ASSESSING PROGRESSIVE FILTERING TO PERFORM HIERARCHICAL TEXT CATEGORIZATION IN PRESENCE OF INPUT IMBALANCE

Andrea Addis, Giuliano Armano and Eloisa Vargiu
Dept. of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy

Keywords: Hierarchical text categorization.

Abstract: The more the amount of available data (e.g., in digital libraries), the greater the need for high-performance text categorization algorithms. So far, the work on text categorization has been mostly focused on "flat" approaches, i.e., algorithms that operate on non-hierarchical classification schemes. Hierarchical approaches are expected to perform better in presence of subsumption ordering among categories. In fact, according to the "divide et impera" strategy, they partition the problem into smaller subproblems, each being expected to be simpler to solve. In this paper, we illustrate and discuss the results obtained by assessing the "Progressive Filtering" (PF) technique, used to perform text categorization. Experiments, on the Reuters Corpus (RCV1-v2) and on DZMOZ datasets, are focused on the ability of PF to deal with input imbalance. In particular, the baseline is: (i) comparing the results to those calculated resorting to the corresponding flat approach; (ii) calculating the improvement of performance while augmenting the pipeline depth; and (iii) measuring the performance in terms of generalization- / specialization- / misclassification-error and unknown-ratio. Experimental results show that, for the adopted datasets, PF is able to counteract great imbalances between negative and positive examples.

1 INTRODUCTION

In the Web 2.0 age 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 reference taxonomies in an effective and efficient way.

Furthermore, due to the increasing importance of term polysemy for large corpora, both precision and recall decrease as the number of categories increases (Apté et al., 1994) (Yang, 1996), so that considering categories organized in a hierarchy may help to improve the overall performances. Another important issue is concerned with the benefits that a hierarchical approach can give in real-world scenarios, typically characterized by imbalanced data (Kotsiantis et al., 2006). In fact, in these contexts, relevant and non relevant documents (i.e., positive and negative examples, respectively) are typically imbalanced, turning classifiers trained with the same percentage of positive and negative examples into not adequate tools.

This paper is aimed at assessing the effectiveness of the "Progressive Filtering" (PF) technique, applied to text categorization in presence of input imbalance. In its simplest setting, PF decomposes a given rooted taxonomy into pipelines, one for each path that exists between the root and any given node of the taxonomy, so that each pipeline can be tuned in isolation. To this end, a Threshold Selection Algorithm (TSA) has been devised, aimed at finding a sub-optimal combination of thresholds for each pipeline.

The rest of the paper is organized as follows: in Section 2, a brief overview of selected related work on both hierarchical text categorization and input imbalance is presented. Sections 3 describes progressive filtering. In Section 4, the adopted threshold selection algorithm is presented. Experiments and results are illustrated in Section 5. Section 6 ends the paper with conclusions and future work.

Addis A., Armano G. and Vargiu E..
ASSESSING PROGRESSIVE FILTERING TO PERFORM HIERARCHICAL TEXT CATEGORIZATION IN PRESENCE OF INPUT IMBALANCE.
DOI: 10.5220/0003066300140023
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2010), pages 14-23
ISBN: 978-989-8425-28-7
Copyright © 2010 SCITEPRESS (Science and Technology Publications, Lda.)

2 RELATED WORK

2.1 Hierarchical Text Categorization

In the last years several researchers have investigated the use of hierarchies for text categorization, which is also the main focus of our proposal.

Until the mid-1990s researchers mostly ignored the hierarchical structure of categories that occur in several domains. In 1997, Koller and Sahami (Koller and Sahami, 1997) carried out the first proper study on hierarchical text categorization on the Reuters-22173 collection. Documents were classified according to the given hierarchy by filtering them through the single best-matching first-level class and then sending them to the appropriate second level. This approach showed that hierarchical models perform well when a small number of features per class is used. In fact, no advantages were found using the hierarchical model for large numbers of features.

McCallum et al. (McCallum et al., 1998) proposed a method based on naïve Bayes. The authors compare two techniques: (i) exploring all possible paths in the given hierarchy, and (ii) greedily selecting at most two branches according to their probability, as done in (Koller and Sahami, 1997). Results show that the latter is more error prone while computationally more efficient.

Mladenić (Mladenic and Grobelnik, 1998) used the hierarchical structure to decompose a problem into a set of subproblems, corresponding to categories (i.e., the nodes of the hierarchy). For each subproblem, a naïve Bayes classifier is generated, considering examples belonging to the given category, including all examples classified in its subtrees. The classification applies to all nodes in parallel, a document is passed down to a category only if the posterior probability for that category is higher than a user-defined threshold.

D'Alessio (D'Alessio et al., 2000) proposed a system in which, for a given category, the classification is based on a weighted sum of feature occurrences that should be greater than the category threshold. Both single and multiple classifications are possible for each document to be tested. The classification of a document proceeds top-down possibly through multiple paths. An innovative contribution of this work is the possibility of restructuring a given hierarchy or building a new one from scratch.

Dumais and Chen (Dumais and Chen, 2000) 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. The sets of positive and negative examples are built considering documents that belong to categories at the same level, and different feature sets are built, one for each category. Several combination rules have also been assessed.

In the work by Ruiz and Srinivasan (Ruiz and Srinivasan, 2002) a variant of the Hierarchical Mixture of Experts model is used. A hierarchical classifier combining several neural networks is also proposed in (Weigend et al., 1999).

Gaussier et al. (Gaussier et al., 2002) proposed a hierarchical generative model for textual data, where words may be generated by topic specific distributions at any level in the hierarchy. This model is aimed at clustering documents in preset- or (automatically) generated-hierarchies, as well as at categorizing new documents in an existing taxonomy.

In (Rousu et al., 2005) a kernel-based algorithm for hierarchical text classification has been presented. Documents are allowed to belong to more than one category at a time. The classification model is a variant of the Maximum Margin Markov Network framework, where the taxonomy is represented as a Markov tree equipped with an exponential family defined on the edges. Experiments showed that the proposed algorithm can feasibly optimize training sets of thousands of examples and taxonomies consisting of hundreds of nodes.

Ceci and Malerba (Ceci and Malerba, 2007) presented a comprehensive study on hierarchical classification of web documents. They extend a previous work (Ceci and Malerba, 2003) considering: (i) hierarchical feature selection mechanisms; (ii) a naïve Bayes algorithm aimed at avoiding problems related to different document lengths; (iii) the validation of their framework for a probabilistic SVM-based classifier; and (iv) an automated threshold selection algorithm.

More recently, in (Esuli et al., 2008), the authors proposed a multi-label hierarchical text categorization algorithm consisting of a hierarchical variant of ADABOOST.MH, a well-known member of the family of "boosting" learning algorithms.

Bennett et al. (Bennett and Nguyen, 2009) studied the problem of the error propagation under the assumption that a mistake is made at "high" nodes in the hierarchy, as well as the problem of dealing with increasingly complex decision surfaces.

Brank et al. (Brank et al., 2010) dealt with the problem of classifying textual documents into a topical hierarchy of categories. They construct a coding matrix gradually, one column at a time, each new column being defined in a way that the corresponding binary classifier attempts to correct the most common mistakes of the current ensemble of binary classifiers.

The goal is to achieve good performance while keeping reasonably low the number of binary classifiers.

2.2 The Input Imbalance Problem

Japkowicz (Japkowicz, 2000) studied the class imbalance problem. In particular, for binary classification, the author studied the problem related to domains in which 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 (Kotsiantis et al., 2006). The problem of imbalance has got more and more emphasis in recent years. Imbalanced data sets exist in many real-world domains, such as spotting unreliable telecommunication customers, detection of oil spills in satellite radar images, learning word pronunciations, text classification, detection of fraudulent telephone calls, information retrieval and filtering tasks, and so on (Wu and Chang, 2003) (Yan et al., 2003). 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 (Kubat and Matwin, 1997) (Chawla et al., 2002) (Kotsiantis and Pintelas, 2003). 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.

3 PROGRESSIVE FILTERING IN TEXT CATEGORIZATION

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 (Armano, 2009) for further details.

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).

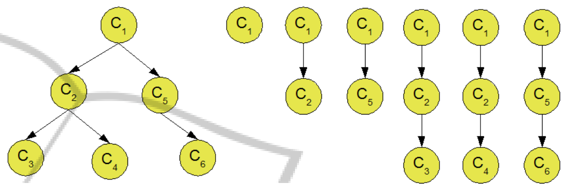

Figure 1: A taxonomy and its corresponding pipelines.

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.

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 it as positive, the token is passed 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.

Finally, let us note that the proposed approach performs a sort of “flattening” though *preserving* the information about the hierarchical relationships embedded in a pipeline. For instance, the pipeline $\langle C_1, C_2, C_3 \rangle$ actually represents the classifier C_3, although the information about the existing subsumption relationships, i.e., $C_3 \leq C_2 \leq C_1$, is preserved.

4 THE THRESHOLD SELECTION ALGORITHM

As we know from classical text categorization, given a set of documents D and a set of labels C, 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. Let us recall that, given $d \in D$, $CSV_i(d) \geq \theta_i$ is interpreted as a decision to categorize d under c_i, whereas $CSV_i(d) < \theta_i$ is interpreted as a decision not to categorize d under c_i.

In PF, let us still assume that CSV_i exists for each $c_i \in C$, with the same semantics adopted in 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: $\forall k = 1..n, CSV_i(d) \geq \theta_k$. On the contrary, d is not categorized under c_i in the event that a classifier in π rejects it. In so doing, a 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 have the same threshold in π_1 and in π_2 (the same holds for C_2).

In PF, given an utility function[1], we are interested in finding an effective (and computationally "light") way to reach a sub-optimum in the task of determining the best vector of thresholds. To this end, for each pipeline π a sub-optimal combination of thresholds is searched for, considering the actual ratio between positive and negative examples, which in turn depends on the given scenario. Unfortunately, finding the best acceptance thresholds is a difficult task. In fact, exhaustively trying each possible combination of thresholds (brute-force approach) is unfeasible, the number of thresholds being virtually infinite. However, the brute-force approach can be approximated by defining a granularity step that requires to assess only a finite number of points in a range $[0,1]$ in which the thresholds are permitted to vary with step δ. This "relaxed" brute force algorithm (RBF for short) for calibrating thresholds would be very helpful in the task of finding a sub-optimal solution, although *still* too heavy from a computational point of view. Thus, to perform PF we adopt a Threshold Selection Algorithm (namely, TSA) devised to deal with this problem, which maintains the capability of finding a near-optimum solution characterized by a low time complexity (Addis et al., 2010).

Searching for a 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 concerns with the given scenario.

Bearing in mind that the lower the threshold the less restrictive the classifier, we can build a greedy bottom-up algorithm for selecting decision threshold 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 and maintains 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, TSA is then defined as follows (all thresholds are initially set to zero):

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

which indicates that $\mathcal{C}$ is applied to each node of the pipeline, starting from the leaf ($k = L$).

Under the assumption that p is a structure that contains all information about a pipeline, including the corresponding vector of thresholds and the utility function to be optimized, the pseudo-code of TSA is:

```
function TSA(p:pipeline):
  for k:=1 to p.length
    do p.thresholds[i] = 0
  for k:=p.length downto 1
    do Calibrate(up,p,k)
  return p.thresholds
end TSA
```

The *Calibrate* function is defined as follows:

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

and

[1]Different utility functions (e.g., precision, recall, F_β, user-defined) can be adopted, depending on the constraint imposed by the underlying scenario.

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

where the $+$ operator actually denotes a sequence operator (meaning that in the formula $a+b$ action a is performed *before* action b). In pseudo-code:

```
function Calibrate(dir:{up,down}, p:pipeline,
                   level:integer):
  Repair(dir,p,level)
  if level < p.length
  then Calibrate(toggle(dir),p,level+1)
end Calibrate
```

where *toggle* is a function that reverses the current direction, and *Repair* is defined as:

```
function Repair(dir:{up,down}, p:pipeline,
                level:integer):
  delta := (dir = up) ? p.delta : -p.delta
  best_threshold := p.thresholds[level]
  max_uf := p.utility_function()
  uf := max_uf
  while uf >= max_uf * 0.8 and
        p.thresholds[level] in [0,1]
    do p.thresholds[level] :=
       p.thresholds[level] + delta
       uf := p.utility_function()
       if uf < max_uf then continue
       max_uf := uf
       best_threshold := p.thresholds[level]
  p.thresholds[level] := best_threshold
end Repair
```

The factor 0.8, experimentally calculated, is used to limit the impact of local minimums during the search.

After calculating 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, as noted in (Lewis, 1995), different goals for the system lead to different optimal behaviors.

As for the time complexity of TSA, to simplify the problem, let us define a granularity step that requires to visit only a finite number of points in a range $[\rho_{min}, \rho_{max}]$, $0 \leq \rho_{min} < \rho_{max} \leq 1$, in which the thresholds could vary with step δ. Therefore, $p = \lfloor \delta^{-1} \cdot (\rho_{max} - \rho_{min}) \rfloor$ being the maximum number of points to be checked for each classifier in a pipeline and L being the length of the pipeline, the average time of TSA, $T(TSA)$, is proportional to $(L + L^2) \cdot p \cdot (\rho_{max} - \rho_{min})$, which implies that TSA has complexity $O(L^2)$. Hence, the time complexity is quadratic with the number of classifiers embedded by a pipeline. As already noted, a comparison between TSA and the brute-force approach is unfeasible, the elements of the threshold vector being real numbers. Nevertheless, experimental results show that the effectiveness of TSA is almost identical to the one obtained by running *RBF*, in which only p points are checked for each classifier in a pipeline. Note that the average time of *RBF*, $T(RBF)$, is proportional to p^L, which in turns implies that its computational complexity is $O(p^L)$.

It is also important to note that, due to the enormous computational time that can be reached, the RBF approach should be applied in practice only by setting p to a value much lower than the one applicable with TSA. For instance, with a ratio of 10^{-2}, i.e., $p_{RBF} = 20$ and $p_{TSA} = 2000$, $T(RBF) \propto 160,000$ and $T(TSA) \propto 20,000$. In other words, TSA is still 8 times faster than RBF, even considering $\rho_{max} = 1$ and $\rho_{min} = 0$, while ensuring a better result due to the higher granularity.

5 EXPERIMENTS AND RESULTS

Experiments have been performed to validate the proposed approach with respect to the impact of PF in the input imbalance. To this end, three series of experiments have been performed: first, performances calculated resorting to PF have been compared with those calculated by resorting to the corresponding flat approach. Then, PF has been tested to assess the improvement of performances while augmenting the pipeline depth. Finally, performances have been calculated in terms of generalization- / specialization- / misclassification-error and unknown-ratio.

Furthermore, to show that the overall performances of PF are not worsened by the adoption of TSA vs. RBF, we performed further experiments aimed at comparing the performance of PF obtained by applying TSA vs. those obtained by applying RBF.

We are also currently performing new experiments aimed at comparing the proposed approach with state-of-the-art systems and techniques. Nevertheless, as the paper is more concerned with studying how input imbalance affects performance in text categorization, we decided to focus on experiments that compare the effectiveness of the proposed hierarchical approach vs. the flat one.

Experiments have been performed by customizing for this specific task X.MAS (Addis et al., 2008), a generic multiagent architecture built upon JADE (Bellifemine et al., 2007), devised to make easier the implementation of information retrieval and information filtering applications.

5.1 The Adopted Datasets

The two corpora selected for this study are the benchmark datasets Reuters Corpus Volume I (RCV1-v2) (Lewis et al., 2004) and DMOZ, the collection of HTML documents referenced in a web directory developed in the Open Directory Project (ODP). The corpora differ considerably in the size of the training set, in the hierarchical structure of categories, as well as in the procedure adopted for the classification of documents. For the sake of completeness, a brief description of such collections is reported.

As for the Reuters corpora, we adopted the second version of the Reuters Corpus dataset (RCV1-v2) (Lewis et al., 2004). In this corpus, stories have been coded into four hierarchical groups: Corporate/Industrial (CCAT), Economics (ECAT), Government/Social (GCAT), and Markets (MCAT). Although the complete list consists of 126 categories, only part of them were used for testing the proposed hierarchical approach. The total number of codes actually assigned to the data is 93, whereas the overall number of documents is about 803,000, each document belonging to at least one category and (on average) to 3.8 categories.

As for DMOZ[2], to be able to explore the effects of the taxonomy depth on PF performances, we selected 17 categories with a maximum depth of 5. In so doing, each category includes at least 300 documents. The corresponding training set includes a subset of the 136,000 documents, each belonging to one category.

5.2 Evaluation Measures

To evaluate the performances of PF, several measures have been considered.

First, the approach is validated resorting to a conventional metric for calculating the performances of a text categorization system –i.e., macro-averaging. Macro-averaged scores are calculated by first calculating precision and recall for each category and then taking their average. In other words, macro-averaging first evaluates P and R "locally" for each category, and then "globally" by averaging over the results of the different categories. In symbols:

$$P^M = \frac{\sum_{i=1} mP_i}{m} \tag{5}$$

$$R^M = \frac{\sum_{i=1} mP_i}{m} \tag{6}$$

where the "M" superscript stands for macroaveraging.

[2]http://www.dmoz.org

To evaluate the performances of the approach, we also applied F_1, obtained from the general definition of F_β by imposing the same degree of importance for P and R. In symbols:

$$F_\beta = \frac{(1+\beta^2)P \cdot R}{\beta^2 P + R} \tag{7}$$

$$F_1 = F_{\beta=1} = \frac{2PR}{P+R} \tag{8}$$

Then, the approach is validated using four additional evaluation measures, defined in (Ceci and Malerba, 2007):

- misclassification error, i.e., the percentage of documents misclassified in a category not related to the correct category in the hierarchy;
- generalization error, i.e., the percentage of documents misclassified in a supercategory of the correct one;
- specialization error, i.e., the percentage of documents misclassified in a subcategory of the correct one;
- unknown ratio, i.e., the percentage of rejected documents.

5.3 Results

The main issue being investigated is the effectiveness of the proposed approach with respect to flat classification. In order to make a fair comparison, the same classification system has been adopted, i.e., a classifier based on the *wk*-NN technology (Cost and Salzberg, 1993). The motivation for the adoption of this particular technique stems from the fact that it does not require specific training and is very robust with respect to noisy data. In fact, as demonstrate in (Takigawa et al., 2005) *wk*-NN-based approaches can reduce the error rate due to robustness against outliers.

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

[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.

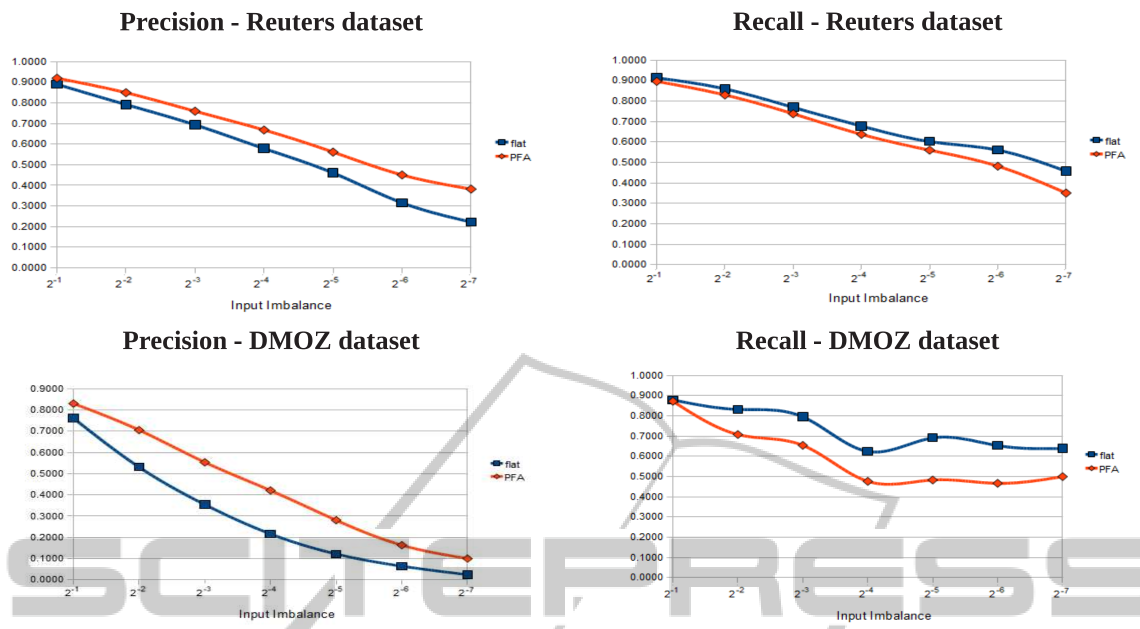

Figure 2: Comparison of precision and recall between PF and flat classification.

Experiments have been performed by assessing the behavior of the proposed hierarchical approach in presence of different ratios 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. Nevertheless, as for DMOZ, let us note that these results depend 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. 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, it depends on the overall number of false negatives (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 measure. As for the specialization-error, it depends on the overall number of false positives (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. As final remark, let us note that different utility functions can be adopted, depending on which aspect or unwanted effect one wants to improve or mitigate. For instance, recall could be adopted as utility function to reduce the overall number of

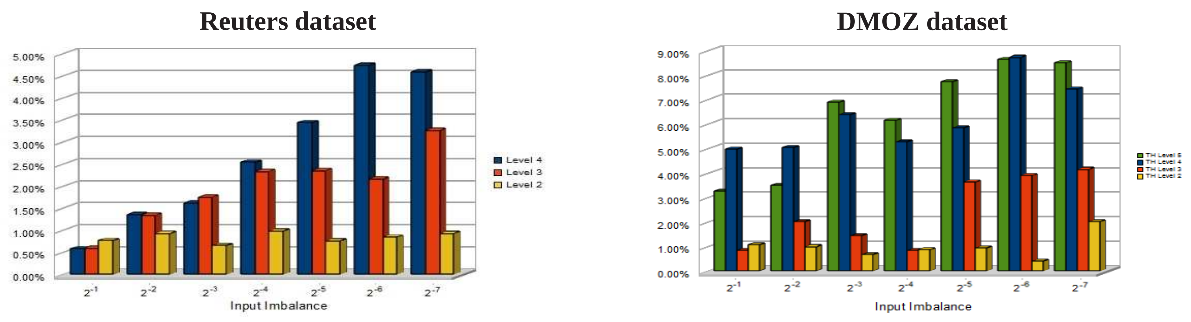

Figure 3: Performance improvement.

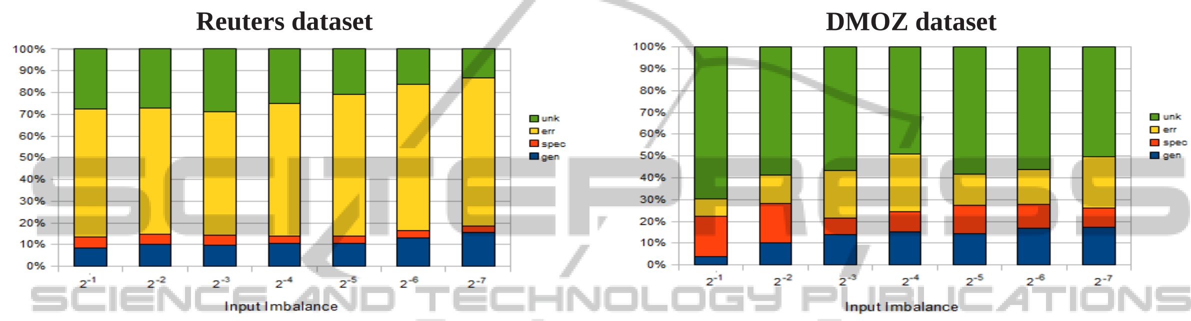

Figure 4: Hierarchical measures.

Table 1: TSA vs. RBF: F_1 in presence of input imbalance.

Input imbalance	$F_1(RBF)$	$F_1(TSA)$
2^{-1}	0.830	0.835
2^{-2}	0.722	0.733
2^{-3}	0.619	0.632
2^{-4}	0.497	0.515
2^{-5}	0.404	0.428
2^{-6}	0.323	0.345
2^{-7}	0.245	0.273

FNs. In fact, optimizing recall implies to accept as positive documents that are in fact true negatives (TNs). In this way, the unknown-ratio (which depends on the amount of FNs) decreases, whereas the misclassification-error (which depends on the amount of FPs) increases.

TSA vs. RBF. Experiments have been performed by assessing the behavior of PF in terms of F_1 in presence of different ratios of positive examples vs. negative examples –i.e., from 2^{-1} to 2^{-7}. Results, summarized in Table 1, show that the performance of TSA is always better than the one obtained with RBF. This is due to the fact that, as previously pointed out, TSA has been used with a higher granularity (i.e., $p_{RBF}/p_{TSA} = 20/2000 = 10^{-2}$).

6 CONCLUSIONS AND FUTURE WORK

In this paper, we made experiments on PF to investigate how the ratio between positive and negative examples affects the performances of a classifier system and how these performances can be improved by adopting PF instead of a classical flat approach. Results show that the proposed approach is able to deal with high imbalance between negative and positive examples.

As for the future work, several issues are currently under investigation. In particular, we are currently performing new experiments aimed at comparing the proposed approach with state-of-the-art systems and techniques. Moreover, we are interested in: (i) investigating the whole taxonomy instead of the corresponding set of pipelines; (ii) adopting and calculating further metrics to assess the performances of PF; (iii) testing PF on further datasets, such as TREC or MeSH.

REFERENCES

Addis, A., Armano, G., and Vargiu, E. (2008). From a generic multiagent architecture to multiagent information retrieval systems. In *AT2AI-6, Sixth International*

Workshop, From Agent Theory to Agent Implementation, pages 3–9.

Addis, A., Armano, G., and Vargiu, E. (2010). 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*.

Apté, C., Damerau, F., and Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251.

Armano, G. (2009). On the progressive filtering approach to hierarchical text categorization. Technical report, DIEE - University of Cagliari.

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

Bennett, P. N. and Nguyen, N. (2009). Refined experts: improving classification in large taxonomies. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18, New York, NY, USA. ACM.

Brank, J., Mladenic, D., and Grobelnik, M. (2010). Large-scale hierarchical text classification using svm and coding matrices. In *Large-Scale Hierarchical Classification Workshop*.

Ceci, M. and Malerba, D. (2003). Hierarchical classification of HTML documents with WebClassII. In Sebastiani, F., editor, *Proceedings of ECIR-03, 25th European Conference on Information Retrieval*, pages 57–72. Berlin Heidelberg NewYork: Springer.

Ceci, M. and Malerba, D. (2007). Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):37–78.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Cost, W. and Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

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

Dumais, S. T. and Chen, H. (2000). Hierarchical classification of Web content. In Belkin, N. J., Ingwersen, P., and Leong, M.-K., editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, GR. ACM Press, New York, US.

Esuli, A., Fagni, T., and Sebastiani, F. (2008). Boosting multi-label hierarchical text categorization. *Inf. Retr.*, 11(4):287–313.

Gaussier, E., Goutte, C., Popat, K., and Chen, F. (2002). A hierarchical model for clustering and categorising documents. In *Proceedings of the 24th BCS-IRSG European Colloquium on IR Research*, pages 229–247, London, UK. Springer-Verlag.

Japkowicz, N. (2000). Learning from imbalanced data sets: a comparison of various strategies. In *AAAI Workshop on Learning from Imbalanced Data Sets*.

Koller, D. and Sahami, M. (1997). Hierarchically classifying documents using very few words. In Fisher, D. H., editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 170–178, Nashville, US. Morgan Kaufmann Publishers, San Francisco, US.

Kotsiantis, S., Kanellopoulos, D., and Pintelas, P. (2006). Handling imbalanced datasets: a review. *GESTS International Transactions on Computer Science and Engineering*, 30:25–36.

Kotsiantis, S. and Pintelas, P. (2003). Mixture of expert agents for handling imbalanced data sets. *Ann Math Comput Teleinformatics*, 1:46–55.

Kubat, M. and Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *In Proceedings of the Fourteenth International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann.

Lewis, D., Yang, Y., Rose, T., and Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Lewis, D. D. (1995). Evaluating and optimizing autonomous text classification systems. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 246–254, New York, NY, USA. ACM.

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

Mladenic, D. and Grobelnik, M. (1998). Feature selection for classification based on text hierarchy. In *Text and the Web, Conference on Automated Learning and Discovery CONALD-98*.

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. (2005). Learning hierarchical multi-category text classification models. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 744–751, New York, NY, USA. ACM.

Ruiz, M. E. and Srinivasan, P. (2002). Hierarchical text categorization using neural networks. *Information Retrieval*, 5(1):87–118.

Takigawa, Y., Hotta, S., Kiyasu, S., and Miyahara, S. (2005). Pattern classification using weighted average patterns of categorical k-nearest neighbors. In *Proceedings of the 1th International Workshop on*

Camera-Based Document Analysis and Recognition, pages 111–118.

Weigend, A. S., Wiener, E. D., and Pedersen, J. O. (1999). Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216.

Wu, G. and Chang, E. Y. (2003). Class-boundary alignment for imbalanced dataset learning. In *In ICML 2003 Workshop on Learning from Imbalanced Data Sets*, pages 49–56.

Yan, R., Liu, Y., Jin, R., and Hauptmann, A. (2003). On predicting rare classes with svm ensembles in scene classification. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 3, pages III–21–4 vol.3.

Yang, Y. (1996). An evaluation of statistical approaches to medline indexing. In *Proceedings of the American Medical Informatic Association (AMIA)*, pages 358–362.