Acelerando o pré-processamento de consultas em sistemas de classificação automática de documentos usando aprendizado postergado

Paulo Henrique da Silva, Wellington Santos Martins

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Alameda Palmeiras, Quadra D, Campus Samambaia
CEP 74690-900 – Goiânia – GO – Brasil

{paulohsilva, wellington}@inf.ufg.br

Abstract. *Despite all efforts, automatic document classification is still a challenging task. This is because learning algorithms must deal with sparse data, noise and ambiguity inherent in human language, asymmetry of data distribution, among other factors. Some studies advocate the use of lazy learning in which only characteristics related to the document being classified are taken into account. A local and custom model is learned for each document being classified. This approach requires a high computational cost since it must operate at the moment of the classification. This work proposes the use of parallelism to enable filtering operations and feature engineering in a step prior to document classification.*

Resumo. *Apesar de todos os esforços, a classificação automática de documentos ainda é uma tarefa desafiadora. Isto porque os algoritmos de aprendizado devem lidar com dados esparsos, ruído e a ambiguidade inerentes à linguagem humana, assimetria da distribuição de dados, entre outros fatores. Alguns trabalhos defendem o uso de um aprendizado postergado no qual somente características relacionadas ao documento sendo classificado sejam levadas em consideração. Um modelo local e customizado é aprendido para cada documento sendo classificado. Esta abordagem requer um alto custo computacional pois deve operar no momento da classificação. Este trabalho propõe o uso de paralelismo para viabilizar operações de filtragem e engenharia de características em uma etapa anterior à classificação do documento.*

1. Introdução

Apesar do desenvolvimento de novas tecnologias na representação de texto (como por exemplo *embeddings*, *meta-características* [Canuto et al. 2015] e algoritmos de classificação), a classificação automática de documentos (ADC) ainda é uma tarefa desafiadora [Mendes et al. 2019]. Podemos citar como dificuldades da tarefa de ADC a esparsidade dos dados textuais, ruído e ambiguidade inerentes à linguagem humana e o desbalanceamento na distribuição dos dados, entre outros fatores. Para tentar diminuir estas dificuldades alguns trabalhos defendem o uso de um aprendizado postergado no qual somente características (termos que ocorrem nos documentos) relacionadas ao documento de consulta sendo classificado sejam levadas em consideração. Dessa forma um modelo local e customizado é aprendido para cada documento que está sendo classificado.

A pesquisa dos k vizinhos mais próximos (kNN) [Manning et al. 2010] é uma técnica de aprendizado postergado que é amplamente utilizada em aplicações do mundo real como filtragem de spam e clusterização. Neste contexto, o kNN permite encontrarmos uma projeção do documento de consulta (instância de teste) na base de treino e construirmos um modelo de aprendizado customizado para o documento. Outra operação que tem gerado bons resultados é a combinação de características (coocorrência de termos em documentos, seção 3.1) do documento de consulta, que tem resultado em uma melhora da representação do espaço para esses documentos, e consequentemente numa maior eficácia na classificação de textos. Após gerar essas novas características os dados resultantes podem ser encaminhados para um classificador externo como o SVM.

Apesar de mais flexíveis, estas abordagens requerem um alto custo computacional pois devem operar no momento da classificação. Este trabalho propõe uma abordagem paralela para acelerar o tempo de execução destas operações de filtragem (projeção) e engenharia (combinação) de características em uma etapa anterior à ADC [Figueiredo et al. 2011]. Ambas operações devem levar em conta a alta dimensionalidade e esparsidade dos dados associados à tarefa de ADC. A implementação destas ideias é feita considerando-se a arquitetura de programação paralela GPU (Unidade de Processamento Gráfico) e o sistema MetaLazy [Mendes et al. 2019].

2. Classificação com Aprendizado Postergado

Os métodos do estado da arte em *Lazy* ADC tem foco no documento de consulta e seus vizinhos, bem como no uso de um único método de classificação depois que a vizinhança for selecionada para treinamento. No entanto, conjuntos de dados diferentes podem ter particularidades para as quais classificadores diferentes podem se ajustar melhor. Outro problema enfrentado por *lazy classifiers* é o desbalanceamento das classes do *dataset*.

2.1. Engenharia de Características

A eficácia da ADC depende das técnicas de classificação (algoritmos) e da representação do espaço do conjunto de dados. Um espaço de representação dos documentos muito grande pode tornar a tarefa dos classificadores inviável, pois a classificação de novos documentos tornaria-se muito lenta. Uma solução para esse problema é reduzir o espaço de representação. Uma tendência recente na classificação automática de documentos é a engenharia de características que trabalha no nível dos dados com a introdução de novas características derivadas da representação original (*bag-of-words*). Com essa nova representação, pode-se obter informações relevantes sobre os dados, que não seriam obtidas com a representação original, e assim melhorar a eficácia da classificação.

Uma estratégia para transformar o espaço de entrada é obter pares de características mais representativos do documento de consulta e usá-los como uma maneira de aumentar o espaço para este documento e as amostras de treinamento desta instância. Aprimorando a representação do documento de consulta através da coocorrência de termos entre os documentos. No entanto, criar o espaço de representação de uma maneira ávida é inviável devido ao custo computacional e ao aumento da esparsidade. Potencialmente quaisquer duas palavras podem coocorrer, podendo aumentar um vocabulário de tamanho $|V|$ para $|V|^2$.

O aprendizado postergado pode ser utilizado para reduzir o custo computacional, pois na classificação do documento de consulta aproveitamos para melhorar seu espaço

de representação e potencialmente a qualidade de sua vizinhança. É criada uma expansão da representação de texto baseada nas coocorrências de termos restritos aos termos que ocorrem no documento de consulta. Usando apenas pares de palavras que coocorrem no documento de consulta (instância de teste) para expandir a sua representação e a representação dos documentos de treinamento. O tamanho do vocabulário expandido é limitado a $|V| + |D_t|^2$, onde $|D_t|$ é o número de termos no documento de consulta e $|D_t| << |V|$.

2.2. FaSST kNN

O método proposto por [Amorin et al. 2018] é uma solução exata de kNN, mas não usa a abordagem de força bruta. Como é usado um conjunto de dados textuais, evita-se comparar a consulta com todos os documentos da coleção, criando um índice invertido e encontrando rapidamente os termos compartilhados entre os documentos do *dataset* com o documento de consulta. Isso poupa muito espaço, já que o índice invertido corresponde a uma representação esparsa dos dados.

Além disso, o algoritmo utiliza técnicas de filtragem para descartar documentos que não possuem termos importantes em comum em relação ao documento de consulta. Essa estratégia evita a computação desnecessária aumentando ainda mais o desempenho. No momento da consulta, é implementada uma filtragem baseada em limiar (*threshold*) selecionando amostras (documentos) que compartilham termos com valores altos de TF-IDF. O limiar é escolhido entre estas amostras, depois todas as distâncias menores do que o limiar devem ser podadas, enquanto as maiores distâncias são compactadas em um matriz menor. O cosseno é utilizado como métrica para o cálculo de distância. Finalmente, os k vizinhos mais próximos são determinados através do uso de um algoritmo *radix sort* na matriz compactada.

2.3. MetaLazy

O algoritmo *MetaLazy* proposto por [Mendes et al. 2019], é um *lazy* ADC *meta-method* que alcança melhorias em relação aos métodos do estado da arte, sendo que suas principais contribuições são: Escolha dinâmica de um classificador por *dataset*; Expansão de características; Reponderação das instâncias de treinamento; Extensa experimentação.

A ideia central do *MetaLazy* é dividida nas seguintes etapas: seleção das características mais importantes do documento de consulta para a classificação; geração de novas características com coocorrência; busca dos k vizinhos do documento dentro dos dados de treinamento; criação de um modelo para a vizinhança e classificação do documento.

3. Método Proposto

3.1. Algoritmo Para Geração de Características de Coocorrência

Para a transformação do espaço de representação do documento adotamos uma estratégia postergada, permitindo considerar informações relevantes do documento de consulta para a criação do novo espaço de representação aumentado. O novo espaço melhora a eficácia da classificação, expandindo características, que coocorrem apenas no documento de consulta, para os documentos de treino e a instância de consulta. Como o número de coocorrências distintas de pares de termos em um documento pode tornar a tarefa impraticável, a abordagem escolhida permite selecionar pares com as características mais representativas (TF-IDF), conforme detalhado a seguir.

Considere F_{x_t} o conjunto de características distintas de um documento de teste x_t com n características. Geramos as características adicionais compostas de *m* características por amostragem de F_{x_t} usando a distribuição de probabilidades criada a partir dos valores de TF-IDF. Dessa forma, geramos $m/2$ pares de características, com $m < n$. O valor de m é passado para o algoritmo como um parâmetro de entrada. Para amostras de F_{x_t}, primeiro definimos $p_f = v(f)/s$, para cada $f \in F_{x_t}$, como a probabilidade de escolha da característica $f \in F_{x_t}$, onde $v(f)$ é o valor assumido pela característica f em x_t e $s = \sum_{f \in F_{x_t}} v(f)$. Então, com base na distribuição de probabilidades, amostramos o par de características (f_i, f_j) (s.t. $f_i \neq f_j$ e $f_i, f_j \in F_{x_t}$), com substituição. Cada par obtido é adicionado para um conjunto de pares que será usado para expandir a representação.

3.2. Integração MetaLazy - FaSST kNN

Neste trabalho também foi realizada uma integração entre os métodos MetaLazy e FaSST kNN. As implementações foram desenvolvidas nas linguagens Python e C/C++ e as etapas a seguir detalham os pontos considerados.

- O método FaSST kNN foi modificado para gerar as coocorrências, conforme descrito na seção anterior;
- As instâncias de teste (documento de consulta) são distribuídas para cada *CPU/GPU* quando é feita a leitura dos dados de teste;
- Para cada documento de teste são gerados os pares de coocorrências ($feature\_id\ feature\_value$);
- Os pares coocorrentes são mapeados para características com *ids* acima do total de características, ou seja, *total + i* para cada novo par;
- Iteração sobre os pares gerados para checar se os documentos de treino da instância de teste possuem os pares (checado através do índice invertido na *CPU*);
- O FaSST kNN retorna os k vizinhos mais próximos e são escritos em arquivo as distâncias em relação à instância de teste, os ids de cada vizinho e as novas características criadas pelos pares, junto com seus valores;
- O arquivo fica com *k + 1* linhas por instância de teste, a primeira com as distâncias e as k próximas linhas com os vizinhos ($doc\_id\ feature\_id\ feature\_value$);
- Lê o arquivo kNN e recupera o conteúdo dos vizinhos na matriz de treino *X_train* com o *doc_id* que tem em cada linha;
- Adiciona as novas características, caso tenha;
- Os dados dos k vizinhos são transformados em uma matriz esparsa e passados para o classificador (NB, RF, Extra RF, LR).

3.3. Experimentos

Para avaliar a integração entre os métodos *MetaLazy* e o *FaSST kNN* foram realizados experimentos com os conjuntos de dados apresentados na tabela 1.

Tabela 1. Informação geral dos datasets textuais.

Dataset	# Termos	# Documentos	Densidade	Tamanho
20NG	61,032	18,766	129.74	30MB
4Uni	40,192	8,274	139.81	14MB
ACM	56,413	24,897	27.76	8.5MB
Reuters90	19,572	13,327	77.26	13MB

Abaixo são apresentados alguns parâmetros utilizados nos experimentos.

- Método *multicore*: versão paralela por documento de consulta;
- Método *mt + FaSST-kNN*: integração MetaLazy e FaSST kNN.
- Foi utilizada validação cruzada com 5 folds;
- k=200, 12 processos python e classificador RF10;
- Sem aplicar seleção de características;
- O tempo da combinação linear está embutido no FaSST kNN, para *mt + FaSST-kNN*;
- A soma de tempo (Comb. linear + FaSST kNN + Classificação) pode não dar o tempo total (Tempo de Teste) devido ao tempo gasto para um processo iniciar (até 75s no *dataset* 20NG).

Os resultados obtidos com a integração dos métodos são apresentados na tabela 2. Comparamos o método original do *MetaLazy* com o método resultante da integração entre *MetaLazy* e *FaSST kNN*.

Tabela 2. Tempo total e de cada fase em segundos.

Dataset	Método	Tempo de Teste	Comb. Linear	FaSST kNN	Classificação	Speedup
4Uni	multicore	293.23	203.53	14.36	74.50	2.55
	mt + FaSST-kNN	115.09	0.00	8 + 15.5	65.12	
20NG	multicore	1215.12	871.32	92.48	249.64	4.20
	mt + FaSST-kNN	289.29	0.00	12 + 33.78	184.49	
ACM	multicore	291.67	95.86	19.29	175.37	0.98
	mt + FaSST-kNN	298.16	0.00	10.4 + 17.90	165.77	
Reuters90	multicore	264.54	178.46	17.49	67.09	1.97
	mt + FaSST-kNN	134.07	0.00	9.8 + 14.5	68.36	

4. Considerações Finais

A avaliação dos resultados mostra um ganho de performance até $x4,2$ da integração entre os métodos *MetaLazy* e *FaSST kNN* em relação ao método *MetaLazy* original para os conjuntos de dados testados. Uma abordagem que pode melhorar o desempenho é fazer a avaliação em conjuntos de dados maiores, como o RCV1 e MEDLINE, para tentar obter maior ganho de performance na fase do kNN.

Futuramente outro ponto que pode ser explorado é a aplicação de técnica de filtragem por prefixo e sufixo em paralelo utilizando a desigualdade de Cauchy-Schwarz para obter um limite superior de poda e assim reduzir o espaço vetorial.

Referências

Amorin, L. A., Freitas, M. F., Silva, P. H., and Martins, W. S. (2018). A fast similarity search knn for textual datasets. In *In Proceedings of the XIX Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD 18)*, page 421–432. SBC.

Canuto, S., Gonçalves, M., Santos, W., Rosa, T., and Martins, W. (2015). An efficient and scalable metafeature-based document classification approach based on massively parallel computing. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 333–342. ACM.

Manning, C., Raghavan, P., and Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

Mendes, L. F., Gonçalves, M., Salles, T., Rocha, L., Ottoni, R., Couto, T., Resende, E., Cunha, W., Freitas, M., Martins, W., and Silva, P. H. (2019). "keep it simple, lazy" – metalazy: a new metastrategy for lazy text classification. In *Relatório Técnico*. DCC/UFMG.