# Segmentação Topical Automática de Atas de Reunião

Ovídio José Francisco
UFSCar – Sorocaba
Email: ovidiojf@gmail.com

***Resumo*—The abstract goes here.**

## I. Introdução

Reuniões são tarefas presentes em atividades corporativas, ambientes de gestão e organizações de um modo geral. São frequentemente registradas em texto na forma de atas de reunião para fins de documentação e consulta posterior. A organização e consulta manual desses arquivos torna-se uma tarefa custosa, especialmente considerando o seu crescimento em uma instituição.

As atas são documentos textuais e portanto constituem dados não estruturados, assim, um sistema que automaticamente responde a consultas do usuário ao conteúdo das atas é um desafio que envolve a sua compreensão [5]. Uma vez que a ata registra a sucessão de assuntos discutidos na reunião, tal sistema tem duas principais tarefas: descobrir quando há uma mudança de assunto, e quais são esses assuntos [1]. Esse trabalho tem foco principal na primeira tarefa, a segmentação topical automática.

Frequentemente atas de reunião tem 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.

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 tópicos.

É útil em aplicações que trabalham com textos sem indicações de quebras de assunto, ou seja, não apresentam parágrafos, seções ou capítulos, como transcrições automáticas de áudio, vídeos e grandes documentos que contêm vários assuntos como atas de reunião e noticias.

O interesse por segmentação textual tem crescido em em aplicações voltadas a recuperação de informação [14] e sumarização de textos [3] [4]. 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 maior que pode conter informações menos pertinentes. A sumarização de texto pode ser aprimorada ao processar segmentos separados por tópicos ao invés de documentos inteiros. A navegação pelo documento pode ser aprimorada, em especial na utilização por usuários com deficiência visual [7].

Assim, esse trabalho trata da adaptação e avaliação de algoritmos tradicionais ao contexto de documentos em português do Brasil, com ênfase especial nas atas de reuniões.

## II. Referencial Teórico

Trabalhos anteriores apontam que a coesão léxica é um forte indicador da estrutura do texto [9] [3]. Os principais algoritmos de segmentação textual baseiam-se na ideia de coesão léxica em um assunto. Isto é, a mudança de tópicos é acompanhada de uma proporcional mudança de vocabulário. A partir disso, vários algoritmos foram propostos sob o pressuposto de que um segmento pode ser identificado e delimitado pela análise das palavras que o compõe.

Uma vez que a coesão léxica é pressuposto básico da maioria dos algoritmos, o cálculo da similaridade entre textos e fundamental. Uma medida de similidade frequentemente utilizada é a *cosine*, a qual pode ser vista na equação 1, sendo $f_{x,j}$ a frequência da palavra $j$ na sentença $x$ e $f_{y,j}$ sendo a frequência da palavra $j$ na sentença $y$.

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

Entre os trabalhos mais influentes podemos citar o *TextTiling* [10] proposto por Hearst. Ela propõe um algoritmo baseado em janelas deslizantes, onde para cada candidato a limite, analisa-se o texto circundante. Um limite ou quebra se segmento é identificado quando a similaridade entres os blocos apresenta uma queda considerável.

Ela propõe um algoritmo baseado em janelas deslizante, para analisar blocos de texto adjacentes e identificar os limites com base nas similaridades dos blocos.

O algoritmo 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 *cosini* (equação 1) para calcular a similaridade entre os blocos.

Finalmente, os limites são identificados sempre que a similaridade entre blocos adjacentes entre cada candidato ultrapassa um determinado *threshold*

Apresenta baixa complexidade computacional, devido a simplicidade do algoritmo e baixa eficiência quando comparado a outros métodos mais sofisticados como mostrando em [7], [11].

Choi [7] apresenta um trabalho que usa *cosine*, a qual é exibida na equação 1, como medida similaridade e apresenta um esquema de ranking em seu algoritmo, o *C99*. Embora muitos dos melhores trabalho utilizarem matrizes de similaridades, o autor traz obervações. Ele aponta que para pequenos segmentos, o cálculo de suas similaridades não é confiável. Pois uma ocorrência adicional de uma palavra causa um impacto desproporcional no cálculo. 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 apresenta um esquema de ranking para contornar esse problema.

Cada valor na matriz de similaridade é substituído por seu ranking local. Onde ranking é o número de elementos vizinhos com similaridade menor, o qual e calculado com a equação 2. Um exemplo é mostrado na Figura 1 abaixo, onde utiliza-se uma máscara de largura igual a 3.

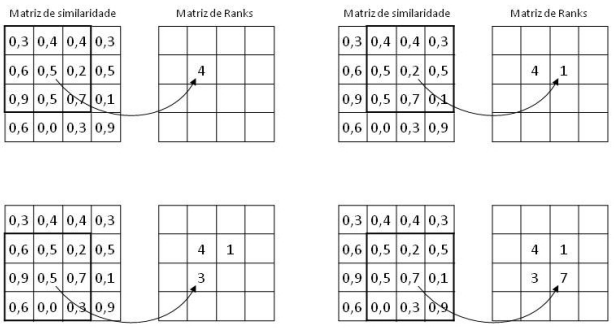


Figura 1: Exemplo de construção de uma matriz de rank

$$r(x,y) = \frac{Numero\ de\ elementos\ com\ similaridade\ menor}{Numero\ de\ elementos\ examinados} \quad (2)$$

Finalmente, na etapa de *clustering*, utiliza um método baseado no algoritmo de maximização de Reynar [13] para identificar os limites entre os segmentos. Como melhoramento, Choi apresenta uma versão em seu trabalho posterior [8] que utiliza *Latent Semantic Analisys* (LSA) para extrair conhecimento semântico do conjunto de documentos.

## III. Trabalhos Relacionados

Semelhante a esse trabalho, outras abordagens foram propostas como em [6] onde os autores adaptam os *TextTiling* e *C99* ao idioma árabe. Apresentam os resultados de experimentos no qual avaliam a performance em jornais de diferentes países em árabe. As adaptações consistem basicamente na etapa de pré-processamento e apontam que diferenças no dialeto de cada pais devem ser consideradas no processo de segmentação e que a adaptação depende da escolha de um *stemmer* adequado.

Banerjee [1] apresenta um segmentador baseado no *TextTiling* ao contexto das reuniões, com múltiplos participantes. Utiliza como *corpus* a transcrição da fala dos participantes durante a reunião a qual foi conduzida por um mediador que propunha os tópicos e anotava o tempo onde os participantes mudavam o assunto.

Ainda no contexto de reuniões com múltiplos participantes, alguns elementos da fala são utilizados para encontrar melhores segmentos. Bokaei [5], traz um trabalho voltado à segmentação funcional do texto, que foca na atividade dos participantes, as quais categoriza em diálogos e monólogos e sugere que alguns comportamentos podem dar pistas de mudança de tópico, como quando um participante toma a palavra por um tempo prolongado. Galley [9] por sua vez considera de elementos como pausas, trocas de falantes e entonação para encontrar melhores segmentos.

Kern aponta em sua pesquisa [11] que algoritmos que apresentam melhor performance o fazem ao custo de maior complexidade computacional, que se deve à construção de matrizes de similaridade entre todas as sentenças como em [7]. Ele apresenta uma abordagem que otimiza o cálculo ao computar as médias das similaridades de cada bloco, a qual chama de *inner similarity* e em seguida usa esses valores para calcular as medias das similaridades entre todos os blocos a qual chama de *outter similarity*. Dessa forma não é criada uma matriz que contem as similaridades de todas as sentenças, mas apenas daquelas mais próximas. Os autores reportam uma eficiência superior e uma eficiência comparável aos algoritmos mais complexos.

## IV. Adaptação às atas de reunião

Os algoritmos *TextTiling* e *C99* foram propostos para o inglês, independente de domínio, ou seja, a proposta inicial é trabalhar em qualquer texto nessa língua. A proposta desse trabalho é adaptá-los ao contexto das atas de reunião em português do Brasil. As subseções seguintes tratam das adaptações para esse nicho mais específico. A seção V mostra a análise dos algoritmos adaptados.

O vocabulário das reuniões, ainda que em tópicos diferentes, compartilham certo vocabulário pertencente ao ambiente onde as se deram as reuniões. Isso é um fator que diminui a o princípio da coesão léxica entre os segmentos. As atas de reunião costumam ter um estilo de escrita que deve ser levando em conta na adaptação do algoritmos, como a identificação de 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 expostas simples alterações para aumentar a eficiência dos algoritmos e encontra o melhor modelo para a tarefa de segmentar o texto das atas em tópicos.

### A. Préprocessamento

O texto a ser segmentado frequentemente é extraído de documentos em formatos como *pdf* ou de processadores de texto. Após a extração, esse pode passar por processos de transformação os quais serão apresentados a seguir.

A etapa de pre-processamento, em um documento contendo texto puro, acontece em dois passos principais. Primeiro elimina-se as palavras consideradas menos informativas, as quais são chamadas de *stop words*, para isso, utiliza-se uma lista contendo 438 palavras. Em seguida, remove-se os sufixos das palavras restantes, mantendo apenas o radical da palavra. A Figura 2 mostra a etapa de pré-processamento em uma sentença em português.

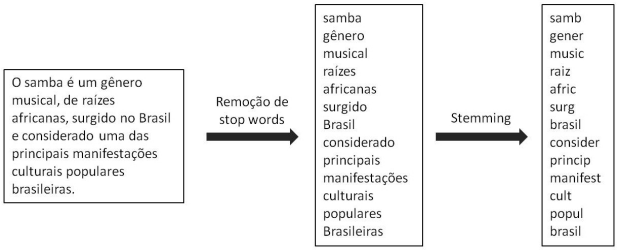


Figura 2: Exemplo de pré-processamento

Há ainda outros passos presentes nessa etapa como remoção de acentos, transformações de caixa, remoção de pontuação, os quais são relativamente simples e não requerem maiores detalhes.

*1) Remoção de ruídos:* As atas frequentemente contém trechos que podem ser considerados pouco informativos e descartados durante o pré-processamento. Após a extração, cabeçalhos e roda-pés se misturam aos tópicos tratados na reunião, podendo ser inseridos no meio de um tópico e criando uma quebra que prejudica tanto o algoritmo de extração, quanto a leitura do texto pelo usuário.

Também é comum o uso de numerais para marcação de páginas e linhas, da mesma forma, são pouco informativos e podem ser removidos.

Nesse trabalho, esses elementos são removidos por meio de heurísticas simples, uma vez que, o descarte não causa perca de informação e pode facilitar a identificação dos segmentos, pois melhora a coesão do texto. Outro benefício é manter os segmentos livres de textos que fogem do assunto.

### B. Identificação de candidatos

É preciso fornecer aos algoritmos os candidatos iniciais a limites de segmento. Aproveitando do estilo de escrita e baseando-se na pontuação do texto é possível indicar quebras de parágrafo, finais de sentenças ou mesmo palavras.

Ocorre que em atas de reunião é uma prática comum redigi-las de forma que o conteúdo discutido fica em parágrafo único, além disso, quebras de parágrafo são usados para formatação de outros elementos como espaço para assinaturas. Indicar todo *token* como ponto candidato obriga a ajustar posteriormente os segmentos de maneira a não quebrar uma ideia ou frase. Assim, nesse trabalho, os finais de sentença são considerado candidatos passíveis a limite entre segmentos.

Devido ao estilo de pontuação desses documentos, como encerrar sentenças usando um *";"* e inserção de linhas extras, usa-se as regras abaixo para identificar os finais de sentenças.

**Algorithm 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**

## V. Avaliação

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, mantendo um conteúdo legível, 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 inicio do seguinte é um limite entre eles. A segmentação manual dos documentos, nesse caso, 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 onde a segmentação é tarefa secundária, analisar seu impacto na aplicação final.

De acordo com [12] há duas principais dificuldades na avaliação de segmentadores automáticos. A primeira é conseguir um 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 do ideal, 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 quantificar a eficiência dos algoritmos, segue uma revisão das principais métricas aplicáveis.

### A. Medidas de Avaliação

As medidas de avaliação tradicionalmente utilizadas em *information retrieval* como precisão e revocação trazem alguns problemas na avalização de segmentadores automáticos. Conforme o algoritmo aponta mais segmentos no texto, tende a melhorar a revocação e ao mesmo tempo, reduzir a precisão, um problema que pode ser contornado usando *F-measure* que faz uma combinação da duas levando em conta seus pesos, o que por outro lado é mais difícil de interpretar. Essas medidas falham ao não serem sensíveis a *near misses*, ou seja, quando um limite não coincide exatamente com o esperado, mas fica próximo [11].

A Figura 3 mostra um exemplo com duas segmentações hipotéticas 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.


Figura 3: Exmplos de segmentação

*1) $P_k$:* A fim de resolver o problema de *near misses*, Beeferman *et. al.* [2] apresentam uma nova medida chama $P_k$ que atribui valores parciais a *near misses*. Esse método move uma janela de tamanho $k$ e a cada posição e verifica se o início e o final da janela estão ou não dentro do mesmo segmento e penaliza o algoritmo em caso de discrepância.

Ou seja, dado duas palavras de distancia $k$, uma discrepância é computada quando o algoritmo e a referência não concordam 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 divido pelo quantidade de segmentações analisadas. Esse valor serve como medida de dissimilaridade entre as segmentações e pode ser interpretada como a probabilidade de duas sentenças extraídas aleatoriamente pertencerem ao mesmo segmento.

*2) WindowDiff:* Pevzner [12] aponta problemas na avaliação mais tradicional Pk [2]. Eles apontam que esse método penaliza demasiadamente os falsos negativos em relação aos falsos positivos e a *near misses*, além disso, desconsidera o tamanho e a quantidade de segmentos, entre outros problemas.

Como solução, propõem um novo método, o qual chamam de *WindowDiff* que traz duas diferenças principais: a dobra a penalidade para os falsos positivos a fim de diminuir o problema da subestimação dessa medida e, diferente de $P_k$, ao mover a janela pelo texto, penaliza o algoritmo sempre que o número de limites proposto pelo algoritmo não coincidir com o número de limites esperados para aquela janela de texto.

Com isso, demonstram em seu trabalho que, em relação a $P_k$, consegue resolver seus principais problemas e mantém sua proposta inicial de sensibilidade a *near misses*, penalizando-os menos que os falsos positivos puros.

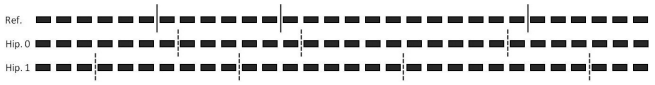


Figura 4: Exemplo de construção de uma matriz de rank

### B. Avaliação dos segmentadores

As implementações dos algoritmos permitem ao usuário a configuração de seus parâmetros. O *TextTiling* permite ajustarmos dois parâmetros, sendo, o tamanho da janela (distância entre a primeira e a última sentença) para o qual atribuiu-se os valores 20, 40 e 60. O segundo parâmetro, o passo (distância que a janela desliza), atribuiu-se os valores 3, 6, 9 e 12. Gerando ao final 18 modelos.

O *C99* permite ajustarmos três parâmetros, sendo, a quantidade segmentos desejados, o qual é calculado como uma proporção dos candidatos a limite. Para isso atribuiu-se as proporções de 0,2 a 1,0 em intervalos de 0,2 O segundo parâmetro, o tamanho da máscara utilizada para gerar a matriz de ranking, atribuiu-se os valores 9 e 11. 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. Gerando ao final 20 modelos.

Pela comparação dos resultados com a segmentação fornecida pelos especialistas, calculou-se para cada modelo as medidas tradicionais acurácia, precisão, revocação, F-medida. Além dessas, computou-se também as métricas mais aplicadas à segmentação textual $P_k$ e *WindowDiff*.

Em seguida aplicou-se o teste de Friedman a fim de saber se há diferenças significativas entre a eficácia dos modelos, o pós-teste de Nemenyi foi aplicado para descobrir quais diferenças são significativas. Exite diferença quando seus *rankings* médios diferirem em um valor mínimo, chamado de diferença critica (CD).

Com isso foi possível, pela análise do diagrama de diferença crítica, verificar qual é o melhor modelo para cada medida em relação aos demais.

A tabela I mostra os dados obtidos com o *C99*, onde `S` é a proporção de segmentos em relação a quantidade de candidatos, `M` é o tamanho da máscara utilizada para criar a matriz de *ranking* e `W` indica se os segmentos são representados por vetores contendo a frequência ou um peso das palavras.

A tabelas II mostra os dados obtidos com o *TextTiling*, `J` é o tamanho da janela e `P` e o passo.

Uma vez sabendo quais valores de parâmetros melhor configuram um algoritmo para uma medida, resta então saber qual

| Medida | S | M | W | **Média** |
|---|---|---|---|---|
| Acuracy | 40 | 11 | Sim | 0.6199 |
| F1 | 60 | 9 | Sim | 0.6167 |
| Precision | 40 | 11 | Sim | 0.7106 |
| Recall | 100 | 9 | Não | 0.8516 |
| Pk | 40 | 11 | Sim | 0.1163 |
| Windiff | 40 | 11 | Sim | 0.3800 |

Tabela I: Médias das medidas obtidas com *C99*

| Medida | J | P | **Média** |
|---|---|---|---|
| Acuracy | 50 | 9 | 0.5510 |
| F1 | 50 | 3 | 0.5898 |
| Precision | 60 | 12 | 0.5746 |
| Recall | 50 | 3 | 0.7717 |
| Pk | 30 | 9 | 0.1572 |
| Windiff | 50 | 9 | 0.4489 |

Tabela II: Médias das medidas obtidas com o *TextTiling*

dos dois algoritmos é mais eficiente segundo essa medida. Para isso aplicou-se novamente o teste de Friedman com pós-teste de Nemenyi, dessa vez, com os melhores modelos dos dois algoritmos para cada medida. O resultado segue na Tabela III

| Medida | Algoritmo | S | M | W |
|---|---|---|---|---|
| Acuracy | C99 | 40 | 11 | Sim |
| Precision | C99 | 40 | 11 | Sim |
| Pk | C99 | 40 | 11 | Sim |
| Windiff | C99 | 40 | 11 | Sim |
| F1 | C99 | 60 | 9 | Sim |
| Recall | C99 | 100 | 9 | Não |

Tabela III: Melhores modelos para cada medida segundo diagramas de diferença crítica

Na análise do diagrama de diferença crítica verificou-se que o algoritmo *C99* apresenta melhor eficiência em todas as medidas e os valores das quatro primeiras os valores de S, M e W se repetiram, sugerindo uma configuração otimizada para o problema da segmentação de atas de reunião.

## VI. Análise dos Resultados

## VII. Conclusão

## Referências


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

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

[3] B. K. Boguraev and M. S. Neff. 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*, 2000.

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

[5] M. H. Bokaei, H. Sameti, and Y. Liu. 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, Nov. 2015.

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

[7] F. Y. Y. Choi. 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, 2000. Association for Computational Linguistics.

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

[9] M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. 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, 2003. Association for Computational Linguistics.

[10] M. A. Hearst. 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, 1994. Association for Computational Linguistics.

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

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

[13] J. C. Reynar. *Topic Segmentation: Algorithms and Applications*. PhD thesis, Philadelphia, PA, USA, 1998. AAI9829978.

[14] J. C. Reynar. 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, 1999. Association for Computational Linguistics.