Segmentação topical automática de atas de reunião

Ovídio José Francisco
ovidiojf@gmail.com

RESUMO

Keywords

1. INTRODUÇÃO

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 quebras de assunto, ou seja, não apresentam parágrafos, seções ou capítulos, como transcrições automáticas de áudio e grandes documentos que contêm assuntos não idênticos como atas de reunião e noticias.

O interesse por segmentação textual tem crescido em em aplicações voltadas a recuperação de informação e sumarização de textos. Essa técnica pode ser usada para aprimorar o acesso a informação quando essa é solicitada por um usuário por meio de uma consulta, onde é possível oferecer porções menores de texto mais relevante ao invés de exibir um documento maior que pode conter informações menos pertinente. A sumarização de texto também pode ser aprimorada ao processar segmentos separados por tópicos ao invés de documentos inteiros.

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.

2. TRABALHOS RELACIONADOS

Os principais algoritmos de segmentação textual baseiam-se na ideia de coesão léxica entre assuntos. Isto é, a mudança de tópicos é acompanhada de uma proporcional mudança de vocabulário. A partir disso, vários algoritmos foram propostos. Dessa forma, assumem o pressuposto que um segmento pode ser identificado e delimitado pela análise das palavras que o compõe.

Uma vez que 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* [4] 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 [3, 5].

Choi [3] 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

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

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.

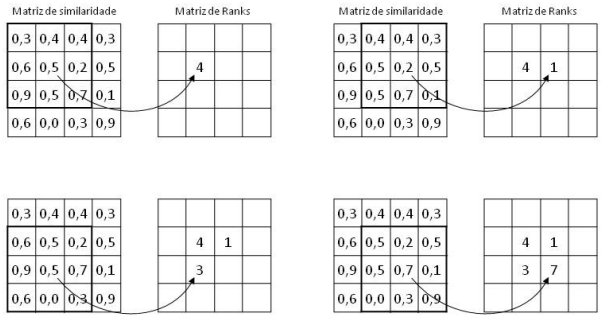

Figure 1: Exemplo de construção de uma matriz de rank

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

Finalmente, na etapa de *clustering*, Choi utiliza um método baseado no algoritmo de maximização de Reynar [**?**] para identificar os limites entre os segmentos.

Semelhante a esse trabalho, outras abordagens foram propostas como ...

[1] faz uma adaptação do *TextTiling* ao contexto das conversas em reuniões com múltiplos participantes.

3. ADAPTAÇÃO ÀS ATAS DE REUNIÃO

Os algoritmos *TextTiling* e *C99* foram propostos para o inglês e sem domínio determinado, 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 4 mostra a análise dos algoritmos adaptados.

3.1 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.

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.

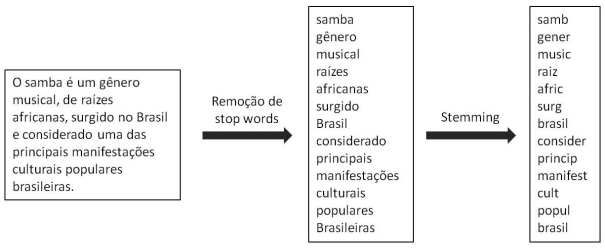

Figure 2: Exemplo de pré-processamento

3.1.1 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.

3.2 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, e quebras de parágrafo são usados para formatação de outros elementos como espaço para assinaturas. Indicar todo *token* como ponto candidato obriga a ajustá-lo de maneira a não segmentar 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.

4. AVALIAÇÃO

Definir o que é um bom algoritmo de segmentação
avaliação todos precisam de um gold text

1 - Concatenação 2 - Juízes concordam ou não 3 - Mediador na reunião 4 - Não avaliar o segmentador e sim o resultado da aplicação final. 5 - Consultar o autor do texto

De acordo com [**?**] há duas principais dificuldades na avaliação de segmentadores automáticos. A primeira é conseguir um referência confiável de texto segmentado, ou seja, uma segmentação ideal, já que juízes humanos costumam não concordar entre si, sobre onde os limites estão. A segunda é que tipos diferentes de erros devem ter pesos diferentes de acordo com a aplicação. Há casos onde certa imprecisão é

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
```

tolerável e outras como a segmentação de notícias, onde a precisão é mais importante.

Para contornar essas dificuldades, algumas abordagens podem ser utilizadas. Algumas autores preferem detectar a segmentação em textos formados pela concatenação de documentos distintos, para que não haja diferenças subjetivas [**?**]. Há ainda outros que não avaliam o algoritmo diretamente, mas seu impacto na aplicação final[**?**, **?**, **?**]. Outras abordagens apenas atribuiem um segmento cada quebra de parágrafo [**?**]

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.

4.1 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 F1 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 [5].

A Figura **??** 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.

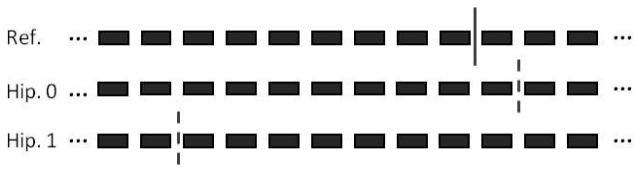

Figure 3: Exmplos de segmentação

4.1.1 P_k

A fim de resolver o problema de *near misses*, Beeferman *at 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 inicio e o final da janela estão dentro do segmento esperado e penaliza o algoritmo em caso de discrepância.

4.1.2 *WindowDiff*

Pevzner [6] aponta problemas na avaliação mais tradicional Pk [2]. Eles apontam que esse métrica penaliza demasiadamente os falsos negativos em relação aos falsos positivos, desconsidera no o tamanho e a quantidade de segmentos, entre outros apontamentos.

Como solução, propõe duas alterações principais. Dobra a penalidade para os falsos positivos a fim de diminuir o problema da subestimação dessa medida e

ao mover a 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 de texto.

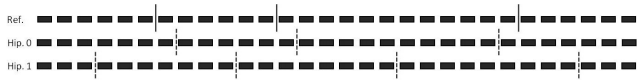

Figure 4: Exemplo de construção de uma matriz de rank

5. ANÁLISE DOS RESULTADOS

6. CONCLUSÃO

7. REFERENCES

[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] 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.

[4] 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.

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

[6] 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.