Implementation in C+CUDA of Multi-Label Text Categorizers

Lucas Veronese, Alberto F. De Souza, Claudine Badue, Elias Oliveira, Patrick M. Ciarelli
Universidade Federal do Espírito Santo, 29075-910, Vitória-ES
{lucas.veronese, alberto, claudine, elias, pciarelli}@lcad.inf.ufes.br

In automated multi-label text categorization problems with large numbers of labels, the training databases are large, which may render the categorization time prohibitive for online systems. In this work, we evaluate the parallel implementation in C+CUDA of two multi-label text categorizers: the first is based on the *k*-Nearest Neighbors (*k*-NN) algorithm [1] and the second is based on Probabilistic Neural Networks (PNN) [2]. We implemented these algorithms in three different ways: sequential in C, parallel in C+CUDA and parallel using the C+CUBLAS library.

The *k*-NN categorizer finds the *k* nearest neighbors of an input document d_x in the set of previously learned documents, *TV*, according to some given distance metric —in the experiments reported in this paper, we used the cosine of the angle between the floating-point vector that represents the input document d_x (bag-of-words document representation [1]) and each document $d_i \in TV$, $\cos(d_x,d_i)$:

$$\cos(d_x, d_i) = \frac{d_x \bullet d_i}{\|d_x\| \ \|d_i\|} \qquad (1)$$

The *k*-NN categorizer employs a function $f(d_x,c_k)$ that returns the highest value of $\cos(d_x,d_i)$, for $d_i \in TV$ and $c_k \in C_i$, where C_i is the set of pertinent categories for the document d_i. It selects the *k* pairs $\langle d_x, c_i \rangle \in \langle \mathbf{D} \times C \rangle$ from the top of the ranking derived from $f(\cdot,\cdot)$.

The PNN used in this work was proposed by Oliveira et. al [2] and is composed of two feed-forward layers: pattern layer and summation layer. In the training phase, for each document d_i is created a set of neurons, one for each category $c_k \in C_i$, where each neuron n_i stores the vector d_i as a vector of term weights, $w_{k,i}$. In the categorization phase, an input document d_x is presented to the pattern layer. The *i*-th neuron, n_i, associated to category c_k in the pattern layer, calculates the activation function $A(d_x, c_k, n_i)$ for document d_x given by:

$$A(d_x, c_k, n_i) = \frac{1}{2\pi\sigma} \exp\left(\frac{d_x^t w_{k,i} - 1}{\sigma^2} \right), \ k=1, ..., |C|, \ i=1, \dots, |D_k| \qquad (2)$$

where σ is a constant for all neurons (adjusted during training for best categorization performance [2]), C is the whole set of possible categories, and D_k is the set of documents associated to category c_k. In the summation layer, which has as many neurons as $|C|$, each neuron is associated with a category c_k and computes the function $f(d_x,c_k)$:

$$f(d_x, c_k) = \sum_{i=1}^{|N_k|} A(d_x, c_k, n_i), \ k=1, ..., |C| \qquad (3)$$

where N_k is the number of neurons of the pattern layer associated to c_k. The categories c_k ranked above a threshold are predicted to the input document d_x.

We ran the C, C+CUDA and C+CUBLAS versions of our categorizers in an AMD Athlon 64 X2 (*Dual Core*) 5,200+ of 2.7 GHz, with 3GB of 800 MHz DRAM DDR2, and video card NVIDIA GeForce GTX 285, with 1GB of DRAM GDDR3.

The data set used is composed of 6,911 documents categorized into 105 different categories by specialists in the domain of the documents. Each one of these categories occurs in exactly 100 different documents, i.e., there are 100 documents of each category. Each document is represented by a vector of single precision floats of size 3,764 (the number of relevant terms in the system vocabulary).

To evaluate the performance of our categorizers in terms of time, we selected 6,910 documents of the data set for training, and a single one for testing the categorizers. Each categorizer was executed 100 times and the average was used to compare them. Table 1 shows the average times for each categorizer (rows) and categorizer implementation (columns), in addition to the speed-ups over the sequential implementation (last two columns). As the table shows, we achieved speed-ups of about 60 for the C+CUDA version and about 45 for the C+CUBLAS version. These results show that, with CUDA, it is possible to implement on-line text categorization and that, in some cases, it is worth implementing the whole code instead of using C+CUBLAS.

Table 1: Average times and speed-ups of our text categorizers

Categ.	**C** **(s)**	**C+CUDA** **(s)**	**C+CUBLAS** **(s)**	***Speed-up*** **C+CUDA**	***Speed-up*** **C+CUBLAS**
***k*-NN**	0.1928	0.0030	0.0042	64.26	45.90
PNN	0.1938	0.0033	0.0044	58.72	44.04

[1] F. Sebastiani, “Machine Learning in Automated Text Categorization”, *ACM Computing Surveys 34(1),* 2002, pp. 1-47

[2] E. Oliveira, P. M. Ciarelli, A. F. De Souza, and C. Badue. Using a Probabilistic Neural Network for a Large Multi-Label Problem. *Proceedings of the 10th Brazilian Symposium on Neural Networks (SBRN'08),* pp. 195-200, Salvador, Bahia, Brazil, October 2008.