medidas P_k e *WindowDiff*.

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) (REZENDE, 2003), 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. Nesse trabalho a representação empregada 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.

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 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 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.

Referências

ALVARES, R. V.; GARCIA, A. C. B.; FERRAZ, I. Stembr: A stemming algorithm for the brazilian portuguese language. In: *Proceedings of the 12th Portuguese Conference on Progress in Artificial Intelligence*. Berlin, Heidelberg: Springer-Verlag, 2005. (EPIA'05), p. 693–701. ISBN 3-540-30737-0, 978-3-540-30737-2. Disponível em: <http://dx.doi.org/10.1007/11595014_67>. Citado na página 25.

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 20.

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 na página 7.

BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. *J. Mach. Learn. Res.*, JMLR.org, v. 3, p. 993–1022, mar. 2003. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=944919.944937>. Citado na página 17.

BOKAEI, M. H.; SAMETI, H.; LIU, Y. Linear discourse segmentation of multi-party meetings based on local and global information. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, IEEE Press, Piscataway, NJ, USA, v. 23, n. 11, p. 1879–1891, nov. 2015. ISSN 2329-9290. Disponível em: <http://dx.doi.org/10.1109/TASLP.2015.2456430>. Citado 2 vezes nas páginas 14 e 25.

BOKAEI, M. H.; SAMETI, H.; LIU, Y. Extractive summarization of multiparty meetings through discourse segmentation. *Natural Language Engineering*, Cambridge University Press, v. 22, n. 1, p. 41–72, 2016. Citado na página 25.

CARDOSO, P.; PARDO, T.; TABOADA, M. Subtopic annotation and automatic segmentation for news texts in brazilian portuguese. *Corpora*, Edinburgh University Press, v. 12, n. 1, p. 23–54, 2017. Citado 4 vezes nas páginas 14, 26, 27 e 28.

CARLETTA, J. Assessing agreement on classification tasks: The kappa statistic. *Comput. Linguist.*, MIT Press, Cambridge, MA, USA, v. 22, n. 2, p. 249–254, jun. 1996. ISSN 0891-2017. Disponível em: <http://dl.acm.org/citation.cfm?id=230386.230390>. Citado 2 vezes nas páginas 27 e 28.

CHAIBI, A. H.; NAILI, M.; SAMMOUD, S. Topic segmentation for textual document written in arabic language. *Procedia Computer Science*, v. 35, p. 437 – 446, 2014. ISSN 1877-0509. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050914010898>. Citado 2 vezes nas páginas 14 e 26.

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 2 vezes nas páginas 14 e 25.

CROFT, B.; METZLER, D.; STROHMAN, T. *Search Engines: Information Retrieval in Practice*. 1st. ed. USA: Addison-Wesley Publishing Company, 2009. ISBN 0136072240, 9780136072249. Citado 3 vezes nas páginas 10, 11 e 12.

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 7.

DIAS, G.; ALVES, E.; LOPES, J. G. P. Topic segmentation algorithms for text summarization and passage retrieval: An exhaustive evaluation. In: *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*. AAAI Press, 2007. (AAAI'07), p. 1334–1339. ISBN 978-1-57735-323-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1619797.1619859>. Citado na página 14.

EISENSTEIN, J.; BARZILAY, R. Bayesian unsupervised topic segmentation. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008. (EMNLP '08), p. 334–343. Disponível em: <http://dl.acm.org/citation.cfm?id=1613715.1613760>. Citado 2 vezes nas páginas 15 e 17.

FELDMAN, R.; SANGER, J. *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. New York, NY, USA: Cambridge University Press, 2006. ISBN 0521836573, 9780521836579. Citado na página 8.

GALLEY, M. et al. Discourse segmentation of multi-party conversation. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003. (ACL '03), p. 562–569. Disponível em: <http://dx.doi.org/10.3115/1075096.1075167>. Citado 2 vezes nas páginas 14 e 26.

GRUENSTEIN, A.; NIEKRASZ, J.; PURVER, M. *MEETING STRUCTURE ANNOTATION – Annotations Collected with a General Purpose Toolkit*. [S.l.]: Springer, Dordrecht, 2007. Citado na página 27.

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 na página 25.

HEARST, M. A. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, MIT Press, Cambridge, MA, USA, v. 23, n. 1, p. 33–64, mar. 1997. ISSN 0891-2017. Disponível em: <http://dl.acm.org/citation.cfm?id=972684.972687>. Citado 3 vezes nas páginas 26, 27 e 28.

HOFMANN, T. Probabilistic latent semantic indexing. In: *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 1999. (SIGIR '99), p. 50–57. ISBN 1-58113-096-1. Disponível em: <http://doi.acm.org/10.1145/312624.312649>. Citado na página 7.

HOVY, E.; LAVID, J. Towards a 'science' of corpus annotation: A new methodological challenge for corpus linguistics. v. 22, p. 13–36, 01 2010. Citado na página 26.

KAZANTSEVA, A.; SZPAKOWICZ, S. Topical segmentation: A study of human performance and a new measure of quality. In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012. (NAACL HLT '12), p. 211–220. ISBN 978-1-937284-20-6. Disponível em: <http://dl.acm.org/citation.cfm?id=2382029.2382058>. Citado na página 26.

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 14 e 20.

KOZIMA, H. Text segmentation based on similarity between words. In: *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 1993. (ACL '93), p. 286–288. Disponível em: <http://dx.doi.org/10.3115/981574.981616>. Citado na página 12.

LEE, D.; SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. v. 401, p. 788–91, 11 1999. Citado na página 7.

MALIOUTOV, I.; BARZILAY, R. Minimum cut model for spoken lecture segmentation. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006. (ACL-44), p. 25–32. Disponível em: <https://doi.org/10.3115/1220175.1220179>. Citado 2 vezes nas páginas 17 e 18.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN 0521865719, 9780521865715. Citado 3 vezes nas páginas 8, 11 e 12.

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 2 vezes nas páginas 14 e 25.

NAILI, M.; CHAIBI, A. H.; GHEZALA, H. H. B. Exogenous approach to improve topic segmentation. *International Journal of Intelligent Computing and Cybernetics*, v. 9, n. 2, p. 165–178, 2016. Disponível em: <https://doi.org/10.1108/IJICC-01-2016-0001>. Citado 2 vezes nas páginas 14 e 26.

PASSONNEAU, R. J.; LITMAN, D. J. Discourse segmentation by human and automated means. *Comput. Linguist.*, MIT Press, Cambridge, MA, USA, v. 23, n. 1, p. 103–139, mar. 1997. ISSN 0891-2017. Disponível em: <http://dl.acm.org/citation.cfm?id=972684.972689>. Citado na página 26.

PEVZNER, L.; HEARST, M. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, v. 28, n. 1, p. 19–36, 2002. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-0037870455&doi=10.1162%2f089120102317341756&partnerID=40&md5=279abc4e76fcfc2c4a1896e76a245034>. Citado na página 21.

REYNAR, J. C. *Topic Segmentation: Algorithms and Applications*. Tese (Doutorado), Philadelphia, PA, USA, 1998. AAI9829978. Citado na página 15.

REZENDE, S. O. *Sistemas Inteligentes*. Barueri, SP: Manole, 2003. 337 - 270 p. Citado 5 vezes nas páginas 7, 8, 9, 25 e 32.

RIJSBERGEN, C. J. V. *Information Retrieval*. 2nd. ed. Newton, MA, USA: Butterworth-Heinemann, 1979. ISBN 0408709294. Citado na página 12.

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 25.

SALTON, G.; ALLAN, J. Automatic text decomposition and structuring. In: *Intelligent Multimedia Information Retrieval Systems and Management - Volume 1*. Paris, France, France: LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 1994. (RIAO '94), p. 6–20. Disponível em: <http://dl.acm.org/citation.cfm?id=2856823.2856826>. Citado na página 10.

SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 24, n. 5, p. 513–523, ago. 1988. ISSN 0306-4573. Disponível em: <http://dx.doi.org/10.1016/0306-4573(88)90021-0>. Citado na página 10.

SHAMSINEJADBABKI, P.; SARAEE, M. A new unsupervised feature selection method for text clustering based on genetic algorithms. *J. Intell. Inf. Syst.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 38, n. 3, p. 669–684, jun. 2012. ISSN 0925-9902. Disponível em: <http://dx.doi.org/10.1007/s10844-011-0172-5>. Citado na página 10.

SHI, J.; MALIK, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 8, p. 888–905, Aug 2000. ISSN 0162-8828. Citado 2 vezes nas páginas 17 e 18.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367. Citado 2 vezes nas páginas 8 e 11.

UTIYAMA, M.; ISAHARA, H. A statistical model for domain-independent text segmentation. In: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2001. (ACL '01), p. 499–506. Disponível em: <https://doi.org/10.3115/1073012.1073076>. Citado na página 16.