Using Progressive Filtering to Deal with Information Overload

Andrea Addis, Giuliano Armano, and Eloisa Vargiu

Abstract— **In the age of Web 2.0 people organize large collections of web pages, articles, or emails in hierarchies of topics, or arrange a large body of knowledge in ontologies. This scenario requires automatic text categorization systems able to cope with underlying taxonomies in an effective and efficient way, so that information overload and input imbalance can be suitably dealt with. In this work, we propose a hierarchical text categorization approach that decomposes a given rooted taxonomy into pipelines, one for each path that exists between the root and each node of the taxonomy, so that each pipeline can be tuned in isolation. Experimental results, performed on Reuters and DMOZ data collections, show that the proposed approach performs better than a flat approach in presence of input imbalance.**

Index Terms— **Hierarchical Text Categorization, Information Overload, Input Imbalance, Reuters Corpus, DMOZ.**

I. Introduction

It is not surprising that in the age of Web 2.0 people organize large collections of web pages, articles, or emails in hierarchies of topics, or arrange a large body of knowledge in ontologies. This organization allows to focus on a specific level of detail, ignoring specialization at lower levels and generalization at upper levels. In this scenario, the main goal of automatic categorization systems is to deal with underlying taxonomies in an effective and efficient way. In fact, considering categories organized in a hierarchy may help to improve the overall performances. Furthermore, a hierarchical approach can give benefits in real-world scenarios, characterized by information overload [1] and imbalanced data [2]. In fact, in these contexts, users are "overloaded" of information and it becomes very difficult for them to select contents according to their personal interests, especially when contents are frequently updated (e.g., news, online newspaper articles, reuters, RSS feeds, wikis, and blogs). Moreover, interesting and uninteresting documents (i.e., positive and negative examples, respectively) are typically imbalanced, turning classifiers trained with the same percent of positive and negative examples into inadequate tools.

In this work, we propose a hierarchical text categorization approach, called Progressive Filtering (PF), focused on dealing with the input imbalance that typically occurs in real-world scenarios. It is worth pointing out in advance that, in its simplest setting, PF decomposes a given rooted taxonomy into pipelines, one for each path that exists between the root and each node of the taxonomy, so that each pipeline can be tuned in isolation. To this end, for each pipeline, a threshold selection algorithm (TSA) has been devised, aimed at finding a sub-optimal combination of thresholds for each pipeline.

University of Cagliari, Piazza d'Armi, I-09123, Cagliari, Italy email: {addis, armano, vargiu}@diee.unica.it

II. Related Work

A. Hierarchical Text Categorization

In 1997, Koller and Sahami [3] carried out the first proper study of hierarchical text categorization on the Reuters-22173 collection. Their approach showed that hierarchical models perform well when a small number of features per class is used, whereas no advantages were found using the hierarchical model for large numbers of features. McCallum et al. [4] compare two techniques: (i) exploring all possible paths in the taxonomy, and (ii) greedily selecting at most two branches according to their probability, as done in [3]. Results show that greedy selection is error prone but computationally efficient. D'Alessio [5] proposed a system in which a document is classified as belonging to a given category in the event that a weighted sum of feature occurrences is greater than the threshold of that category. Dumais and Chen [6] used the hierarchical structure for two purposes: (i) training several SVMs, one for each intermediate node, and (ii) classifying documents by combining scores from SVMs at different levels. In the work by Ruiz and Srinivasan [7] a variant of the Hierarchical Mixture of Experts model is used. A hierarchical classifier combining several neural networks is also proposed in [8]. Ceci and Malerba [9] presented a comprehensive study on hierarchical classification of web documents. More recently, a multi-label hierarchical text categorization algorithm is proposed [10], which consists of a hierarchical variant of ADABOOST.MH –a well-known member of the family of "boosting" learning algorithms.

B. The Input Imbalance Problem

Japkowicz [11] studied the class imbalance problem, i.e. the problem related to domains in which (in binary classification) one class is represented by a large number of examples whereas the other is represented by only a few. High imbalance occurs in real-world domains where the decision system is aimed at detecting a rare but important case [2]. A number of solutions to the class imbalance problem have been proposed both at the data- and algorithmic-level. Data-level solutions include many different forms of resampling such as random oversampling with replacement, random undersampling, directed oversampling, directed undersampling, oversampling with informed generation of new samples, and combinations of the above techniques. To counteract the class imbalance, algorithmic-level solutions include adjusting the costs of the

various classes, adjusting the decision threshold, and adopting recognition-based rather than discrimination-based learning. Hybrid approaches have also been used to handle the class imbalance problem.

III. The Proposed Approach

In this work, we are interested in studying how to cope with input imbalance in a hierarchical text categorization setting. The underlying motivation is that, in real world applications, the ratio between interesting and uninteresting documents is typically very low, so that classifiers trained with a balanced training set are inadequate to deal with those environments. To this end, we perform the training activity in two phases. First, each classifier is trained by using a balanced dataset, then, a threshold selection algorithm is applied and thresholds are calculated taking into account the input imbalance.

A theoretical assessment of the approach is beyond the scope of this paper, the interested reader could refer to [12] for further details.

A. The Proposed Progressive Filtering Approach

A way to implement Progressive Filtering (PF) consists of unfolding the given taxonomy into pipelines of classifiers, as depicted in Figure 1. Each node of the pipeline is a binary classifier able to recognize whether or not an input belongs to the corresponding class (i.e., to the corresponding node of the taxonomy).

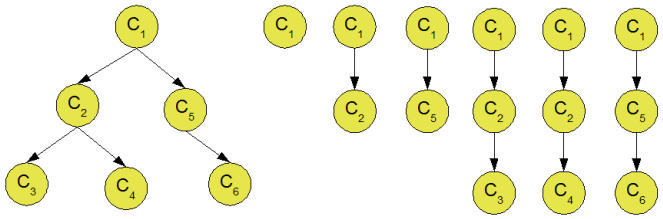

Fig. 1. A taxonomy and its corresponding pipelines.

In principle, PF could be applied to classify any kind of items, including images, audios, videos, textual documents. As in this paper we are interested in applying PF to hierarchical text categorization, only textual documents (documents for short, hereinafter) have been considered.

During the training activity, first, each classifier is trained by generating a balanced dataset from the given hierarchical training set. For any given node, the training set contains documents taken from the corresponding subtree and documents of the sibling subtrees –as positive and negative examples, respectively.

Given a taxonomy, where each node represents a classifier entrusted with recognizing all corresponding positive inputs (i.e., interesting documents), each input traverses the taxonomy as a "token", starting from the root. If the current classifier recognizes the token as positive, it passes it on to all its children (if any), and so on. A typical result consists of activating one or more branches within the taxonomy, in which the corresponding classifiers have been activated by the given token. Let us note that, partitioning the taxonomy in pipelines gives rise to a set of new classifiers, each represented by a pipeline. For instance, the taxonomy depicted in Figure 1 gives rise to six pipelines.

B. The Proposed Threshold Selection Algorithm

It is worth pointing out in advance that the same classifier may have different behaviors, depending on which pipeline it is embedded. As a consequence, each pipeline can be considered in isolation from the others. For instance, given $\pi_1 = \langle C_1, C_2, C_3 \rangle$ and $\pi_2 = \langle C_1, C_2, C_4 \rangle$, the classifier C_1 is not compelled to share the same threshold (the same holds for C_2).

A relevant problem is how to calibrate the threshold of the binary classifiers embedded by each pipeline in order to optimize the pipeline behavior. Given a set of documents D and a set of labels C, in classical text categorization a function $CSV_i : D \rightarrow [0, 1]$ exists for each $c_i \in C$. The behavior of c_i is controlled by a threshold θ_i, responsible for relaxing or restricting the acceptance rate of the corresponding classifier. In particular, given $d \in D$, $CSV_i(d) \geq \theta_i$ entails the decision of categorizing d under c_i, whereas $CSV_i(d) < \theta_i$ entails the decision of not to categorize d under c_i. In PF, let us still assume that CSV_i exists with the same semantics adopted for the classical case. Considering a pipeline π, composed by n classifiers, the acceptance policy strictly depends on the vector of thresholds $\Theta_\pi = \langle \theta_1, \theta_2, \cdots, \theta n \rangle$ that embodies the thresholds of all classifiers in π. In order to categorize d under π, the following constraint must be satisfied: for $k = 1..n$, $CSV_i(d) \geq \theta_k$. On the contrary, d is not categorized under c_i in the event that a classifier in π exists that rejects it.

Searching for a optimal or sub-optimal combination of thresholds in a pipeline can be actually viewed as the problem of finding a maximum in a utility function F that depends on the corresponding thresholds θ:

$$\theta^* = \underset{\theta}{argmax} \;\; F(\theta) \tag{1}$$

Unfortunately, the task of threshold calibration is characterized by a high time complexity. Hence, given the utility function F, we are interested in finding an effective way to reach a sub-optimum in the task of determining which are the best thresholds, while taking into account the actual ratio between positive and negative examples that depends on the given scenario.

Bearing in mind that the lower the threshold the less restrictive the classifier, the sub-optimal combination of thresholds is calculated according to the constraints imposed by TSA, a greedy bottom-up algorithm that relies on two functions:

- *repair* ($\mathcal{R}$), which operates on a classifier C by increasing or decreasing its threshold –i.e., $\mathcal{R}(up, C)$ and $\mathcal{R}(down, C)$, respectively, until the selected utility function reaches a local maximum.
- *calibrate* ($\mathcal{C}$), which operates going downwards from the given classifier to its offspring by repeatedly calling $\mathcal{R}$. It is intrinsically recursive and at each step it calls $\mathcal{R}$ to calibrate the current classifier.

Given a pipeline $\pi = \langle C_1, C_2, \cdots, C_L \rangle$, where L is its depth, first all thresholds are set to zero (so that the corresponding classifiers accept every token). TSA is then defined as follows:

$$TSA(\pi) := for\, k = L\, downto\, 1\, do\, \mathcal{C}(up, C_k) \tag{2}$$

which indicates that calibrate is applied at each node of the pipeline, starting from the leaf ($k = L$). The calibrate function is defined as follows:

$$\begin{aligned} &\mathcal{C}(up, C_k) := \mathcal{R}(up, C_k),\ k = L \\ &\mathcal{C}(up, C_k) := \mathcal{R}(up, C_k) + \mathcal{C}(down, C_{k+1}),\ k < L \end{aligned} \tag{3}$$

and

$$\begin{aligned} &\mathcal{C}(down, C_k) := \mathcal{R}(down, C_k),\ k = L \\ &\mathcal{C}(down, C_k) := \mathcal{R}(down, C_k) + \mathcal{C}(up, C_{k+1}),\ k < L \end{aligned} \tag{4}$$

where the $+$ operator denotes a sequence operator, meaning that in the formula $a + b$ action a is performed *before* action b.

To better illustrate the approach, let us consider the unfolding corresponding to a pipeline of three classifiers $\pi = \langle C_1, C_2, C_3 \rangle$:

$$\begin{aligned} TSA(\pi) \quad &:= for\, k = 3\, downto\, 1\, do\, \mathcal{C}(up, C_k) \\ &\textbf{\textit{step 1}} \\ \mathcal{C}(up, C_3) \quad &= \mathcal{R}(up, C_3) \\ &\textbf{\textit{step 2}} \\ \mathcal{C}(up, C_2) \quad &= \mathcal{R}(up, C_2) + \mathcal{C}(down, C_3) \\ &= \mathcal{R}(up, C_2) + \mathcal{R}(down, C_3) \\ &\textbf{\textit{step 3}} \\ \mathcal{C}(up, C_1) \quad &= \mathcal{R}(up, C_1) + \mathcal{C}(down, C_2) \\ &= \mathcal{R}(up, C_1) + \mathcal{R}(down, C_2) + \mathcal{C}(up, C_3) \\ &= \mathcal{R}(up, C_1) + \mathcal{R}(down, C_2) + \mathcal{R}(up, C_3) \end{aligned}$$

Once calculated the sub-optimal combination of thresholds for a given imbalance, the pipelines are ready to be used in the corresponding scenario. Let us note, here, that this combination depends on both the adopted dataset and the actual input imbalance. In fact, different goals for the system lead to different behaviors.

As for the time complexity of TSA, to simplify the problem, let us define in advance the granularity step δ ($0 < \delta << 1$). This step identifies the maximum number of points $p = \delta^{-1}$ to be checked for each classifier in a pipeline. L being the length of the pipeline, the average time of TSA is proportional to $(L + L^2) \cdot p/2$, which implies a complexity $O(p \cdot L^2)$. Hence, the time complexity is quadratic with the number of classifiers embedded by a pipeline. A comparison between TSA and the brute-force approach is unfeasible, the vector of thresholds being composed by real numbers. Nevertheless, experimental results show that the effectiveness of TSA is almost identical to the one obtained by running a relaxed version of the brute-force approach, say RBF, in which only p points are checked for each classifier in a pipeline. Note that, although relaxed with reference to the brute-force approach, the average time of RBF is proportional to p^L, in turns implies a complexity $O(p^L)$.

For an experimental assessment of TSA, the interested reader could refer to [13].

IV. Experiments and Results

Since we are interested in studying how the input imbalance affects text categorization performance, we performed experiments to assess how PF deals with the input imbalance problem.

A. Setting Up Experiments

Experiments have been performed by customizing to this specific task X.MAS [14], a generic multiagent architecture built upon JADE [15], devised to make it easier the implementation of information retrieval and information filtering applications.

We chose Reuters Corpus Volume I (RCV1-v2)[1] and DMOZ[2] as benchmark datasets. As for RCV1-v2, the total number of codes actually assigned to the data is 93, whereas the overall number of documents is about 803,000 and each document belongs to at least to one category and, on average, to 3.8 categories. As for DMOZ, being interested in assessing the effects of the taxonomy depth on PF performances, we selected 17 categories with a maximum depth of 5. The underlying motivation is that in this way each category has at least 300 documents for different test phases. The corresponding training set includes a subset of the 136,000 documents, each belonging to one category.

To evaluate the performances of PF, we considered several measures. First, the approach is assessed resorting with macro-averaging, a conventional metric for calculating the performances of a text categorization system. We also applied F_1, obtained from the general definition of F_β by imposing the same degree of importance for precision and recall. Then, the approach is validated using four additional evaluation measures: misclassification-, generalization-, and specialization-error, as well as unknown ratio.

B. Results

The main question we are interested in is the effectiveness of the proposed approach with respect to flat classification. In order to guarantee a fair comparison, the same classification system has been adopted. In our experiments we adopted a classifier based on the *wk*-NN technology. The motivation for the adoption of this particular technique stems from the fact

[1] http://www.reuters.com
[2] http://www.dmoz.org

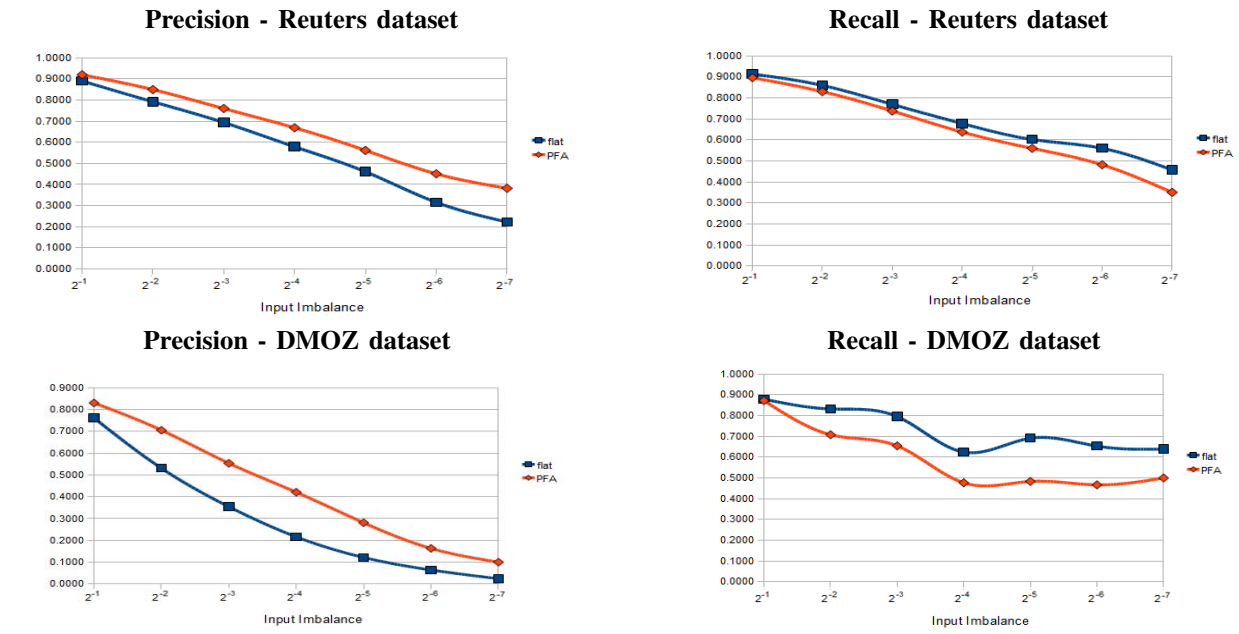

Fig. 2. Comparison of precision and recall between PF and flat classification.

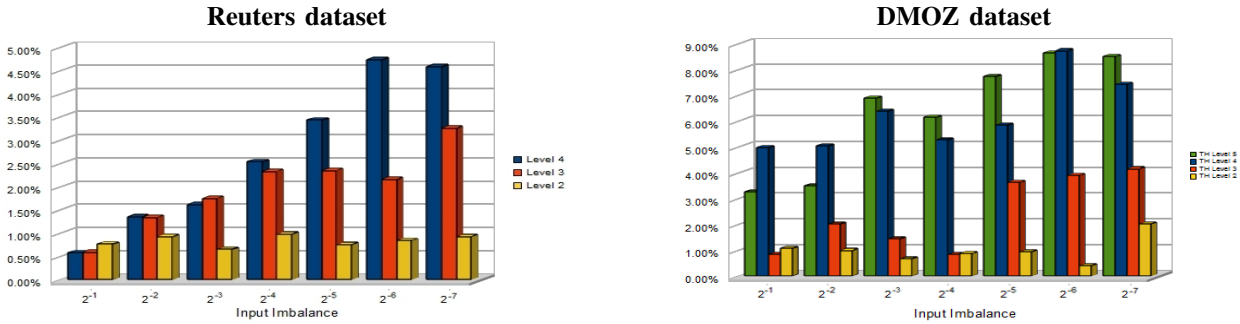

Fig. 3. Performance improvement.

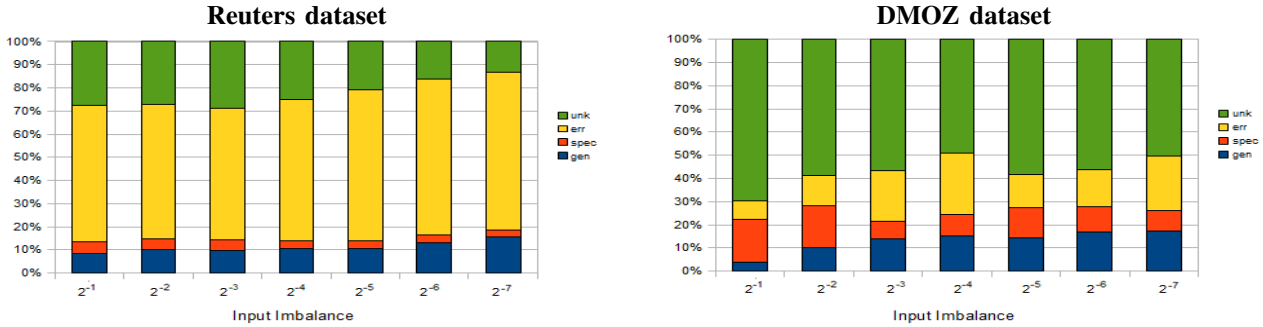

Fig. 4. Hierarchical measures.

that it does not require specific training and is very robust when applied to noisy data.

First, each classifier is trained with a balanced data set of 1000 documents (for Reuters) and 100 (for DMOZ) by using 200 (TFIDF) features selected in accordance with their information gain. Then, the best thresholds are selected. Both the thresholds of the pipelines and of the flat classifiers have been chosen by adopting F_1 as utility function[3]. As for

[3]The utility function can be adopted depending on the constraint imposed by the given scenario. For instance, F_1 is suitable if one wants to give equal importance to precision and recall.

pipelines, we used a step δ of 10^{-4} for TSA.

Experiments have been performed by assessing the behavior of the proposed hierarchical approach in presence of different percentages of positive examples versus negative examples, i.e., from 2^{-1} to 2^{-7}. We considered only pipelines that end with a leaf node of the taxonomy. Accordingly, for the flat approach, we considered only classifiers that correspond to a leaf.

PF vs Flat Classification. Figure 2 shows macro-averaging of precision and recall for Reuters and DMOZ datasets. Precision and recall have been calculated for both the flat classifiers and the pipelines by varying the input imbalance. As pointed out by experimental results (for precision), the distributed solution based on pipelines has reported better results than those obtained with the flat model. On the contrary results on recall are worse than those obtained with the flat model.

Improving performance along the pipeline. Figure 3 shows the performance improvements in terms of F_1 on Reuters and DMOZ datasets of the proposed approach with respect to the flat one. The improvement has been calculated in percentage with the formula $(F_1(pipeline) - F_1(flat)) \times 100$. Experimental results –having the adopted taxonomies a maximum depth of five– show that PF performs always better than the flat approach, the filtering effect of a pipeline being not negligible. Nevertheless, as for DMOZ, let us note that these results are not always statistically significant, depending on the amount of examples of the categories under analysis. This is an unavoidable side effect related to the decision of performing experiments with pipelines of length five.

Hierarchical Metrics. According to [9], we studied the overall error in terms of generalization-, specialization-, and misclassification-error, as well as unknown-ratio. Figure 4 depicts the results obtained varying the imbalance on Reuters and DMOZ datasets. Analyzing the results, it is easy to note that the generalization-error and the misclassification-error grow with the imbalance, whereas the specialization-error and the unknown-ratio decrease. As for the generalization-error, i.e., the percentage of documents misclassified in a supercategory of the correct one, it depends on the overall number of FNs, the greater the imbalance the greater the amount of FNs. Hence, the generalization-error increases with the imbalance. In presence of input imbalance, the trend of the generalization-error is similar to the trend of the recall. As for the specialization-error, i.e., the percentage of documents misclassified in a subcategory of the correct one, it depends on the overall number of FPs, the greater the imbalance, the lower the amount of FPs. Hence, the specialization-error decreases with the imbalance. In presence of input imbalance, the trend of the specialization-error is similar to the trend of the precision. As for the unknown-ratio and misclassification-error, let us point out that an imbalance between positive and negative examples can be suitably dealt with by exploiting the filtering effect of classifiers in the pipeline.

V. Conclusions

In this paper, we proposed a hierarchical text categorization approach. In particular, we were interested in studying the impact of the input imbalance that typically occurs in real-world scenarios. The approach decomposes a given rooted taxonomy into pipelines, one for each path that exists between the root and each node of the taxonomy, so that each pipeline can be studied in isolation. Experimental results, performed on Reuters and DMOZ data collections, validate the assumption that the proposed approach performs better than a flat approach in presence of input imbalance.

References

[1] C. C. Yang, H. Chen, and K. Hong, "Visualization of large category map for internet browsing," *Decis. Support Syst.*, vol. 35, no. 1, pp. 89–102, 2003.

[2] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: a review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, pp. 25–36, 2006.

[3] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," in *Proceedings of ICML-97, 14th International Conference on Machine Learning*, D. H. Fisher, Ed. Nashville, US: Morgan Kaufmann Publishers, San Francisco, US, 1997, pp. 170–178.

[4] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng, "Improving text classification by shrinkage in a hierarchy of classes," in *Proceedings of ICML-98, 15th International Conference on Machine Learning*, J. W. Shavlik, Ed. Madison, US: Morgan Kaufmann Publishers, San Francisco, US, 1998, pp. 359–367.

[5] S. D'Alessio, K. Murray, and R. Schiaffino, "The effect of using hierarchical classifiers in text categorization," in *Proceedings of of the 6th International Conference on Recherche dInformation Assiste par Ordinateur (RIAO)*, 2000, pp. 302–313.

[6] S. T. Dumais and H. Chen, "Hierarchical classification of Web content," in *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, N. J. Belkin, P. Ingwersen, and M.-K. Leong, Eds. Athens, GR: ACM Press, New York, US, 2000, pp. 256–263.

[7] M. E. Ruiz and P. Srinivasan, "Hierarchical text categorization using neural networks," *Information Retrieval*, vol. 5, no. 1, pp. 87–118, 2002.

[8] A. S. Weigend, E. D. Wiener, and J. O. Pedersen, "Exploiting hierarchy in text categorization," *Information Retrieval*, vol. 1, no. 3, pp. 193–216, 1999.

[9] M. Ceci and D. Malerba, "Classifying web documents in a hierarchy of categories: a comprehensive study," *Journal of Intelligent Information Systems*, vol. 28, no. 1, pp. 37–78, 2007.

[10] A. Esuli, T. Fagni, and F. Sebastiani, "Boosting multi-label hierarchical text categorization," *Inf. Retr.*, vol. 11, no. 4, pp. 287–313, 2008.

[11] N. Japkowicz, "Learning from imbalanced data sets: a comparison of various strategies," in *AAAI Workshop on Learning from Imbalanced Data Sets*, 2000.

[12] G. Armano, "On the progressive filtering approach to hierarchical text categorization," DIEE - University of Cagliari, Tech. Rep., 2009.

[13] A. Addis, G. Armano, and E. Vargiu, "Experimental assessment of a threshold selection algorithm for tuning classifiers in the field of hierarchical text categorization," in *Proceedings of 17th RCRA International Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion*, 2010.

[14] A. Addis, G. Armano, and E. Vargiu, "From a generic multiagent architecture to multiagent information retrieval systems," in *AT2AI-6, Sixth International Workshop, From Agent Theory to Agent Implementation*, 2010, pp. 3–9.

[15] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley and Sons, 2007.