Ovídio José Francisco

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

Sorocaba, SP

12 de fevereiro de 2018

Lista de símbolos

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

Sumário

1 Introdução

2 Conceituação Teórica

A popularidade dos computares permite a criação e compartilhamento de textos onde a quantidade de informação facilmente extrapola a capacidade de humana de leitura e análise de coleções de documentos, estejam eles disponíveis na Internet ou em computadores pessoais. A necessidade de simplificar e organizar grandes coleções de documentos criou uma demanda por modelos de aprendizado de máquina para extração de conhecimento em bases textuais. Para esse fim, foram desenvolvidas técnicas para descobrir, extrair e agrupar textos de grandes coleções, entre essas, a modelagem de tópicos (HOFMANN, 1999; DEERWESTER et al., 1990; LEE; SEUNG, 1999; BLEI, 2012).

2.1 Segmentação Textual

A tarefa de segmentação textual consiste em dividir um texto em partes ou segmentos que contenham um significado relativamente independente. Em outras palavras, é identificar as posições nas quais há uma mudança significativa de assuntos. As técnicas de segmentação textual consideram um texto como uma sequência linear de unidades de informação que podem ser, por exemplo, cada termo presente no texto, os parágrafos ou as sentenças. Cada unidade de informação é um elemento do texto que não será dividido no processo de segmentação e cada ponto entre duas unidades é considerado um candidato a limite entre segmentos. Nesse sentido, um segmento pode ser visto como uma sucessão de unidades de informação que compartilham o mesmo assunto.

Os primeiros trabalhos dessa área se apoiam na ideia de que a mudança de assunto em um texto é acompanhada de uma proporcional mudança de vocabulário. Essa ideia, chamada de coesão léxica, sugere que a distribuição das palavras é um forte indicador da estrutura do texto, e demonstrou-se que há uma estreita correlação entre quedas na coesão léxica em janelas de texto e a transição de assuntos (KOZIMA, 1993). Em seu trabalho, Kozima calculou a coesão léxica de uma janela de palavras usando *spreading activation* em uma rede semântica especialmente elaborada para o idioma Inglês. Contudo, a implementação de um algoritmo para outros domínios depende da construção de uma rede adequada.

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

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

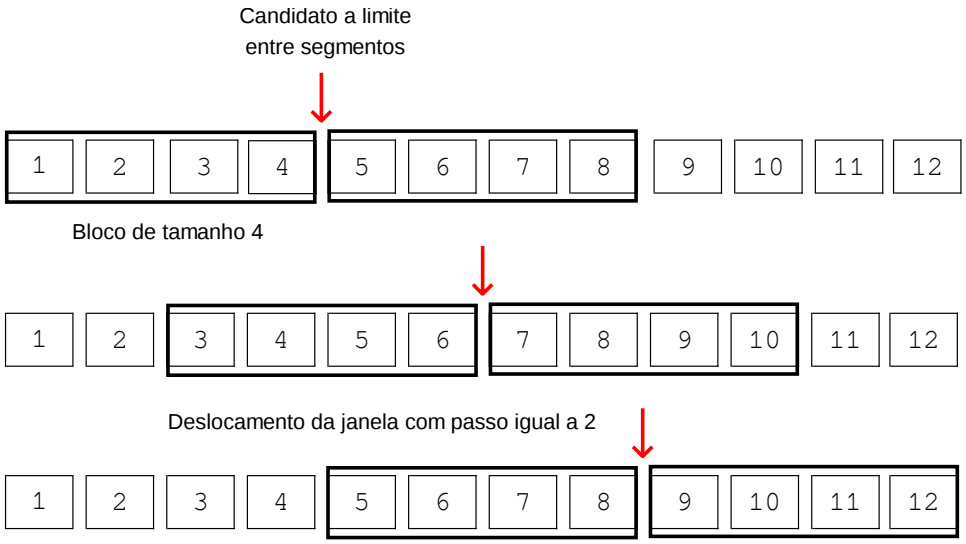

Figura 1 – Processo de deslocamento da janela deslizante. Os quadrados numerados representam as sentenças e os retângulos representam os blocos de texto a serem comparados. O deslocamento movimenta o candidato a limite e por consequência os blocos que o antecede e sucede.

Em seguida, os blocos de texto são representados por vetores que contém as frequências de suas palavras. Diferente da proposta de Kozima, o *TextTiling* utiliza cosseno (Equação **??**) como medida para a similaridade entre os blocos adjacentes. Um limite ou transição entre segmentos é identificado sempre que a similaridade entre as unidades que antecedem e precedem o ponto candidato cai abaixo de um limiar, indicando uma diminuição da similaridade entre os blocos adjacentes. Ou seja, identifica-se uma transição entre segmentos pelos vales na curva de dissimilaridades. Para cada final de sentença representada por c_i atribui-se uma profundidade dada por $(c_{i-1} - c_i) + (c_{i+1} - c_i)$ e será um limite entre segmentos caso a profundidade exceda $\overline{s} - \sigma$, onde $\overline{s}$ é a média da profundidade de todos os vales do documento e σ, o desvio padrão. Na Figura 2 é ilustrado a curva de dissimilaridade entre os blocos adjacentes.

O TextTiling apresenta como vantagens a facilidade de implementação e baixa complexidade computacional, favorecendo a implementação de trabalhos similares (NAILI;

CHAIBI; GHEZALA, 2016; BOKAEI; SAMETI; LIU, 2015; CHAIBI; NAILI; SAMMOUD, 2014; KERN; GRANITZER, 2009; GALLEY et al., 2003), e sua utilização como base line em outros trabalhos (CARDOSO; PARDO; TABOADA, 2017; DIAS; ALVES; LOPES, 2007). Por outro lado, algoritmos mais complexos, como os baseados em matrizes de similaridade, apresentam acurácia relativamente superior como apresentado em (CHOI, 2000; KERN; GRANITZER, 2009; MISRA et al., 2009).

Outro algoritmo frequentemente referenciado na literatura é o C99 (CHOI, 2000) o qual é baseado em uma matriz de *ranking* das similaridades. A utilização de da coesão léxica pode não ser confiável para segmentos pequenos nessa abordagem, pois a ocorrência adicional de uma palavra pode causar certo impacto e alterar o cálculo da similaridade. Além disso, o estilo da escrita normalmente não é constante em todo o texto. Por exemplo, textos iniciais dedicados a introdução costumam apresentar menor coesão do que trechos dedicados a um tópico específico. Portanto, comparar a similaridade entre trechos de diferentes regiões não é apropriado. Devido a isso, as similaridades não podem ser comparadas em valores absolutos. Contorna-se esse problema fazendo uso de matrizes de similaridade para encontrar os segmentos de texto. Para isso, o *C99* constrói uma matriz que contém as similaridades de todas as unidades de informação (normalmente sentenças ou parágrafos).

Na Figura 3 é mostrado um exemplo de uma matriz de similaridade onde a intensidade do ponto(i, j) representa a similaridade entre as sentenças i e j. Observa-se que a matriz é simétrica, assim cada ponto na linha diagonal representa a similaridade quando $i = j$ (ou seja, com a mesma sentença) e revela quadrados com maior concentração de pontos ao longo da diagonal. A concentração de pontos ao longo da diagonal indica porções de texto com maior coesão léxica.

Em seguida, cada valor na matriz de similaridade é substituído por seu *ranking* local. Para cada elemento da matriz, seu *ranking* será o número de elementos vizinhos com valor de similaridade menor que o seu. Assim, para cada elemento determina-se uma região quadrada de tamanho n em que o elemento em questão será comparado com $n \times n - 1$ elementos vizinhos. Na Figura 2.1 é destacado um quadro 3 x 3 de uma matriz. Tomando como exemplo o elemento com valor $0,5$, a mesma posição na matriz de *rankings* terá o valor 4, pois esse é o número de vizinhos com valores inferiores a $0,5$ dentro do quadro analisado na matriz de similaridades. Da mesma forma, na Figura 2.1 para o valor $0,2$ a matriz de *rankings* conterá o valor 1 na mesma posição. Após a construção da matriz de ranking obtêm-se um maior contraste entre os pontos facilitando a detecção de limites quando a queda de similaridade entre sentenças é mais sutil.

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

da matriz. Calcula-se a densidade dessa região como mostrado na Equação 2.1. Seja $s_{i,j}$ a somatória dos *rakings* de um segmento e $a_{i,j}$ sua área interior. Seja $B = \{b1, ..., b_m\}$ a lista de m segmentos e s_k e a_k são a somatória dos valores dos rankings e a área de um segmento k em B. Então, a densidade é computada por:

$$D = \frac{\sum_{k=1}^{m} s_k}{\sum_{k=1}^{m} a_k} \tag{2.1}$$

O processo incia com um único segmento formado por todas as sentenças do documento e o divide recursivamente em m segmentos. Cada passo divide um dos segmentos em B no ponto (i, j) que maximiza D (Equação 2.1). O processo se repte até atingir o número de segmentos desejados ou um limiar de similaridade.

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

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

2.1.1 Medidas de Avaliação em Segmentação Textual

As medidas de avaliação tradicionais como precisão e revocação permitem medir o desempenho de modelos de Recuperação de Informação e Aprendizado de Máquina por meio da comparação dos valores produzidos pelo modelo com os valores observados em uma referência. Usa-se uma tabela, chamada matriz de confusão, para visualizar o desempenho de um algoritmo. Na Tabela 1 é apresentada uma matriz de confusão para duas classes (Positivo e Negativo).

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

Tabela 1 – Matriz de confusão.

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

Nesse sentido, a precisão indica a proporção de limites corretamente identificados pelo algoritmo, ou seja, correspondem a um limite real na segmentação de referência. Porém, não diz nada sobre quantos limites reais existem. É calculada dividindo-se o número de limites identificados automaticamente pelo número de candidatos a limite (Equação 2.2).

$$Precisão = \frac{VP}{VP + FP} \tag{2.2}$$

A revocação, é a proporção de limites verdadeiros que foram identificados pelo algoritmo. Porém não diz nada sobre quantos limites foram identificados incorretamente. É calculada dividindo-se o número de limites identificados automaticamente pelo número limites verdadeiros (Equação 2.3).

$$Revocação = \frac{VP}{VP + FN} \tag{2.3}$$

Existe uma relação inversa entre precisão e revocação. Conforme o algoritmo aponta mais segmentos no texto, este tende a melhorar a revocação e ao mesmo tempo, reduzir a precisão. Esse problema de avaliação pode ser contornado utilizado a medida F^1 que é a média harmônica entre precisão e revocação onde ambas tem o mesmo peso (Equação 2.4).

$$F^1 = \frac{2 \times Precisão \times Revocação}{Precisão + Revocação} \tag{2.4}$$

As medidas de avaliação tradicionais, precisão e revocação, podem não ser confiáveis, por não considerarem a distância entre os limites, mas penalizam o algoritmo sempre que um limite que não coincide perfeitamente com a referência. Essas medidas podem ser mais adequadas quando necessita-se de segmentações com maior exatidão. Em outras palavras, computam apenas os erros do algoritmo quando se detecta falsos positivos ou falsos negativos, o que nesse contexto de segmentação textual pode não ser suficiente, dado a subjetividade da tarefa. Além dessas medidas, que consideram apenas se um segmento foi perfeitamente definido conforme uma referência, pode-se também considerar a distância entre o segmento extraído automaticamente e o segmento de referência (KERN; GRANITZER, 2009). Chama-se *near misses* o caso em que um limite identificado automaticamente não coincide exatamente com a referência, mas é necessário considerar a proximidade entre eles.

Na Figura 5 é apresentado um exemplo com duas segmentações extraídas automaticamente e uma referência. Em ambos os casos não há nenhum verdadeiro positivo, o que implica em zero para os valores de precisão, acurácia, e revocação, embora o resultado do algoritmo A possa ser considerado superior ao primeiro se levado em conta a proximidade dos limites.

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

$$P_k(ref, hyp) = \frac{1}{N-k} \sum_{i=1}^{N-k} (\delta_{ref}(i, i+k) \ \bar{\oplus} \ \delta_{hyp}(i, i+k)) \tag{2.5}$$

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

WindowDiff (**??**) é uma medida alternativa à P_k. De maneira semelhante, move

uma janela pelo texto e penaliza o algoritmo sempre que o número de limites proposto pelo algoritmo não coincidir com o número de limites esperados para aquela janela. Ou seja, o algoritmo é penalizado quando não concordar com a segmentação de referência quanto ao número de segmentos na janela. Mais formalmente, para cada intervalo k, compara o número de segmentos obtidos pela referência r_i com o obtido pelo algoritmo a_i e penaliza o algoritmo se $r_i \neq a_i$. Na Equação 2.6 é mostrada a definição de WindowDiff onde $b(i, i+k)$ representa o número de limites entre as sentenças i e $i+k$ e N, o total de sentenças no texto.

$$WindowDiff(ref, hyp) = \frac{1}{N-k} \sum_{i=1}^{N-k} (|b(ref_i - ref_{i+k}) - b(hyp_i - hyp_{i+k})| > 0) \quad (2.6)$$

Assim, consegue manter a sensibilidade a *near misses* e além disso, considerar o tamanho das janelas. A fim de melhor equilibrar o peso dos falsos positivos em relação a *near misses*, dobra-se a penalidade para falsos positivos, evitando-se a supervalorização dessa medida.

As medidas *WindowDiff* e P_k, consideram a quantidade e proximidade entre os limites, sendo mais tolerantes a pequenas imprecisões. Essa é uma característica desejável, visto que as segmentações de referência possuem diferenças consideráveis. *WindowDiff* equilibra melhor os falsos positivos em relação a *near misses*, ao passo que P_k os penaliza com peso maior. Isso significa que segmentadores melhores avaliados em P_k ajudam a selecionar as configurações que erram menos ao separar trechos de texto com o mesmo assunto, enquanto *WindowDiff* é mais tolerante nesse aspecto. De maneira geral, observa-se melhores resultados de *WindowDiff* quando os algoritmos aproximam a quantidade de segmentos automáticos da quantidade de segmentos da referência. Por outro lado, P_k avalia melhor as configurações que retornam menos segmentos. Contudo, não é possível definir um valor adequado, uma vez que os segmentadores humanos frequentemente apontam segmentações diferentes.

Ao final do processo de segmentação, são produzidos fragmentos de documentos, aqui chamados de subdocumentos. Esses subdocumentos contém um texto, assim como no documento original, em um estágio de processamento inicial, pois ainda não estão estruturados. Ocorre que as técnicas de aprendizado de máquina exigem uma representação estruturada dos textos conforme será visto na Seção **??**.


conteudo/capitulos/figs/curva-similaridade.pdf

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

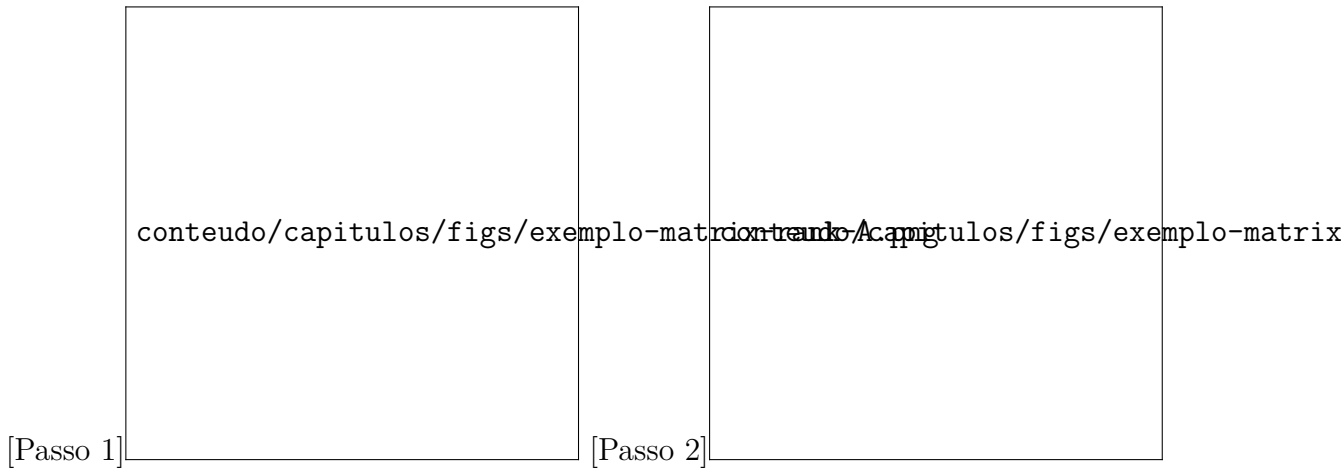

Figura 4 – Exemplo de construção de uma matriz de rankings.

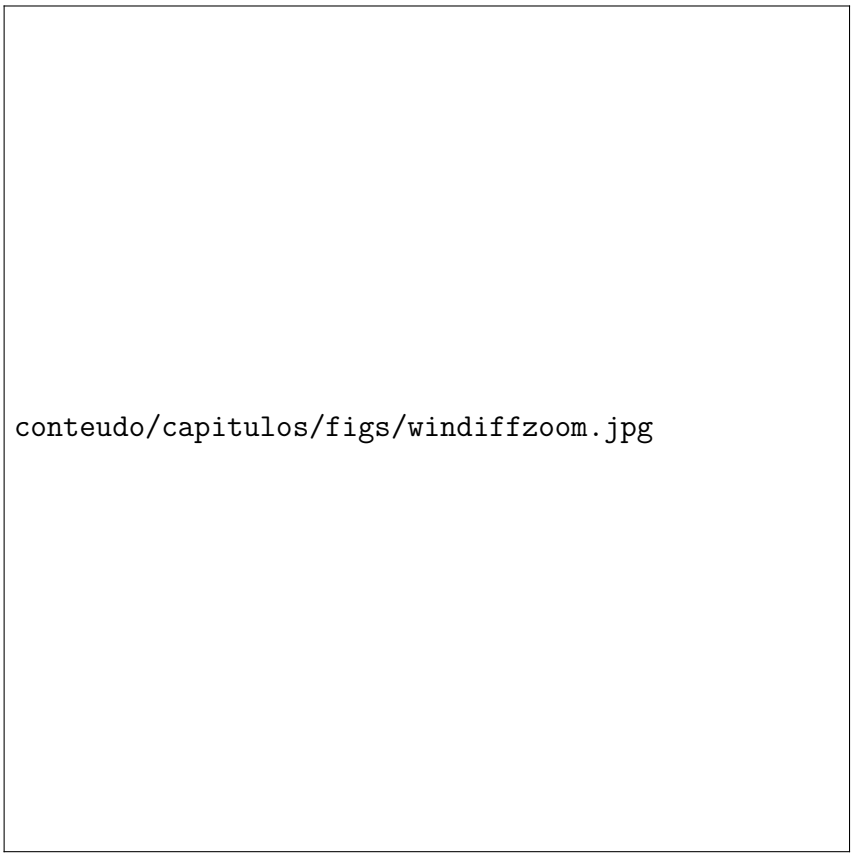

Figura 5 – Exemplos de *near missing* e falso positivo puro. Os blocos indicam uma unidade de informação e as linha verticais representam uma transição de assunto.

3 Sistema Proposto

Referências

BLEI, D. M. Probabilistic topic models. *Commun. ACM*, ACM, New York, NY, USA, v. 55, n. 4, p. 77–84, abr. 2012. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2133806.2133826>. Citado na página 7.

BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. *J. Mach. Learn. Res.*, JMLR.org, v. 3, p. 993–1022, mar. 2003. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=944919.944937>. Citado na página 10.

BOKAEI, M. H.; SAMETI, H.; LIU, Y. Linear discourse segmentation of multi-party meetings based on local and global information. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, IEEE Press, Piscataway, NJ, USA, v. 23, n. 11, p. 1879–1891, nov. 2015. ISSN 2329-9290. Disponível em: <http://dx.doi.org/10.1109/TASLP.2015.2456430>. Citado na página 9.

CARDOSO, P.; PARDO, T.; TABOADA, M. Subtopic annotation and automatic segmentation for news texts in brazilian portuguese. *Corpora*, Edinburgh University Press, v. 12, n. 1, p. 23–54, 2017. Citado na página 9.

CHAIBI, A. H.; NAILI, M.; SAMMOUD, S. Topic segmentation for textual document written in arabic language. *Procedia Computer Science*, v. 35, p. 437 – 446, 2014. ISSN 1877-0509. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050914010898>. Citado na página 9.

CHOI, F. Y. Y. Advances in domain independent linear text segmentation. In: *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000. (NAACL 2000), p. 26–33. Disponível em: <http://dl.acm.org/citation.cfm?id=974305.974309>. Citado na página 9.

DEERWESTER, S. et al. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, v. 41, n. 6, p. 391–407, 1990. Citado na página 7.

DIAS, G.; ALVES, E.; LOPES, J. G. P. Topic segmentation algorithms for text summarization and passage retrieval: An exhaustive evaluation. In: *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*. AAAI Press, 2007. (AAAI'07), p. 1334–1339. ISBN 978-1-57735-323-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1619797.1619859>. Citado na página 9.

EISENSTEIN, J.; BARZILAY, R. Bayesian unsupervised topic segmentation. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008. (EMNLP '08), p. 334–343. Disponível em: <http://dl.acm.org/citation.cfm?id=1613715.1613760>. Citado 2 vezes nas páginas 10 e 15.

GALLEY, M. et al. Discourse segmentation of multi-party conversation. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003. (ACL '03), p.

562–569. Disponível em: <http://dx.doi.org/10.3115/1075096.1075167>. Citado na página 9.

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

KERN, R.; GRANITZER, M. Efficient linear text segmentation based on information retrieval techniques. *Proceedings of the International Conference on Management of Emergent Digital EcoSystems, MEDES '09*, p. 167–171, 2009. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-74549147972&doi=10.1145%2f1643823.1643854&partnerID=40&md5=1c6f73bc0e07446fcc178440e48bbc40>. Citado 2 vezes nas páginas 9 e 12.

KOZIMA, H. Text segmentation based on similarity between words. In: *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 1993. (ACL '93), p. 286–288. Disponível em: <http://dx.doi.org/10.3115/981574.981616>. Citado na página 7.

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

MISRA, H. et al. Text segmentation via topic modeling: An analytical study. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2009. (CIKM '09), p. 1553–1556. ISBN 978-1-60558-512-3. Disponível em: <http://doi.acm.org/10.1145/1645953.1646170>. Citado na página 9.

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