Incremental Reranking for Hierarchical Text Classification

Qi Ju and Alessandro Moschitti

DISI, University of Trento, Italy
{qi,moschitti}@disi.unitn.it

Abstract. The top-down method is efficient and commonly used in hierarchical text classification. Its main drawback is the error propagation from the higher to the lower nodes. To address this issue we propose an efficient incremental reranking model of the top-down classifier decisions. We build a multiclassifier for each hierarchy node, constituted by the latter and its children. Then we generate several classification hypotheses with such classifiers and rerank them to select the best one. Our rerankers exploit category dependencies, which allow them to recover from the multiclassifier errors whereas their application in top-down fashion results in high efficiency. The experimentation on Reuters Corpus Volume 1 (RCV1) shows that our incremental reranking is as accurate as global rerankers but at least one magnitude order faster.

1 Introduction

Among others, two very well-known methods for the design of hierarchical text classifiers (HTC) are the *big bang* and the *top-down* approaches. The former learns a single (but generally complex) hierarchical classification model whereas the latter uses the hierarchical structure to decompose the entire problem into a set of smaller sub-problems. Then it proceeds in top-down fashion along the hierarchy, achieving high efficiency in both learning and prediction phase. The so-called pachinko-machine model [4] defines a multiclassifier for each node of the hierarchy with its node children. If a document is assigned to them, the multiclassifiers of their children are recursively activated. This way, the decisions are made from the root until the leaf nodes.

As shown in [4] larger hierachies, e.g., Yahoo! Categories and Dmoz, make the *big bang* approach inadequate as it is too slow. At the same time, the complexity of the task critically affects the accuracy of the top-down approach. To improve accuracy, node dependencies can be used, e.g., in [7, 2] SVM-struct optimizes the output by introducing dependencies between labels and in [6] hierarchical dependencies (not just label dependencies) are encoded in the learning algorithms. Unfortunately, such approaches are not enough efficient for a large-scale scenario.

In [5] we proposed a hybrid solution: the combination of *top-down* methods with a reranking algorithm based on Support Vector Machines (SVMs). The use of structural hierarchy features from tree kernel spaces [1] allowed us to achieve state-of-the-art accuracy on the entire RCV1. However, when the number of

categories becomes huge (thousands of categories) the generative model as well as the reranker become inadequate.

In this paper, we propose an efficient local incremental reranking model (LIR) consisting of a reranker for each multiclassifier associated with each hierarchy node like in the pachinko-machine approach (with the difference that we also assign documents to internal nodes). According to the top-down method, LIR recursively deals with the the subproblems instantiated by the hierarchy by applying the corresponding subrerankers. We carried out experiments on the entire RCV1 using the same setting we used in [5]. The results show that our LIR significantly improves the efficiency of our previous models while basically matching their accuracy. In the remainder of this paper, Section 2 introduces the LIR model, Section 3 illustrates our experiments and comparative results.

2 From Global to Local Incremental Reranker

In this section we briefly described our global reranker proposed in [5] and then we show our incremental version along with a comparative complexity analysis.

2.1 Global Reranker (GR)

Our approach consists of three different steps: (i) the application of the one-vs-all method to build a multiclassifier over all hierarchy categories; (ii) the use of the classification probability of the binary node classifiers to generate k global classification hypotheses, i.e., the set of categories that the target document belong to; and (iii) reranking them by means of an SVM using tree kernels applied to the hierarchy tree, i.e., each hypothesis is represented by the tree associated with the hierarchy, where the classification decisions are marked in the node themselves. It should be noted that in Step (i) no information about the hierarchy is used. Step (ii) generates global classification hypotheses by also deriving their joint probability, which is used for preliminary ranking them. Step (iii) uses a reranker that exploits structural features. This includes co-occurrences, e.g., given three categories, C_1, C_2 and C_3, it encodes their subsets $\{C_1, C_2, C_3\}$, $\{C_1, C_3\}$, $\{C_1, C_2, C_3\}$ as features. Additionally, it also encodes their structures, e.g., C_1 is father of C_2 which is father of C_3 as features. More details about hypothesis representation and their use are given in [5].

2.2 Local Incremental Reranker (LIR)

The global reranker suffers from the inefficiency of the big bang approach, with also the disadvantage of using a unique tree kernel classifier (i.e., the reranker), trained with large amount of data and applied to a potentially huge hierarchy tree. We solve both inefficiencies by defining a reranker for each multiclassifier, i.e., for each internal node with its children. To build the subreranker we need to: (i) obtain the individual decision probabilities output by the top-down one-vs-all classifiers, e.g., the probability of each local multiclassifier; (ii) generate the top k hypotheses based on the probabilities above for each internal node; and (iii) learn a reranker for each local multiclassifier, using the tree kernels applied to the hypothesis representation. The latter is just a tree constituted by a node and its children (obviously such classifier also labels internal nodes). In the classification phase, we apply the node multiclassifiers in top-down way and we rerank their

decisions with the local rerankers. Of course, we progress to the children of a node only after the reranking step of the multiclassifier associated with its father is terminated. This way, LIR exploits the efficient top-down algorithm but at the same time allows for capturing dependencies between father and its children. These dependencies are then propagated in a top-down fashion.

2.3 Computational Complexity Analysis

The focus of our paper is to improve the efficiency of rerankers. Thus, we will analyze the computational complexity of GR vs. the one of LIR. There are two sources of complexity in SVM using tree kernels: (i) the learning algorithm working in dual space; and (ii) the computation of the tree kernel function. Let us to define: m the number of hierarchy nodes, μ the number of the internal nodes and n the size of training data. The GR worst case complexity is given by the SVM learning, i.e., $O(n^c)$, where $2 < c < 3$, multiplied by the tree kernel times, which is quadratic in the number of tree nodes, i.e., $O(m^2)$. Thus GR runs in $O(n^c m^2)$.

The LIR worst case complexity happens when the hierarchy is flat ($m = 1$) but this is not an interesting case. Thus, let us consider, a non trivial hierarchy with $m >> 1$. We also consider the average case in which the training data is distributed uniformly between the categories[1]. With these assumptions, we have μ multiclassifiers, each with n/μ training examples. It follows that their learning complexity is $\mathrm{O}(\mu(n/\mu)^c)$ multiplied by the tree kernel complexity. This, considering that the local classifiers have on average $m/\mu+1$ nodes, is $(m/\mu+1)^2$. As a result, LIR shows a complexity of $\mathrm{O}(\mu(n/\mu)^c(m/\mu)^2) = O((n/\mu)^c m^2/\mu) < O(n^c m^2/\mu^3) < O(n^c)$, which is lower than the GR's one (we used the fact that $O(\mu^3) > O(m^2)$). The classification analysis is similar as there is (i) a quadratic term $O(n^2)$ wrt the number of support vectors (lower but proportional to n) and (ii) the usual $O(m^2)$ term for the tree kernel evaluation.

3 Experiments

We compare GR against LIR wrt accuracy and running time. We used Reuters Volume 1 (RCV1) with Lewis' split [3], which includes 23,149 news for training and 781,265 news for testing. We implement the top-down classifiers with SVMs using the default parameters (trade-off and cost factor = 1), linear kernel, normalized vectors (using the Euclidean norm), stemmed bag-of-words representation, $log(TF + 1) \times IDF$ weighting scheme and a common stop list. All the performance figures are provided by means of Micro/Macro-Average F1, evaluated from our test data over all 103 categories.

3.1 GR and LIR comparison on the whole RCV1

Table 1 reports the accuracy whereas Table 2 illustrates the learning and classification time. The table columns have the following meanings: (i) *flat* refers to the results achieved in [3] and [5], respectively; (ii) *top-down* is our reimplementation

[1] The usual case of the node father containing all the documents of the children, clearly violates such assumption. More complex equations taking into account this assumption can be defined but this is beyond the purpose of this paper.

F1	baseline		GR	LIR
	flat	top down		
Micro-F1	0.816	0.819	0.849	0.841
Macro-F1	0.567	0.578	0.615	0.611

Table 1. Micro/Macro-F1 of different models on RCV1.

time cost	GR	LIR
Training (s)	9023.24	508.75
Test (h)	43.40	4.31

Table 2. Classification and training time of GR and LIR on RCV1.

of the conventional top-down method; (iii) GR represents the best accuracy of kernel-based reranking models applied to the hypotheses made on all hierarchy classification; and (iv) LIR refers to our faster method.

We can clearly see from Table 1 that the top-down model slightly improves the flat models reported in [3], i.e., by $81.9 - 81.6 = 0.3$. This is significant with $p = 10^{-5}$, according to our significance test using approximate randomization (please consider that the test set contains about 800k examples). When LIR is applied to the top-down baseline, the latter improves by 2.2 absolute percent points (significant at $p = 10^{-5}$) in Micro-F1; similarly the baseline of flat model improves by 2.5 points (in Micro-average). Most importantly, LIR remarkably outperforms the reranking model proposed in [5] in efficiency, i.e., 9023.24/508.75= 17.8 times in learning and 43.40/4.31= 10.0 times in testing. In contrast, it looses 0.8 points in Micro-F1.

In conclusion, our local incremental reranking model based on the conventional top-down approach allows for efficiently using structural dependencies provided by tree kernels in HTC. The comparative experiments with the state-of-the-art model, GR, show that LIR is much more efficient while showing almost the same accuracy.

Acknowledgments

This research has been partially supported by the EC's Seventh Framework Programme (FP7/2007-2013) under the grants #247758: EternalS – Trustworthy Eternal Systems via Evolving Software, Data and Knowledge, and #288024: LiMoSINe – Linguistically Motivated Semantic aggregation engiNes

References

1. Collins, M., Duffy, N.: New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In: ACL'02.
2. Finley, T., Joachims, T.: Parameter learning for loopy markov random fields with structural support vector machines. In: ICML Workshop on Constrained Optimization and Structured Output Spaces (2007)
3. Lewis, D.D., Yang, Y., Rose, T., Li, F.: RCV1: A new benchmark collection for text categorization research. JMLR (2004)
4. Liu, T.Y., Yang, Y., Wan, H., Zeng, H.J., Chen, Z., Ma, W.Y.: Support vector machines classification with a very large-scale taxonomy. SIGKDD (2005)
5. Moschitti, A., Ju, Q., Johansson, R.: Modeling topic dependencies in hierarchical text categorization. In: ACL 2012
6. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Kernel-based learning of hierarchical multilabel classification models. JMLR (2006)
7. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: ICML (2004)