Segmentação Textual Automática de Atas de Reunião

Abstract. *The abstract goes here!*

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.*

1. Introdução

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. Normalmente esse tipo de busca exige a inserção de termos exatos e apresentam ao usuário um documento com as palavras buscadas em destaque, mantendo o texto longo, o que 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.

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 2000, Dias et al. 2007].

As atas de reunião diferem dos textos comumente estudados em outros trabalhos em alguns pontos. O estilo de escrita mais sucinto, com poucos detalhes reduz dificulta o processo de segmentação [Choi et al. 2001]. Há também um maior foco no idioma inglês, presente na maioria dos artigos publicados. 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 e detalhado a avaliação dos experimentos e seus resultados são reportados. Por fim, na Seção 5 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. 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 e cada ponto entre duas unidades é 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 2000].

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.

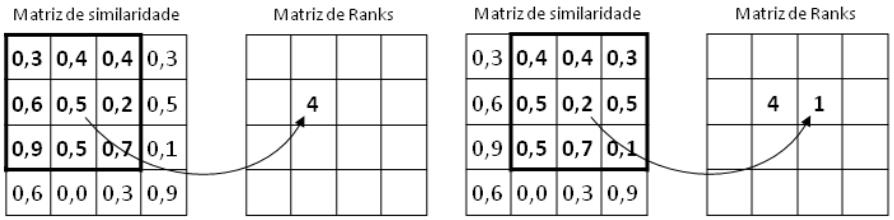
O *TextTiling* possui baixa complexidade computacional. Por outro lado, algoritmos mais complexos apresentam acurácia relativamente superior como apresentado em [Choi 2000, Kern and Granitzer 2009, Misra et al. 2009].

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, cada elemento e comparado com seus vizinhos dentro de uma região denominada máscara.

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

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.



(a) Passo 1 (b) Passo 2

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

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: 1) 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. 2) 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. 3) 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 computam os erros do algoritmo, isto é, falsos positivos e falsos negativos, a fim de calcular seu desempenho. 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]. Chama-se *near misses* o caso em que um limite identificado automaticamente não coincide exatamente com a referência, mas é necessário considerar a proximidade entre eles.

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

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

Proposta por [Beeferman et al. 1999], P_k, atribui valores parciais a *near misses*, ou seja, limites sempre receberão um peso proporcional à sua proximidade, desde que

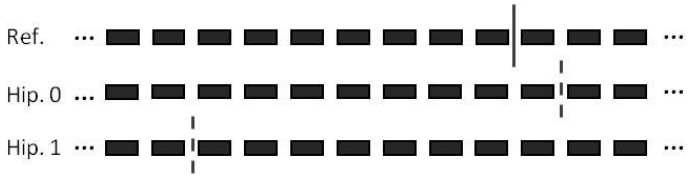


Figure 2. Exemplos de *near missing* e falso positivo puro. Os blocos indicam uma unidade de informação e as linha verticais representam uma transição de assunto.

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.

[Pevzner and Hearst 2002] apresentam uma medida alternativa à P_k a fim de melhorar alguns aspectos. De maneira semelhante, *WindowDiff* 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 re-

união. 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].

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].

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. Neste trabalho, 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 implementações utilizadas, bem como ferramentas utilizadas e os resultados completos estão disponíveis para utilização e consulta em `link`. As subseções seguintes tratam da aplicação desses algoritmos para esse nicho mais específico.

4.1. Coleção de documentos

A fim de obter um conjunto de atas a serem segmentadas automaticamente, coletou-se 6 atas de reunião do Conselho de Pós-Graduação da UFSCar Sorocaba[2]. Selecionou-se atas que registram cinco reuniões ordinárias e uma extraordinária (Ata 6). Tomou-se esse cuidado para manter a representatividade e diversidade do conjunto.

Os documentos apresentam um texto com poucas quebras de parágrafo e sem marcações de estrutura, como capítulos, seções ou quaisquer indicações sobre o assunto do texto. É comum a presença de cabeçalhos, rodapés e numeração de páginas e linhas o que pode prejudicar tanto similaridade entre sentenças como a apresentação dos segmentos ao usuário.

4.2. Pré-processamento

As atas a serem segmentadas são extraídas 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. A fim de preparar o texto e selecionar as palavras mais significativas, as atas passaram por processos de transformação os quais serão apresentados a seguir.

1. Remoção de cabeçalhos e rodapés: as atas contém trechos que podem ser considerados pouco informativos e descartados durante o pré-processamento, como

[1]http://www.ai.sri.com/project/CALO
[2]http://www.ppgccs.net/?page_id=1150

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: 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. Os detalhes sobre essas regras estão disponíveis para consulta em `link`.
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*: Extraiu-se o radical de cada palavra. Para isso, as letras foram convertidas em caixa baixa e aplicou-se o algoritmo *Orengo*[3] para remoção de sufixos.

Na Tabela 1 é mostrado, para cada ata, a quantidade de *tokens* após a extração dos documentos e durante o pré-processamento.

Processo	**Ata 1**	**Ata 2**	**Ata 3**	**Ata 4**	**Ata 5**	**Ata 6**
Extração do texto	809	851	1043	1407	834	496
Remoção de Cabeçalhos e Rodapés	665	704	840	1247	708	392
Redução de termos	461	489	566	872	485	286

Table 1. Quantidade de *tokens* por ata.

4.3. Segmentação de Referência

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

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

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

Table 2. Quantidade de sentenças e segmentos de referência por ata.

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

4.4. Configuração experimental

As implementações dos algoritmos permitem ao usuário a configuração de seus parâmetros. 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*.

4.5. 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 4.4, 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 `link`.

4.6. Resultados

Obteve-se, por meio dos testes estatísticos apresentados, as melhores configurações para as principais medidas de avaliação de segmentadores. Com essas configurações calculou-se a média de cada medida considerando o conjunto de documentos. Na Tabela 3 são

Medida	Sem Pré-processamento			Com Pré-processamento		
	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

Table 3. Resultados obtidos com o *TextTiling*

apresentadas, as médias obtidas com o *TextTiling* bem como as configurações utilizadas, onde **J** é o tamanho da janela e **P** é o passo.

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

Medida	Sem Pré-processamento				Com Pré-processamento			
	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

Table 4. Resultados obtidos com o *C99*

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

Algoritmo	**Medida**	**Parâmetros**	**Pré-processamento**	**Média**
C99	P_k	S=20 M=11 W=Não	Sim	0,116
C99	*WindowDiff*	S=60 M=09 W=Sim	Sim	0,390
C99	Acurácia	S=60 M=09 W=Sim	Sim	0,609
C99	Precisão	S=20 M=11 W=Não	Sim	0,720
C99	F^1	S=80 M=11 W=Sim	Sim	0,655
TextTiling	Revocação	J=20 P=3	Sim	0,917

Table 5. Melhores resultados obtidos.

De maneira geral, o algoritmo *C99* apresentou melhores resultados em relação ao *TextTiling*, sobre tudo quando aplicado o pré-processamento, contudo, testes estatísticos realizados indicaram que não houve diferença significativa entre os métodos. A etapa de preprocessamento proporciona melhora de desempenho quando aplicada, porém o seu

maior benefício é a diminuição do custo computacional, uma vez que não prejudica a qualidade dos resultados.

As medidas de avaliação tradicionais, 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.

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.

Observa-se melhores resultados de *WindowDiff* quando os algoritmos aproximam a quantidade de segmentos automáticos da quantidade de segmentos da referência. Por outro lado, observa-se que P_K avalia melhor as configurações que retornam menos segmentos. A configuração do tamanho do passo (P) e da proporção de segmentos em relação ao número de candidatos (S), influenciam os algoritmos na quantidade de segmentos extraídos. Contudo, não é possível definir um valor adequado, uma vez que os segmentadores humanos frequentemente apontam valores diferentes.

5. Conclusão

As atas de reunião, objeto de estudo desse artigo, apresentam características peculiares em relação à discursos e textos em geral. Características como ausência de parágrafos, segmentos curtos e 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. Os segmentos gerados automaticamente foram comparados à segmentação manual fornecida por participantes das reuniões. Por meio da análise dos chegou-se as configurações cujos segmentos melhor se aproximaram das amostras fornecidas. Os resultados obtidos indicam que os algoritmos originalmente propostos para o idioma inglês e usualmente avaliados em outros contextos podem ser aplicados as atas de reunião.

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. A metodologia apresentada e as ferramentas disponibilizadas podem ser úteis na avaliação e configuração de algoritmos de segmentação em outros contextos.

Em trabalhos futuros, serão investigadas técnicas de classificação e 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. (2000). 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*.

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.

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.

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 Association 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.

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

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.

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.

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*.