Classificação Automática de Texto Buscando Similaridade de Palavras e Significados Ocultos

Fabricio S. Catae and Ricardo L. A. Rocha

Laboratório de Tecnologia Adaptativa - Escola Politécnica
Universidade de São Paulo (USP)
{fcatae,rlarocha}@usp.br

Resumo Adotamos o método da indexação da semântica latente (LSI) para classificar documentos que estejam relacionados por algum meio não restrito apenas aos termos presentes, mas buscando outras formas de similaridades. A redução de dimensionalidade da matriz Termo-Documento não é novidade, sendo normalmente adotado entre 200 a 300 dimensões. Nesse trabalho, transformamos o LSI em um algoritmo semi-supervisionado e determinamos o número ideal de dimensão durante a fase de treinamento. O algoritmo utiliza um espaço isométrico a aquele definido pela matriz Termo-Documento para acelerar os cálculos.

Keywords: clustering, information retrieval, latent semantic indexing

1 Introdução

A classificação de texto é um processo importante na organização das informações disponíveis. Além da melhoria na eficiência de busca, a categorização de documentos possui outras finalidades como, por exemplo, mineração de dados, detecção de conteúdo adulto ou inadequado, filtragem de spam, entre outros. A decisão é feita por um especialista, que considera o conteúdo do documento e define a qual categoria que ele pertence. A classificação pode ser automatizada com o intuito de melhorar a precisão e diminuir a subjetividade associada ao processo manual.

No campo de recuperação de informação, o texto do documento é frequentemente representado como um conjunto não-ordenado de palavras, constituindo um bag-of-words. Salton [15] introduziu a ideia de representação de documento através de vetores, cujas componentes são calculadas com base na contagem de palavras e associados a pesos, criando como resultado a matriz Termo-Documento. A partir dessa representação, uma forma de determinar sua categoria é identificar palavras específicas ou presentes somente naquele tipo de documento. Os termos "motor" e "piloto" aparecem frequentemente em assuntos relacionados a "carro", portanto, podem ser usados para classificação.

Uma forma mais elaborada de classificar seria calcular as probabilidades associadas às palavras e aplicar Naïve Bayes [13]. Apesar dos resultados positivos,

esse modelo assume que cada dimensão é independente das demais, fato que está longe de ser verdade. Por exemplo, quando nos referimos ao termo "carro", estamos nos referindo também aos documentos que utilizam a palavra "automóvel". As palavras possuem inter-relação e o modelo deve considerar a similaridade entre elas.

2 Fundamento Teórico

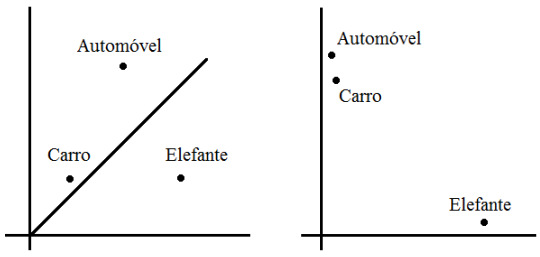
2.1 Indexação da semântica latente

Landauer buscou uma relação entre os termos e os conceitos ocultos, ou seja, uma "semântica latente" [10] que permitisse relacionar os documentos por uma similaridade não restrita somente aos termos presentes. A relação de dependência entre palavras pode ser modelada através de uma rotação do espaço, cujos vetores de base são alterados para um novo conjunto de coordenadas. Por exemplo, podemos dizer que haverá uma transformação na forma:

(a) $\text{Carro} = 1.0 * motorista + 4.0 * rodas + 2.0 * portas + 1.0 * motor$
(b) $\text{Automóvel} = 1.0 * motorista + 4.0 * rodas + 2.0 * portas + 1.0 * motor$

Apesar de "automóvel" e "carro" serem termos distintos, a soma dos componentes os torna equivalentes. A indexação da semântica latente (LSI, do inglês "Latent Semantic Indexing") projeta os vetores em um espaço de menor dimensão, também denominado de espaço semântico [6]. Palavras similares ou relacionadas seriam projetadas em uma mesma dimensão do espaço.

Por exemplo, podemos adotar k=2, que representa a matriz A projetada em um plano. Assim, os documentos ("automóvel", "carro" e "elefante") estariam automaticamente agrupados por similaridade nesse espaço de dimensão reduzida (Figura 1).



(a) Dimensão n (b) Dimensão k=2

Figura 1. (a) Os documentos estão dispersos no espaço; (b) Após a projeção dos vetores em um plano, os documentos ficam agrupados por similaridade.

Uma formalização matemática dessa proposição foi feita parcialmente por Papadimitriou [14] e apresenta indicativos de que a técnica funciona para caso geral.

2.2 Decomposição em Valores Singulares

LSI é um método baseado na álgebra linear [4]. A idéia é obter uma aproximação da matriz A Termo-Documento, determinando os vetores singulares mais significativos da matriz.

Dada uma matriz A de tamanho m x n, podemos decompor na forma:

$$A = U\Sigma V^T \tag{1}$$

Onde U e V^T são matrizes ortonormais, ou seja, $U^TU = V^TV = I_n$. A matriz Σ é uma matriz diagonal $\Sigma = diag(\sigma_1, \cdots, \sigma_n)$. A decomposição matricial é única quando os valores singulares σ_i estão em ordem decrescente.

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$$

Dizemos que a matriz A tem posto r ou $rank(A) = r$ quando:

$$\begin{cases} \sigma_k > 0 \text{ para } 1 \leq k \leq r \\ \sigma_{r+1} = \cdots = \sigma_n = 0 \end{cases}$$

Podemos encontrar uma matriz aproximada A' truncando a matriz Σ com somente os k maiores valores singulares e zerando os demais valores. Sejam U', V' e Σ' as matrizes truncadas de U, V e Σ. Assim, temos que:

$$A' = U'\Sigma'V'^T \tag{2}$$

Se $k < r$, então a matriz A' será uma aproximação de A. Pelo Teorema de Eckart-Young [2], a matriz A' é a melhor aproximação matricial de posto k. Em outras palavras, dada uma matriz M qualquer, tal que $rank(M) = k$, então calculamos o erro mínimo usando a norma de Frobenius:

$$||A - M||^2 \geq ||A - A'||^2 = \sigma_{k+1}^2 + \cdots + \sigma_n^2 \tag{3}$$

2.3 Parâmetro k

Deerwester [4] realizou testes com o conjunto de dados MED composto por documentos médicos, variando o número de dimensões entre 10 e 100. Ao final, reportou que o melhor desempenho ocorreu com 100 dimensões.

Segundo os relatórios de acompanhamento da aplicação do método LSI no Text Retrieval Conference (TREC), o número de dimensão afeta diretamente no número de documentos recuperados e na precisão [7][8][9]. Na primeira edição da conferência, foram utilizadas 235 e 250 dimensões, enquanto que, no TREC-2, empregaram-se 199 e 204 dimensões. Na versão seguinte, TREC-3, o experimento rodou com maior variação do parâmetro k: 199, 250, 300 e 346 dimensões.

Diminuir o número de dimensões, parâmetro k, aumenta a quantidade de documentos recuperados em uma pesquisa. Em contrapartida, espaços com dimensão excessivamente reduzida apresentam problemas de precisão. Ajustar adequadamente o número de dimensões é um desafio. Dumais [5] sugere a utilização

de um espaço composto por 200 a 300 valores singulares, observando que, a partir de um determinado valor, o aumento do valor do parâmetro k passa a degradar o resultado.

Landauer apresentou o efeito da variação no número de dimensões através de um experimento usando 80 questões de uma prova de inglês, ETS Test of English as a Foreign Language (TOEFL) [11]. O corpus adotado foi composto por jornais disponibilizados pela Associated Press, a enciclopédia Grolier's Academic American e textos de conteúdo infantil. O parâmetro k foi testado entre 2 a 1032, obtendo a melhor taxa de acertos com 300 e 325 dimensões.

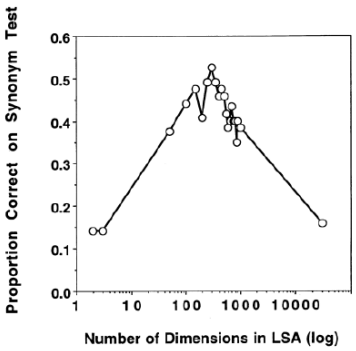


Figura 2. A influência do parâmetro k no experimento de sinônimos usando a prova TOEFL. Esse gráfico ressalta a importância da escolha adequada do número de dimensões.

3 Proposta

O objetivo desse trabalho é dimensionar o parâmetro k, que determina o número de dimensões usadas no LSI. O número de dimensão afeta diretamente a precisão e a quantidade de documentos recuperados. Apesar de ser um fator importante, não foi encontrada nenhuma referência sobre como determinar o valor ideal para a tarefa de categorização de documentos. [3]

LSI é um algoritmo de aprendizado não-supervisionado. Durante a fase de treinamento, todos os documentos são considerados: conjunto de treinamento e testes. Através da decomposição em valores singulares, encontramos um conjunto de vetores unitários e ortogonais, que vão compor a base do subespaço representado. Em nenhum momento, a categoria é levada em consideração.

Podemos transformar o LSI em um algoritmo semi-supervisionado. Durante a fase de inicial, variamos o parâmetro k e calculamos a taxa de acerto correspondente aos documentos de treinamento. Traçamos, assim, a curva $T(k)$ que

representa a taxa de acerto conforme variamos o número de dimensões. Ao final da fase de treinamento, selecionamos o valor k que maximiza o número de acertos $T(k)$.

Os dados de treinamento serão usados, portanto, para calibrar o parâmetro k. Assumindo que os conjuntos de teste e treinamento sejam aleatórios, podemos inferir que o modelo trará resultados semelhantes a ambos. Esperamos observar $T(k) = P(k)$, onde $P(k)$ corresponde ao desempenho do modelo contra os dados finais de teste.

4 Experimento

Os termos extraídos do corpus e os documentos são transformados em vetores usando pesos TF-IDF. Apesar da literatura sugerir melhorias, não foi considerado nenhum processo de tratamento de palavras como lista de exclusão (stop words), normalização morfológica (stemming), identificação de termos compostos, correção sintática ou uso de dicionários. Termos com menos de 3 caracteres ou menos de 10 ocorrências foram desconsiderados.

Os vetores resultantes do conjunto de treinamento e testes são colocados em uma matriz Termo-Documento de tamanho m x n. Durante os experimentos, observamos que a condição $m > n$ foi mantida. A decomposição em valores singulares foi realizada por meio da biblioteca SVDPACKC [1]. Como resultado, obtivemos as matrizes U, S e V. Os vetores singulares foram, então, truncados e o espaço, reduzido a somente k dimensões. Assim chegamos às matrizes U', S', V' e a matriz A' de posto k: $A' = U'S'V'^T$

A' é uma aproximação da matriz Termo-Documento e será usada para predição. Ainda na fase inicial, fazemos uma prévia da classificação usando o cálculo da similaridade com os documentos de treinamento. A determinação da categoria é feita através do método do vizinho mais próximo (kNN com k=1). A similaridade é dada pelo cosseno do ângulo formado pelos vetores dos documentos correspondentes.

Ao repetir o processo variando o número de dimensão k entre 2 e 1000 (adotamos esse valor arbitrário como máximo devido ao requisito de grande quantidade de memória do computador), obtivemos a curva $T(k)$: acertos x dimensão, com base nos dados de treinamento. No final, selecionamos o parâmetro k igual ao pico máximo da curva $T(k)$ e a fase de treinamento do modelo é encerrada

Com o objetivo de medir a precisão do método, calculamos a taxa de acerto P associada ao conjunto de teste. Se utilizarmos o método LSI variando o número de dimensões k, então traçamos uma nova curva $P(k)$ e esperamos que ela seja igual a $T(k)$. A única diferença entre elas é a origem dos dados: T é baseado nos dados de treinamento e P, nos dados de teste final. Analisamos a razão entre $P(k)$ e $T(k)$ em único gráfico, pois esperamos que a relação entre as funções seja uma constante.

5 Dados

Utilizamos o corpus Reuters 21578 [12] com textos originais da Reuters de 1987 manualmente classificados por especialistas em 135 categorias. Na separação padrão conhecida como "modApté", alguns documentos estão classificados em nenhum ou em múltiplos assuntos. Nesse trabalho, consideramos somente os documentos com uma única categoria.

6 Otimização do Algoritmo Proposto

A implementação mais simples do algoritmo pode ser n vezes mais demorado do que o LSI padrão. Durante a iteração de $k = 2, 3, \ldots, n$, realizamos sempre os cálculos de decomposição em valores singulares (SVD) e cosseno, tornando o processo lento. Notamos a presença de cálculos repetitivos que, portanto, poderiam ser convertidos em um processo incremental. A seguir, descrevemos as otimizações empregadas.

6.1 Cálculo Único do SVD

A decomposição SVD utiliza o método iterativo de Lanczos implementada pela biblioteca SVDPACKC. Nesse momento, manteremos a precisão numérica ao mesmo tempo que evitamos alterar o algoritmo empregado.

A cada iteração $k = 2, 3, 4, \ldots, n$ realizamos a decomposição SVD da matriz, repetindo o cálculo da multiplicação AA^T para encontrar os valores singulares. Modificamos o processo para realizar uma única chamada a biblioteca SVDPACKC, obtendo a decomposição matricial com o maior número de valores singulares. O resultado foi mantido em memória. A cada iteração, a decomposição de matriz era refeita truncando-se os k maiores valores singulares sem a necessidade de chamar novamente o SVDPACKC.

Apesar de ser uma alteração simples, o tempo economizado foi significativo.

6.2 Rotação da Matriz Termo Documento

A importância do SVD é conseguir encontrar a matriz aproximada $A' = U'\Sigma' {V'}^T$, permitindo calcular a similaridade de documentos através de:

$$similar(i,j) = cos(d'_i, d'_j) \quad \text{, onde } d'_i \text{ e } d'_j \text{ são colunas de } A' \tag{4}$$

Seja $B = \Sigma' {V'}^T$. Por substituição, temos que:

$$\begin{aligned} A' &= U' \Sigma' {V'}^T \\ &= U'B \end{aligned}$$

Como U' é uma matriz ortonormal, então B é uma rotação espacial de A', ou seja, os espaços são isométricos. Logo, a distância entre pontos, o produto escalar e o cosseno se mantém após a transformação. Assim, temos que:

$$similar(i,j) = cos(b_i, b_j) \quad \text{, onde } b_i \text{ e } b_j \text{ são colunas de } B \tag{5}$$

A otimização corresponde a considerar o espaço $B = \Sigma' V'^T$ ao invés de $A' = U'\Sigma' V'^T$. Além de economizar uma multiplicação matricial, o cálculo de $\Sigma' V'^T$ é rápido, pois Σ' é uma matriz diagonal.

6.3 Cálculo Incremental da Similaridade

Adotamos o espaço vetorial B^k truncado em k dimensões. Durante a fase de treinamento, variamos o valor de k entre 1 e n com o objetivo de traçar a curva $T(k)$ e determinar o pico máximo. A cada iteração de k, determinamos a similaridade dentre todos os m documentos.

$$similar(i,j,k) = \cos(b_i^k, b_j^k) = \frac{b_i^k . b_j^k}{||b_i^k||.||b_j^k||}$$

Considerando a representação dos documentos $b_i^k = (b_{i1}, b_{i2}, ..., b_{ik})$, então podemos definir as funções de produto escalar e o quadrado do módulo.

$$Prod(i,j,k) = b_i^k \cdot b_j^k = \sum_t^k b_{it} b_{jt} \tag{6}$$

$$Mods(i,k) = ||b_i^k||^2 = \sum_t^k (b_{it})^2 \tag{7}$$

Assim, temos que:

$$similar(i,j,k) = \frac{Prod(i,j,k))}{\sqrt{Mods(i,k).Mods(j,k)}} \tag{8}$$

Cada operação de $Prod$ e $Mods$ requer n operações (considerando somente multiplicação, divisão e raiz quadrada). O cálculo de similaridade requer, portanto, um total de $3n + 3$ operações.

O ideal seria calcular a similaridade de forma incremental conforme se aumentam o número de dimensões. Se transformarmos $Prod$ e $Mods$ em funções recursivas, então é possível eliminar o somatório de produtos:

$$Prod(i,j,k) = Prod(i,j,k-1) + b_{ik}.b_{jk} \tag{9}$$

$$Mods(i,k) = Mods(i,k-1) + b_{ik}.b_{ik} \tag{10}$$

A cada iteração, realizam-se apenas 5 operações (3 multiplicações, 1 divisão e 1 raiz quadrada). Essa é uma redução significativa de operações em relação a $3n + 3$, visto que, nos experimentos, adotamos $n = 1000$ dimensões.

7 Resultados

7.1 Tempo

O experimento consistiu em comparar o impacto da otimização proposta em relação a implementação mais simples do algoritmo. Variamos o número de dimensão k de 10 a 400, obtendo o resultado apresentado na Tabela 1.

Tabela 1. Comparativo de Tempo (medida em segundos)

Método	k=10	k=50	k=100	k=200	k=400
Simples	2919	14063	27071	54695	117169
Otimizado	17	58	116	265	544

O resultado chama a atenção pelo alto tempo gasto com o método de determinação do parâmetro k da forma simples. É praticamente inviável realizar o processamento dessa forma sem nenhum tipo de otimização.

7.2 Precisão

Quando se utiliza o método do LSI, o parâmetro k é normalmente ajustado para um valor entre 100 e 300. Com base nos dados utilizados, o resultado apresentaria uma precisão próxima a 0,87 a 0,88 (Tabela 2).

Tabela 2. Precisão do LSI (k dimensões)

Dimensões	Precisão
k=50	0.81
k=100	0.87
k=200	0.88
k=300	0.88
k=400	0.87
k=500	0.87
k=1000	0.85
k=2000	0.84

Utilizamos os dados de treinamento para traçar uma curva $T(k)$ variando o número de dimensões k. Ao identificar o pico máximo dessa curva (Figura 3), chegamos ao valor ideal de $k = 201$. Observamos que o máximo ocorre entre 200 e 400. Portanto, o experimento determinou corretamente um valor de k dentro da faixa esperada. Observamos que a variação de $P(k)$ em relação a $T(k)$ foi inferior a 5%, tornando a hipótese $T(x) \approx P(x)$ válida.

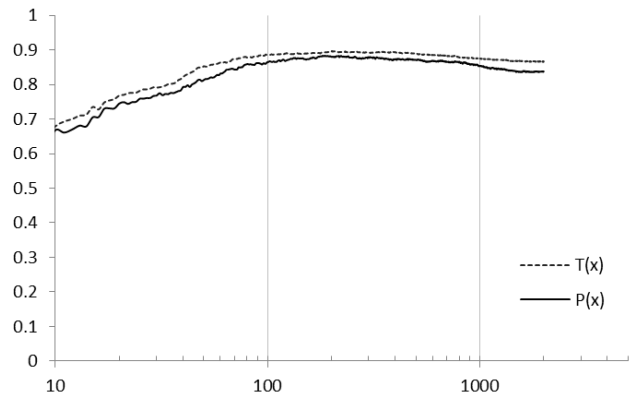


Figura 3. Gráfico da precisão em relação ao número de dimensões utilizadas (escala logarítmica).

8 Conclusão

Nesse trabalho, apresentamos o método do LSI como uma forma de classificação automática de textos através da similaridade de palavras. Ainda que não haja co-ocorrência de termos entre dois documentos, é possível classificá-los dentro da mesma categoria por meio da "semântica latente" associada.

A escolha do número de dimensão k afeta diretamente no desempenho do classificador. Utilizar valores abaixo ou acima do ideal impactam no resultado final. A proposta do trabalho foi utilizar os dados de treinamento para apoiar na decisão do k ideal, transformando o LSI em um algoritmo semi-supervisionado

No experimento, traçamos as curvas $T(k)$ e $P(k)$ correspondentes a precisão do classificador em relação aos dados de treinamento e teste respectivamente. Observamos que a diferença entre elas era inferior a 5%, concluindo assim que é possível determinar o parâmetro k apenas com os dados de treinamento.

Ao longo do trabalho, apresentamos também formas de otimizar o algoritmo proposto para que rodasse dentro de um tempo aceitável: cálculo único do SVD, rotação da matriz Termo-Documento e cálculo incremental da similaridade.

A intenção do trabalho futuro será investigar formas iterativas para o cálculo da Decomposição de Valores Singulares (SVD) ao mesmo tempo que o parâmetro k é determinado. Dessa forma, o calculo parcial do SVD da matriz consumiria menos tempo de processamento em relação ao cálculo completo.

Referências

1. Berry, M., Do, T., Krishna, V., Varadhan, S.: Svdpackc (version 1.0) user's guide (1993)
2. Berry, M., Dumais, S.T.: Using linear algebra for intelligent information retrieval. SIAM Review 37, 573–595 (1995)

3. Bradford, R.B.: An empirical study of required dimensionality for large-scale latent semantic indexing applications. In: International Conference on Information and Knowledge Management. pp. 153–162 (2008)
4. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391–407 (1990)
5. Dumais, S.T.: Improving the retrieval of information from external sources. Behavior Research Methods, Instruments and Computers 23(2), 229–236 (1991)
6. Dumais, S.T., Furnas, G.W., Landauer, T.K., Deerwester, S., Harshman, R.: Using latent semantic analysis to improve access to textual information. In: Computer Human Interaction. pp. 281–285 (1988)
7. Dumais, S.T.: Lsi meets trec: A status report. In: In: D. Harman (Ed.), The First Text REtrieval Conference (TREC1), National Institute of Standards and Technology Special Publication. pp. 137–152 (1993)
8. Dumais, S.T.: Latent semantic indexing (lsi) and trec-2. In: The Second Text REtrieval Conference (TREC-2. pp. 105–115 (1994)
9. Dumais, S.T.: Latent semantic indexing (lsi): Trec-3 report. In: Overview of the Third Text REtrieval Conference. pp. 219–230 (1995)
10. Landauer, T.K., Dumais, S.T.: A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. Psychological Review 104, 211–240 (1997)
11. Landauer, T.K., Foltz, P.W., Laham, D.: Introduction to Latent Semantic Analysis (1998)
12. Lewis, D.D.: Reuters-21578 text categorization test collection (1997), `http://www.daviddlewis.com/resources/testcollections/reuters21578`
13. Lewis, D.D.: Naive (bayes) at forty: The independence assumption in information retrieval. In: Proceedings of ECML-98, 10th European Conference on Machine Learning. pp. 4–15. Springer Verlag (1998)
14. Papadimitriou, C.H., Tamaki, H., Raghavan, P., Vempala, S.: Latent semantic indexing: a probabilistic analysis. In: Symposium on Principles of Database Systems. pp. 159–168 (1998)
15. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18, 613–620 (November 1975), `http://doi.acm.org/10.1145/361219.361220`