Segmentação Textual Automática de Atas de Reunião

1

Resumo. *A tarefa de segmentação textual consiste em dividir um texto em porções com significado relativamente independente, de maneira que cada segmento contenha um assunto. Muitas pesquisas avaliam técnicas de segmentação textual usando textos longos com a concatenação de notícias de vários assuntos ou com a transcrição de discursos ou reuniões com múltiplos participantes. Este artigo concentra-se nos principais algoritmos da literatura aplicando-os ao contexto das atas de reunião em português, as quais as além do idioma menos estudado com essas técnicas, possuem estilo de redação mais formal e sucinto. Ao final, chegou-se a uma técnica capaz de segmentar as atas com performance similar a outros trabalhos da literatura.*

Abstract. *The abstract goes here!*

1. Introdução

Reuniões são tarefas presentes em atividades corporativas, ambientes de gestão e nas organizações de um modo geral. O conteúdo das reuniões é frequentemente registrado em texto na forma de atas para fins de documentação e posterior consulta. A organização e consulta manual desses arquivos torna-se uma tarefa custosa, especialmente considerando o seu crescimento em uma instituição [Lee et al. 2011, TAKAHASHI et al. 2013, Schwartz-Ziv and Weisbach 2013].

As atas são documentos textuais e portanto constituem dados não estruturados. Assim, um sistema que responde a consultas do usuário ao conteúdo das atas, retornando trechos de textos relevantes à sua intenção, é um desafio que envolve a compreensão de seu conteúdo [Bokaei et al. 2015].

Devido a fatores como a não estruturação e volume dos textos, a localização de assunto é uma tarefa custosa. Usualmente, o que se faz são buscas manuais guiadas pela memória ou com uso de ferramentas computacionais baseadas em localização de palavras-chave.

Buscas por palavras-chave normalmente exigem a inserção de de termos exatos que devem necessariamente estar contidos no trecho onde está o assunto. Localizadas as palavras-chave as ferramentas desse tipo apresentam ao usuário um documento com as palavras buscadas em destaque, mantendo o texto longo. Isso dificulta por exemplo o ranqueamento por relevância.

Uma vez que a ata registra a sucessão de assuntos discutidos na reunião, há interesse em um sistema que aponte trechos de uma ata que tratam de um assunto específico. Tal sistema tem duas principais tarefas: descobrir quando há uma mudança de assunto e quais são esses assuntos. Este trabalho tem foco principal na na primeira tarefa, detecção de mudança de assuntos, que pode ser atendida pela segmentação automática de textos [Banerjee and Rudnicky 2006, Beeferman et al. 1999, Reynar 1998].

A tarefa de segmentação textual consiste dividir um texto em partes que contenham um significado relativamente independente. Em outras palavras, é identificar as posições onde há uma mudança significativa de assuntos.

A segmentação de textos 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.

O interesse por segmentação textual tem crescido em em aplicações voltadas a recuperação de informação [Reynar 1999] e sumarização de textos. Essa técnica pode ser usada para aprimorar o acesso a informação quando essa é 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. Além disso, encontrar pontos onde o texto muda de assunto, pode ser útil como etapa de preprocessamento em aplicações voltadas ao entendimento do texto, principalmente em textos longos. 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 [Choi 2000]. A sumarização de texto pode ser aprimorada ao processar segmentos separados por tópicos ao invés de documentos inteiros [Boguraev and Neff 2000a, Boguraev and Neff 2000b, Dias et al. 2007].

Frequentemente atas de reunião têm a característica de apresentar um texto com poucas quebras de parágrafo e sem marcações de estrutura, como capítulos, seções ou quaisquer indicações sobre o tema do texto. As atas de reunião diferem dos textos comumente estudados em outros trabalhos em alguns pontos. O estilo contribui para a escrita de textos sucintos com poucos detalhes, pois o ambiente dá preferência a textos curtos. Segundo Choi [Choi et al. 2001], o segmentador tem a acurácia reduzida em segmentos curtos (em torno de 3 a 5 sentenças).

Há também um maior foco no idioma inglês, presente na maioria dos artigos publicados. Embora haja abordagens voltadas para outros idiomas, falta ainda uma maior atenção na literatura sobre língua portuguesa e a documentos com características próprias como as atas de reunião. Diferenças de performance podem ser vistas no mesmo algoritmo quando aplicado em documentos de diferentes idiomas, onde a aplicação em textos em inglês apresenta um taxa de erro significativamente menor que outro como o alemão e o espanhol [Kern and Granitzer 2009, Sitbon and Bellot 2004]. Assim, esse trabalho trata da aplicação e avaliação de algoritmos tradicionais ao contexto de documentos em português do Brasil, com ênfase especial nas atas de reuniões.

O restante deste artigo está dividido da seguinte forma. Na Seção 2 o processo de segmentação textual é melhor descrito, bem como os principais algoritmos da literatura. Na Seção 3 são apresentados trabalhos recentes que desenvolveram as técnicas existentes em contextos específicos. Na Seção 4 é apresentada a proposta desse trabalho voltada às atas de reunião. Na Seção 5 é detalhado a avaliação dos experimentos e seus resultados são reportados. Por fim, na Seção 6 são apresentadas as considerações finais e trabalhos futuros.

2. Referencial Teórico

Um documento textual, sobretudo quando longo, é frequentemente uma sucessão de assuntos. A segmentação textual ou segmentação topical é a tarefa de dividir um texto mantendo em cada parte um tópico com seu significado completo.

O texto deve ser previamente dividido em unidades de informação, que podem ser palavras, parágrafos e mais usualmente em sentenças. Essas unidades são processadas pelos algoritmos como a menor parte de um segmento, ou seja uma unidade não pode divida, mas deve pertencer integralmente a um segmento. Por exemplo, algoritmos baseados em janelas deslizantes avançam uma unidade a cada passo. Da mesma forma, algoritmos baseados em matrizes de similaridade, devem calcular a similaridade entre essas unidades. Dessa forma, um segmento pode ser visto como uma sucessão de unidades de informação que compartilham o mesmo assunto.

Usualmente, utiliza-se sentenças como unidades informação, ou seja, as janelas deslizam uma sentença a cada passo e matrizes de similaridades armazenam as similaridades entre as sentenças do texto. Assim, cada ponto entre duas sentenças e considerado um candidato a limite entre segmentos.

Trabalhos anteriores se apoiam na ideia de que a mudança de tópicos 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 e delimitado pela análise das palavras que o compõe [Galley et al. 2003, Boguraev and Neff 2000a].

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 1, onde $f_{x,j}$ é a frequência da palavra j na sentença x e $f_{y,j}$ é a frequência da palavra j na sentença y.

$$Sim(x, y) = \frac{\Sigma_j f_{x,j} \times f_{y,j}}{\sqrt{\Sigma_j f_{x,j}^2 \times \Sigma f_{y,j}^2}} \quad (1)$$

2.1. Principais algoritmos

Entre os principais trabalhos 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 de segmento é identificado sempre que a similaridade 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 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 apresentam acurácia relativamente superior como apresentado em [Choi 2000, Kern and Granitzer 2009, Misra et al. 2009].

O C99 é um algoritmo basedo 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 causa um impacto que pode alterar significativamente o cálculo da similaridade [Choi 2000]. Além disso, o estilo da escrita pode não ser constante em todo o texto. Choi sugere que, 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, o autor contorna esse problema fazendo uso de *rankings* de similaridade para encontrar os segmentos de texto.

Inicialmente é construída uma matriz que contém as similaridades de todas as unidades de texto. 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, para cada elemento delimita-se uma região da matriz, denominada máscara, para comparar com seus vizinhos. Na Figura 1 é apresentado um quadro de dimensões 3 x 3 destacado na matriz de similaridades, onde cada elemento da matrix é a similaridade entre duas unidades de informação. Tomando como exemplo o elemento com valor $0,5$, a mesma posição na matriz de *ranks* 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.

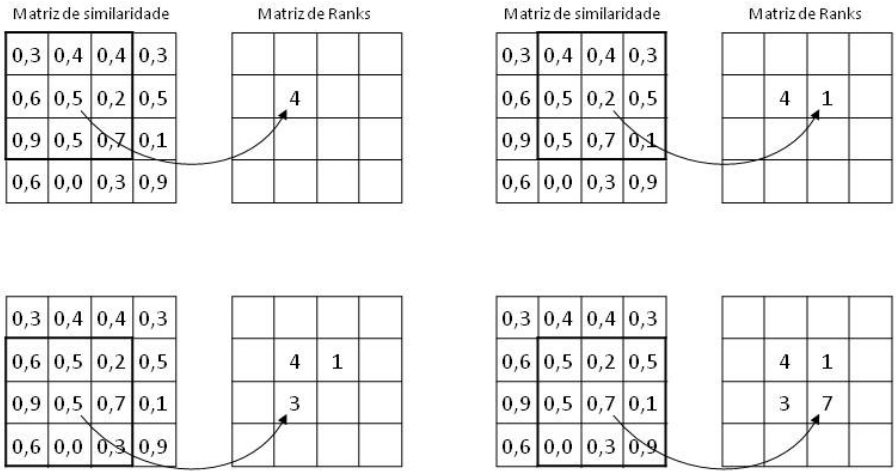

Figure 1. Exemplo de construção de uma matriz de rank. [Choi 2000]

Com base na matriz de *rank*, o *C99* utiliza um método de *clustering* baseado no algoritmo de maximização de Reynar [Reynar 1998] para identificar os limites entre os segmentos.

2.2. Segmentação de Referência

Para que se possa avaliar um segmentador automático de textos é preciso uma referência, isto é, um texto com os limites entre os segmento 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.

Entre as abordagens mais comuns para se conseguir essas referências, encontramos:

- A concatenação aleatória de documentos distintos, onde o ponto entre o final de um texto e o início do seguinte é um limite entre eles.
- A segmentação manual dos documentos, em que, pessoas capacitadas, também chamadas de juízes, ou mesmo o autor do texto, são consultadas e indicam manualmente onde há uma quebra de segmento.
- Em transcrição de conversas faladas em reuniões com múltiplos participantes, um mediador é responsável por encerrar um assunto e iniciar um novo, nesse caso o mediador anota manualmente o tempo onde há uma transição de tópico.

Em aplicações em que a segmentação é uma tarefa secundária e quando essas abordagens são custosas ou não se aplicam, é possível, ao invés de avaliar o segmentador, analisar seu impacto na aplicação final.

2.3. Medidas de Avaliação

As medidas de avaliação tradicionais como precisão e revocação são usadas em recuperação de informação e classificação automática para medir o desempenho de modelos de classificação e predição. São baseadas na comparação dos valores produzidos por uma hipótese com os valores reais.

Essas medidas de avaliação computam os erros do algoritmo, isto é, falsos positivos e falsos negativos, a fim de calcular seu desempenho.

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, no mesmo ponto, não há.

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, é a proporção de limites corretamente identificados pelo algoritmo. É calculada dividindo-se o número de limites identificados automaticamente pelo número de candidatos a limite (Equação 2).

$$Presisão = \frac{VP}{VP + FP} \tag{2}$$

Essa medida varia entre $0,0$ e $1,0$, que indica a proporção de limites identificados pelo algoritmo que são corretos, ou seja, correspondem a um limite real na segmentação de referência. Porém não diz nada sobre quantos limites reais existem.

A revocação, é a proporção de limites verdadeiros que foram identificados pelo algoritmo. É calculada dividindo-se o número de limites identificados automaticamente pelo número limites verdadeiros.

$$Revocação = \frac{VP}{VP + FN} \tag{3}$$

Pode variar entre $0,0$ e $1,0$, onde indica que a proporção de limites corretos que foram identificados. Porém não diz nada sobre quantos limites foram indentificados incorretamente.

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.

Além dessas medidas, que consideram apenas se um segmento foi corretamente definido, pode-se também considerar a distância entre o segmento extraído automaticamente e o segmento de referência [Kern and Granitzer 2009].

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 a segunda hipótese possa ser considerada superior à primeira se levado em conta a proximidade dos limites.

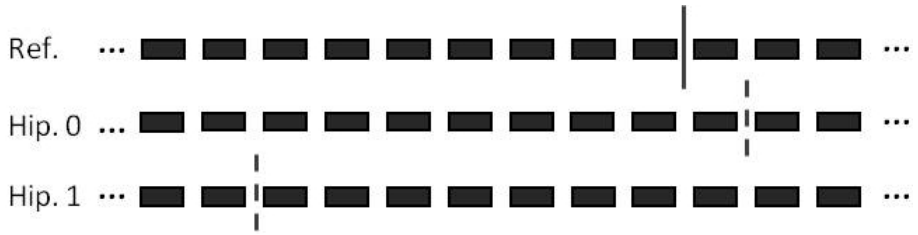

Figure 2. Exemplos de *near missing* e falso positivo puro. Os blocos indicam uma unidade de informação e as linha verticais representam uma transição de assunto.

Considerando o conceito de *near misses*, algumas soluções foram propostas. As medidas de avaliação mais utilizadas são:

P_k, proposta por Beeferman [Beeferman et al. 1999], 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 distancia 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. Pevzner [Pevzner and Hearst 2002] apresenta uma medida alternativa à P_k [Beeferman et al. 1999] a fim de melhorar alguns aspectos. 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.

3. Trabalhos Relacionados

Semelhante a este trabalho, outras abordagens foram propostas no sentido de propor segmentadores para outros idiomas além do inglês bem como alvaliá-los em contextos específicos como discursos e conversas em reuniões.

A fim de aplicar os algoritmos *TextTiling* e *C99* ao idioma árabe, foi utilizado como *corpus* a concatenação de textos extraídos de notícias de diferentes países como Tunísia, Egito e Algeria, que tratam de assuntos distintos como política, esporte, cultura, história, tecnologia e artes, e que trazem particularidades nos estilos de escrita e até diferenças entre dialetos locais [Chaibi et al. 2014].

Foram propostos ajustes na etapa de preprocessamento onde verificou-se que diferenças no dialeto de cada pais devem ser consideradas no processo de segmentação pois que melhores resultados dependem da escolha de um *stemmer* adequado à cada dialeto.

As técnicas de segmentação também foram aplicadas em conversas em reuniões com múltiplos participantes onde se estuda os textos extraídos dos discursos, ou seja, o texto a ser segmentado é uma transcrição das falas dos participantes durante a reunião. Nesse sentido, como parte do projeto *CALO-MA*[1], foi apresentado um segmentador baseado no *TextTiling*, que foi aplicado em reuniões com múltiplos participantes. Os autores apoiaram-se em elementos da fala como pausas, trocas de falantes e entonação para encontrar melhores segmentos [Galley et al. 2003]. O *corpus* utilizado contém a transcrição da fala dos participantes durante as reuniões que foram conduzidas por um mediador que propunha os tópicos e anotava o tempo em que os participantes mudavam de assunto [Banerjee and Rudnicky 2006, Tur et al. 2010].

Ainda no contexto de reuniões com múltiplos participantes temos a segmentação funcional dos discursos, onde outros aspectos podem ser analisados, como a participação dos presentes na reunião. Nesse sentido, estuda-se a contribuição das pessoas as quais

[1]http://www.ai.sri.com/project/CALO

podem ser categorizadas, por exemplo, em diálogos, discussões e monólogos. Sugere-se que alguns comportamentos podem dar pistas de mudança de discurso, como quando um participante toma a palavra por um tempo prolongado [Bokaei et al. 2015].

A presença de *pistas* pode ser um indicativo que ajuda a aprimorar a detecção de limites entre segmentos, uma vez que alguns elementos são frequentes na transição de tópicos. Essas *pistas* podem ser palavras como *"Ok"*, *"continuando"* e frases como *"Boa noite"*, *"Dando prosseguimento"* ou pausas prolongadas que ajudam a indicar uma mudança de tópicos. Essas *pistas* podem ser detectadas por meio de algoritmos de aprendizado de máquina ou anotadas manualmente [yun Hsueh et al. 2006, Galley et al. 2003, Beeferman et al. 1999].

Quanto a eficácia dos algoritmos, observou-se que aqueles que apresentam melhor performance o fazem ao custo de maior complexidade computacional, que se deve muitas vezes à construção de matrizes de similaridade entre todas as sentenças como em [Choi 2000]. Nesses casos pode-se diminuir o tempo computacional evitando calcular a similaridade entre trechos muito distantes no texto.

Nesse sentido, foi apresentada uma abordagem que otimiza o processo ao computar as médias das similaridades entre sentenças de cada bloco, a qual chamou de *inner similarity* e em seguida usou esses valores para calcular as medias das similaridades entre todos os blocos a qual chamou de *outter similarity* [Kern and Granitzer 2009]. Dessa forma não é criada uma matriz que contem as similaridades de todas as sentenças, mas apenas daquelas mais próximas. Os autores reportaram uma eficiência superior e uma eficácia comparável aos algoritmos mais complexos.

Para a obtenção do corpus, bem como segmentações de referência, muitos trabalhos utilizam a concatenação de textos na avaliação, os quais foram extraídos de coleções documentos como *TDT*[2] e *WSJ*[3]. Nesses casos, tem-se facilmente os segmentos de referência onde cada limite de segmento separa os documentos originais. Outros avaliam seus trabalhos utilizando coleções de transcrições de áudios gravados durante conversas de reuniões entre pessoas como *ICSI* [Janin et al. 2003] e *AMI* [Carletta et al. 2006]. Nesses casos, os segmentos de referência são dados por anotações feitas durante a reunião, onde registrou-se o tempo em que os participantes trocaram de assunto.

4. Proposta: Segmentação Linear Automática de Atas de Reunião

Os algoritmos *TextTiling* e *C99* foram propostos para o inglês, independentemente de domínio, ou seja, a proposta inicial dos autores é trabalhar em qualquer texto nessa língua. Assim, propõe-se aplicá-los ao contexto das atas de reunião em português do Brasil, ou seja, em uma língua diferente e dentro de um contexto específico. As subseções seguintes tratam da aplicação desses algoritmos para esse nicho mais específico. A Seção 5 mostra a avaliação experimental e resultados obtidos.

O vocabulário das reuniões, ainda que em tópicos diferentes, compartilham certo vocabulário pertencente ao ambiente onde se deram as reuniões. Isso é um fator que diminui a coesão léxica entre os segmentos. As atas de reunião costumam ter um estilo de escrita que deve ser levado em conta na aplicação dos algoritmos, como a identificação de

[2]https://catalog.ldc.upenn.edu/LDC98T25
[3]https://catalog.ldc.upenn.edu/ldc2000t43

finais de sentença na ausência de quebras de parágrafo, inserção de linhas que não separam assuntos, utilização de pontuação para transição de tópicos, e cabeçalhos e numerais ruidosos.

Nas subseções a seguir serão apresentados o pré-processamento e a identificação de segmentos candidatos considerados para a segmentação de atas.

4.1. Pré-processamento

O texto a ser segmentado frequentemente é extraído de documentos do tipo *pdf*, *doc*, *docx* ou *odt* que normalmente possuem formato binário. A fim de extrair o texto desses documentos, aplicou-se um processo que transforma esses formatos em aquivos de texto plano. Após isso, uma ata tem em média 906 *tokens*, incluindo elementos menos informativos. A fim de preparar o texto e selecionar as palavras mais significativas, este passa por processos de transformação os quais serão apresentados a seguir.

1. Remoção de cabeçalhos e rodapés: as atas frequentemente 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 o algoritmo de segmentação, quanto a leitura do texto pelo usuário.
2. Identificação de finais sentenças: cada final de sentença é identificado e marcado com uma *string* especial, esse processo é melhor descrito na Subseção 4.2.
3. Redução de termos: Eliminou-se a acentuação, sinais de pontuação, numerais e todos os *tokens* menores que três caracteres. Palavras de uso muito frequente como artigos, preposições e pronomes, chamadas de *stop words*, foram removidas utilizando-se uma lista de 438 palavras.
4. *Stemming*: Extrai-se o radical de cada palavra, para isso, as letras são convertidas em caixa baixa e aplica-se o algoritmo *Orengo*[4] para remoção de sufixos.

Processo	**Média de *tokens***
Extração do texto	906
Remoção de Cabeçalhos Rodapés	813
Identificação de finais de sentença	813
Limpeza	535
Remoção de Numerais	526
Remoção de *Stop Words*	441
Stemming	441

Table 1. Quantidade média de *tokens* extraídos após cada passo do pré-processamento.

A Figura 3 mostra a etapa de pré-processamento em uma sentença em português.

4.2. Identificação de candidatos

É preciso fornecer aos algoritmos os candidatos iniciais a limites de segmento. Para isso, é necessário escolher qual será a unidade de informação mínima que constitui um

[4]http://www.inf.ufrgs.br/ viviane/rslp/

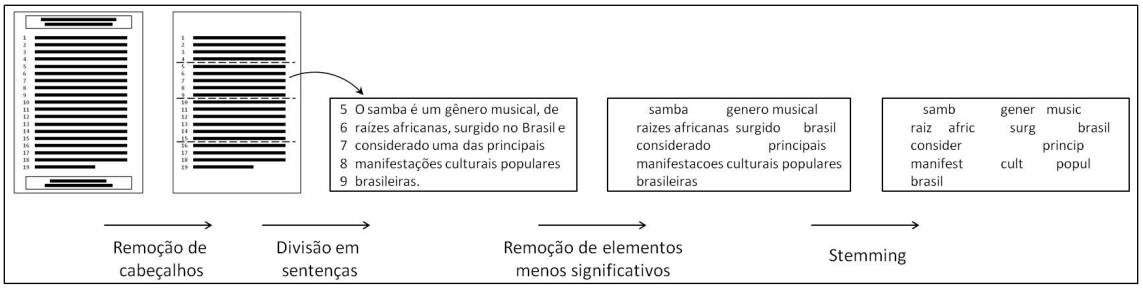

Figure 3. Exemplo de pré-processamento.

segmento. Baseando-se no estilo de escrita e considerando as pontuações de um texto, é possível, em alguns casos, indicar quebras de parágrafo, finais de sentenças ou mesmo palavras como elementos que encerram um segmento.

Ocorre que em atas de reunião é uma prática comum redigí-las de forma que o conteúdo discutido fica em parágrafo único. Além disso, as quebras de parágrafo são usadas para formatação de outros elementos como espaço para assinaturas. Assim, neste trabalho, os finais de sentença são considerados unidades de informação e portanto, passíveis a limite entre segmentos.

Devido ao estilo de pontuação desses documentos, como encerrar sentenças usando um *";"* e inserção de linhas extras, foram usadas as regras apresentadas no Algoritmo 1 para identificar os finais de sentenças.

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

5. Avaliação Experimental

De acordo com [Pevzner and Hearst 2002] há duas principais dificuldades na avaliação de segmentadores automáticos. A primeira é conseguir uma referência, já que juízes humanos costumam não concordar entre si, sobre onde os limites estão e outras abordagens podem não se aplicar ao contexto. A segunda é que tipos diferentes de erros devem ter pesos diferentes de acordo com a aplicação. Há casos onde certa imprecisão é tolerável e outras, como a segmentação de notícias, onde a precisão é mais importante.

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.

Para obter um melhor resultado com as técnicas de segmentação textual apresentadas nesse artigo, faz-se necessário conhecer quais valores otimizam a configuração de um algoritmo. Deseja-se também saber qual a influência do preprocessamento no desempenho dos algoritmos. Assim, nesse trabalho, refere-se como configuração os valores que foram atribuídos aos parâmetros de um algoritmo e à presença ou ausência da etapa de preprocessamento.

5.1. Bases

A fim de obter um conjunto de documentos segmentados que possam servir como referência na avaliação, foram coletadas as atas de reuniões do Conselhos de Pós Graduação, Conselho de Cursos e Conselho de Departamento de da UFSCar-Sorocaba. Os documentos foram oferecidos à profissionais que participam de reuniões desse departamento para segmentá-las. Para isso, utilizou-se um *software* que permitiu aos voluntários visualizar um documento, e indicar livremente as divisões entre segmentos com um assunto relativamente independente. Ao final, o software coletou os dados da segmentação de doze das os quais serviram como referência para a avaliação dos algoritmos.

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.

5.2. Configuração experimental

As implementações dos algoritmos permitem ao usuário a configuração de seus parâmetros. O *TextTiling* permite ajustarmos dois parâmetros, sendo o primeiro o tamanho da janela para o qual atribuiu-se os valores 20, 40 e 60. Para o segundo parâmetro, o passo, atribuiu-se os valores 3, 6, 9 e 12. Gerando ao final 20 configurações.

O *C99* permite ajustarmos três parâmetros, sendo, o primeiro a quantidade segmentos desejados, uma vez que, não se conhece o número ideal de segmentos, calcula-se uma proporção dos candidatos a limite. Para isso atribuiu-se os valores 0,2; 0,4; 0,6; 0,8; 1,0. 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. Permite ainda, definirmos se as sentenças serão representados por vetores contendo a frequência ou o peso de cada termo, onde ambas as representações foram utilizadas. Considerando todos os parâmetros, foram gerados 20 configurações para o algoritmo C99.

5.3. Critérios de avaliação

Os algoritmos foram comparados com a segmentação fornecida pelos especialistas e calculou-se as medidas tradicionais acurácia, precisão, revocação, F^1. Além dessas, computou-se também as medidas mais aplicadas à segmentação textual, P_k e *WindowDiff*.

5.4. Resultados

Inicialmente testou-se os algoritmos configurando seus parâmetros conforme mostrado na Subseção 5.2 sem aplicar o preprocessamento, afim de conhecer quais parâmetros melhor

configuram os agoritmos considerando o texto integral.

Na Tabela 2 são apresentadas, para cada medida, as configurações que otimizam o *TextTiling* e a média computada considerando as bases, onde **J** é o tamanho da janela e **P** é o passo.

Medida	**J**	**P**	**Média**
Acurácia	50	6	0,612
Precisão	40	9	0,611
Revocação	20	3	0,886
F^1	30	6	0,605
P_k	50	9	0,142
WindowDiff	50	6	0,387

Table 2. Resultados obtidos com o *TextTiling* sem o preprocessamento.

Na Tabela 3 são apreentadas, para cada medida, as configurações que otimizam o *C99* e a média computada considerando as bases, 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 *ranking* e **W** indica se os segmentos são representados por vetores contendo a frequência ou um peso das palavras.

Medida	**S**	**M**	**W**	**Média**
Acurácia	60	9	Sim	0,588
Precisão	40	9	Sim	0,645
Revocação	80	9	Sim	0,869
F^1	80	9	Sim	0,638
P_k	20	9	Sim	0,134
WindowDiff	60	9	Sim	0,411

Table 3. Resultados obtidos com *C99* sem o preprocessamento.

Repetiu-se os testes anteriores incluindo o preprocessando para verificar o impacto dessa etapa na segmentação das atas e conhecer quais valores melhor configuram os algoritmos considerando o texto preprocessado. Nas Tabelas 4 e 5 são apresentadas as configurações que otimizam os algoritmos incluindo essa etapa.

Medida	**J**	**P**	**Média**
Acurácia	40	9	0,603
Precisão	50	12	0,613
Revocação	20	3	0,917
F^1	40	3	0,648
P_k	50	9	0,144
WindowDiff	40	9	0,396

Table 4. Resultados obtidos com o *TextTiling* com o preprocessamento.

Com o teste de Friedman e a análise do diagrama de diferença crítica foi possível testar os dois principais algoritmos e o impacto do preprocessamento, gerando quatro

Medida	S	M	W	Média
Acurácia	60	9	Sim	0,609
Precisão	20	11	False	0,720
Revocação	80	11	Sim	0,897
F^1	80	11	Sim	0,655
P_k	20	11	False	0,116
WindowDiff	60	9	Sim	0,390

Table 5. Resultados obtidos com *C99* com o preprocessamento.

configurações para cada medida. Deseja-se encontrar entre essas, qual delas melhor otimiza determinada medida. Para isso, aplicou-se novamente os critérios de avaliação. De acordo com os testes, o algoritmo *C99* obteve melhor desempenho em acurácia, precisão, F^1, P_k e *WindowDiff*, como pode ser visto na Tabela 6.

Medida	S	M	W	P	Média
Acurácia	60	9	Sim	Sim	0,609
Precisão	20	11	Não	Sim	0,720
F^1	80	11	Sim	Sim	0,655
P_k	20	11	Não	Sim	0,116
WindowDiff	60	9	Sim	Sim	0,390

Table 6. Resultados obtidos com *C99*.

O *TextTiling* obteve o melhor desempenho em revocação, utilizando-se janelas de tamanho igual a 20 e passo igual a 3, onde registrou-se uma média de $0,917$ para essa medida.

6. Conclusão

As atas de reunião, objeto de estudo desse artigo, apresentam características peculiares em relação à discursos e textos em geral. Características como segmentos curtos e coesão mais fraca devida ao estilo que evita repetição de palavras e ideias em benefício da leitura por humanos, dificultam o processamento por computadores.

Os algoritmos *TextTiling* e *C99* foram testados em um conjunto de atas coletadas do Departamento de Computação da UFSCar-Sorocaba. Por meio da análise dos dados chegou-se a uma configuração cujos segmentos melhor se aproximaram das amostras segmentadas por participantes das reuniões.

Na maioria das medidas, o algoritmo *C99* sobressaiu-se em relação ao *TextTiling*, contudo, os testes estatísticos, não apresentam diferença significativa.

Da mesma forma, a etapa de preprocessamento proporciona melhora de performance quando aplicada, porém o seu maior benefício é a diminuição do custo computacional, uma vez que não prejudica a qualidade dos resultados.

A segmentação de atas de reunião pode ajudar na organização, busca e compreensão dos conteúdos nelas contidos. Também outros domínios e aplicações diferentes podem se beneficiar dos resultados apresentados, como aplicações voltadas a resgate de

informação, sumarização e acessibilidade. Assim, espera-se que outros trabalhos possam aproveitar deste.

Em trabalhos futuros, serão investigadas técnicas de extração de tópicos para descrever os segmentos e com isso aprimorar o acesso ao conteúdo das atas de reunião.

References

Banerjee, S. and Rudnicky, A. (2006). A texttiling based approach to topic boundary detection in meetings. volume 1, pages 57–60. cited By 3.

Beeferman, D., Berger, A., and Lafferty, J. (1999). Statistical models for text segmentation. *Machine Learning*, 34(1):177–210.

Boguraev, B. K. and Neff, M. S. (2000a). Discourse segmentation in aid of document summarization. In *In Proceedings of Hawaii Int. Conf. on System Sciences (HICSS-33), Minitrack on Digital Documents Understanding, IEEE.*

Boguraev, B. K. and Neff, M. S. (2000b). Lexical cohesion, discourse segmentation and document summarization. In *In RIAO-2000, Content-Based Multimedia Information Access*.

Bokaei, M. H., Sameti, H., and Liu, Y. (2015). Linear discourse segmentation of multi-party meetings based on local and global information. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 23(11):1879–1891.

Carletta, J., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., Kraaij, W., Kronenthal, M., Lathoud, G., Lincoln, M., Lisowska, A., McCowan, I., Post, W., Reidsma, D., and Wellner, P. (2006). The ami meeting corpus: A pre-announcement. In *Proceedings of the Second International Conference on Machine Learning for Multimodal Interaction*, MLMI'05, pages 28–39, Berlin, Heidelberg. Springer-Verlag.

Chaibi, A. H., Naili, M., and Sammoud, S. (2014). Topic segmentation for textual document written in arabic language. *Procedia Computer Science*, 35:437 – 446.

Choi, F. Y. Y. (2000). Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 26–33, Stroudsburg, PA, USA. Association for Computational Linguistics.

Choi, F. Y. Y., Wiemer-Hastings, P., and Moore, J. (2001). Latent semantic analysis for text segmentation. In *In Proceedings of EMNLP*, pages 109–117.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

Dias, G., Alves, E., and Lopes, J. G. P. (2007). 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'07, pages 1334–1339. AAAI Press.

Galley, M., McKeown, K., Fosler-Lussier, E., and Jing, H. (2003). Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting on Asso-*

ciation for Computational Linguistics - Volume 1, ACL '03, pages 562–569, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hearst, M. A. (1994). Multi-paragraph segmentation of expository text. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 9–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Janin, A., Baron, D., Edwards, J., Ellis, D., Gelbart, D., Morgan, N., Peskin, B., Pfau, T., Shriberg, E., Stolcke, A., and Wooters, C. (2003). The icsi meeting corpus. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 364–367.

Kern, R. and Granitzer, M. (2009). Efficient linear text segmentation based on information retrieval techniques. pages 167–171. cited By 10.

Lee, J.-K., Song, H.-J., Park, S.-B., and Bassiliades, N. (2011). Two-step sentence extraction for summarization of meeting minutes. volume 39, pages 614–619.

Misra, H., Yvon, F., Jose, J. M., and Cappe, O. (2009). Text segmentation via topic modeling: An analytical study. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1553–1556, New York, NY, USA. ACM.

Pevzner, L. and Hearst, M. (2002). A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36. cited By 154.

Reynar, J. C. (1998). *Topic Segmentation: Algorithms and Applications*. PhD thesis, Philadelphia, PA, USA. AAI9829978.

Reynar, J. C. (1999). Statistical models for topic segmentation. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 357–364, Stroudsburg, PA, USA. Association for Computational Linguistics.

Schwartz-Ziv, M. and Weisbach, M. S. (2013). What do boards really do? evidence from minutes of board meetings. *Journal of Financial Economics*, 108(13):349–366.

Sitbon, L. and Bellot, P. (2004). Adapting and comparing linear segmentation methods for french. In *Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval*, RIAO '04, pages 623–637, Paris, France, France. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

TAKAHASHI, M., KIDO, K., and HASHIMOTO, K. (2013). Towards the profitability trend extraction from the board meeting proceedings. *17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013*, 78.

Tur, G., Stolcke, A., Voss, L., Peters, S., Hakkani-Tür, D., Dowding, J., Favre, B., Fernández, R., Frampton, M., Frandsen, M., Frederickson, C., Graciarena, M., Kintzing, D., Leveque, K., Mason, S., Niekrasz, J., Purver, M., Riedhammer, K., Shriberg, E., Tien, J., Vergyri, D., and Yang, F. (2010). The calo meeting assistant system. *Trans. Audio, Speech and Lang. Proc.*, 18(6):1601–1611.

yun Hsueh, P., Moore, J. D., and Renals, S. (2006). Automatic segmentation of multiparty dialogue. In *EACL*.