RESEARCH AND DEVELOPMENT
ERCIM News No.44 - January 2001 [contents]

Boosting Algorithms for Automated Text Categorization

by Fabrizio Sebastiani, Alessandro Sperduti and Nicola Valdambrini

---

As part of its Digital Library activities, and in collaboration with the Department of Computer Science of the University of Pisa, IEI-CNR is working on the construction of tools for the automatic or semi-automatic labeling of texts with thematic categories or subject codes.

Text categorization (TC) uses machine learning (ML) techniques to build automatic text classifiers, ie programs capable of labelling natural language texts with thematic categories from a predefined set. TC has many applications, including automated document indexing for Boolean information retrieval systems, adaptive document filtering and spam e-mail blocking, Web page classification under hierarchical directories, and even author identification for documents of controversial paternity. In TC a general inductive process (called the 'learner') automatically builds a classifier by learning the characteristics of the categories from a 'training set' of preclassified documents. Once a classifier has been built, its effectiveness (ie its capability to take the right categorization decisions) may be tested by applying it to a 'test set' of preclassified documents and checking the degree of correspondence between the decisions of the automatic classifier and those encoded in the test set.

A wealth of different ML methods have been applied to TC, including probabilistic classifiers, decision trees, regression methods, batch and incremental linear methods, neural networks, example-based methods, and support vector machines. Recently, the 'classifier committees' method has gained popularity. The underlying idea is that, given a task that requires expert knowledge, several independent experts are better than one if their individual judgments are appropriately combined. In TC, this means applying several different classifiers to the same task of deciding under which category (or categories) a document should be classified, and then combining their outcome appropriately.

Boosting is a method for generating classifier committees which is characterized by state-of-the-art effectiveness and a strong theoretical grounding in computational learning theory. In the boosting method, the S classifiers that make up the committee are not trained in a conceptually parallel and independent way, as in other committee-based methods, but sequentially. In this way, when generating the i-th classifier the learner can take into account how the previously generated classifiers perform on the training documents, and can force the i-th classifier to concentrate on those training documents where the previously generated classifiers performed worst.

Each classifier is a one-level decision tree, ie tests for the presence or absence of a term in a document, depending on whether a positive or negative classification decision is taken. The term on which this decision hinges is generally different for each of the S classifiers and is chosen each time, according to a complex error-minimization policy. The key step of this policy consists in scoring each term by a scoring function, and picking the term with the smallest score. The scoring function adopted by the boosting method is extremely complex, and computationally onerous. As this computation has to be performed for each of the terms and this has to be iterated S times, boosting is a

computationally expensive method (in typical TC applications both the number of terms and the number S of iterations are in the tens of thousands).

We have devised a new method, called AdaBoost.MH(KR), based on the construction of a sub-committee of classifiers at each iteration of the boosting process. Instead of using just the best term, as the standard AdaBoost.MH(R) algorithm does, we select the best K terms, and generate K classifiers, grouping them in a sub-committee. This idea comes from the observation that, while in the first iterations (ie in the generation of the first classifiers) the scores of the best terms are markedly different from each other and from the worst ones, in the last iterations the differences among scores are very small. In AdaBoost.MH(KR) we choose, at each iteration, K top-ranked terms that would be good candidates for selection in the next K AdaBoost.MH(R) iterations. We can thus build a committee composed of S sub-committees at a computational cost comparable to that required by AdaBoost.MH(R) to build a committee of S classifiers. In fact, most of the computation required by the boosting process is devoted to calculating the term scores, and by using only the top-scoring term AdaBoost.MH(R) exploits these hard-won scores only to a very small extent.

In all our experiments (conducted on the standard Reuters-21578 benchmark), AdaBoost.MH(KR) achieved better effectiveness and much better efficiency than AdaBoost.MH(R). For instance, in one experiment the effectiveness achieved by AdaBoost.MH(KR) with 300 iterations was not obtained by AdaBoost.MH(R) in 16,000 iterations. AdaBoost.MH(KR) is more effective than AdaBoost.MH(R) because of the latter's 'greedy' approach when selecting the best classifier for the final committee. This approach does not guarantee optimality of selection, and AdaBoost.MH(R) is unable to search the classifier space. On the contrary, AdaBoost.MH(KR) allows the designer to explore, at least partly, the space of classifiers by setting its internal parameters (eg number of classifiers in each committee) to different values or according to different policies.

Links:
http://faure.iei.pi.cnr.it/~fabrizio/Publications/CIKM00/CIKM00.pdf

Please contact:
Fabrizio Sebastiani - IEI-CNR
Tel: +39 050 3152 892
E-mail: fabrizio@iei.pi.cnr.it