MADSPAM Consortium at the ECML/PKDD Discovery Challenge 2010

Artem Sokolov,
Tanguy Urvoy
Orange Labs
Lannion, France
gposhta@gmail.com
tanguy.urvoy@orange-ftgroup.com

Ludovic Denoyer
Univ. Pierre et Marie Curie
Paris, France
ludovic.denoyer@lip6.fr

Olivier Ricard
BlogSpirit
Paris, France
olivier.ricard@blogspirit.fr

ABSTRACT

We present here the contribution of the MADSPAM consortium to the ECML/PKDD Discovery Challenge 2010. The submitted method is based on both a RankBoost algorithm and on propagation techniques.

1. INTRODUCTION

The ECML/PKDD Discovery Challenge 2010 was focused on Web content quality. It involved three different tasks:

- task 1 was a categorization task (for English sites);
- task 2 was a quality ranking task (for English sites);
- task 3 was a multilingual transfer from task 2 for German and French sites.

These tasks involved 10 different categories: *spam, news, commercial, educational, discussion lists, personal, neutral, biased, trust* and *quality*. These categories where not independent (for instance *spam* vs. *trust* or *neutral* vs. *biased*). For each category, the teams had to provide a global ordering of the sites with highly ranked sites first. These ordering were evaluated with NDCG, an information retrieval metric which emphasizes the top of lists.

The challenge dataset [1] was very rich: it included 85 compressed text files providing a wide variety of information about web sites including training labels, URLs and hyperlinks, several content-based and link-based features, term frequencies and weightings, natural language processing features and many others.

2. PROPOSED APPROACH

2.1 Formal definition of the problem

We denote $\mathcal{I}$ the set of host instances. This set is partitioned in two subsets $\mathcal{I}_{train}$ and $\mathcal{I}_{test}$, respectively for training and testing. We consider also a set C of categories ($|C| = 10$ in this challenge). Reference scores y_i^c are indexed by training instance $i \in \mathcal{I}_{train}$ and category $c \in C$. These scores are known for the training hosts only. While the goal of the challenge was to provide a rank for each remaining host and for each category, we consider a task where we predict a numerical score $\hat{y}_i^c$ for all test hosts $i \in \mathcal{I}_{test}$. The final ranking for category c is derived by sorting all the testing instances by decreasing scores $\hat{y}_i^c$. To lighten the notations, we drop the c superscripts when the category is clear from context.

2.2 Outline of the approach

To combine efficiently the relational information and the instance-based information, we used a semi-transductive two-steps approach:

- the first step, described in section 2.3, was inductive: we trained a ranking model on instance-based features;
- the second step, described in section 2.4, was transductive: we used the relational data to regularize the predictions of the ranking model of the first step.

For the first step we built independent models for each category, for the second one we also tested a global multi-categories discriminant model. Our experiments and results are described in section 3.

2.3 Instance-based model

As a baseline we used the pair-wise ranking algorithm RankBoost [5] to train a ranking model for each category. Each host instance i is assigned a vector of features $\boldsymbol{x_i} = (x_i^1, \ldots, x_i^d)$. These features are composed of the information extracted from both the content of the hosts and the relations between them. In the experiments section 3, we describe different possible set of features.

For a given number of training steps T the RankBoost algorithm learns a scoring function H which is a linear combination of "simple" functions h_t called weak learners. The final ranking function is defined as:

$$H(\boldsymbol{x_i}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x_i}) \ ,$$

where each α_t is the weight assigned to the weak function h_t at step t of the learning process. We used the simplest family of weak learners – decision stumps that depend on one feature only:

$$h(\boldsymbol{x_i}) = [\![x_i^k > \theta]\!] \ ,$$

where k is the selected feature index and θ is a learnt threshold. These learners were trained using the approximate "3-rd method" described in [5].

In order to learn the final scoring function H, we minimize a convex approximation of the pair-wise loss $L(H)$:

$$L(H) = \sum_{i,j \in \mathcal{I}_{train} \,:\, y_i < y_j} D(i,j) [\![H(\boldsymbol{x_i}) \geq H(\boldsymbol{x_j})]\!].$$

The positively-valued preference matrix D encodes the orderings observed in the training set: the higher the value of $D(i,j)$ is the more important it is to preserve the order of hosts i and j. If D is chosen constant, the optimized criterion is Kendall τ (equivalent to AUC when labels are binary). For this challenge, to give more emphasis on score differences, we choose $D(i,j) = \frac{y_j - y_i}{Z_D}$, where Z_D is a global normalization factor.

2.4 Propagation schemes

While the RankBoost algorithm computes a score for each host and each category using (among others) precomputed relation-based features (section 3.1.4), it does not directly consider inter-dependencies between the predicted scores of the different hosts. We present here two methods to make use of the graph structure information that aim at propagating both the real scores of the training hosts and the predicted scores of the testing hosts through inter-host links:

- The first one (section 2.4.1) is based on the minimization of a **regularized loss function** and is very similar to the work presented in [4].
- The second one (section 2.4.2) is based on the **iterative algorithm** proposed in [7] and aims at learning a propagation scheme that learns and uses correlations between different labels resulting in a more complex propagation scheme.

In the following, we will not consider the content of the graph nodes (host instances), and will denote $\hat{y}_i^c$ the instance-based score predicted by the method described in section 2.3, i.e. $\hat{y}_i^c = H^c(\boldsymbol{x_i})$ where H^c is the RankBoost model learned for category c. We consider that $w_{i,j}$ is the weight of the link between instances i and j. This weight is equal to 0 if there is no edge between two hosts. We denote z_i^c the score of node i for category c obtained by using one of the propagation methods presented below.

2.4.1 *Regularization-based propagation*

We define a loss function based on two assumptions. First, a good final ranking of nodes is a ranking that is close the ranking resulting from the instance-based model. Second, it has to associate close ranks to connected nodes. This second assumption is called smoothness and has been studied in different recent works.

For a category c, the loss is defined as:

$$L^c = \sum_{i \in \mathcal{I}} (z_i^c - \hat{y}_i^c)^2 + \lambda \sum_{i,j \in \mathcal{I}} w_{i,j} (z_i^c - z_j^c)^2.$$

The first term corresponds to the first assumption while the second is the smoothness regularization term. Parameter λ corresponds to the balance between the two hypothesis and is tuned by cross validation.

```
1: $\forall i, z_i^c \leftarrow \hat{y}_i^c$
2: **repeat**
3:     Choose node $x_i$ from the testing set randomly
4:     $z_i^c \leftarrow \frac{\sum_j \lambda w_{i,j} z_j^c + 2\hat{y}_i^c}{\sum_j \lambda w_{i,j} + 2}$
5: **until** convergence
```

Figure 1: Inference algorithm for the regularization-based propagation model.

```
1: **repeat**
2:     Choose node $x_i$ from the testing set randomly
3:     $z_i^c \leftarrow g_{\theta_c}(i)$
4: **until** convergence
```

Figure 2: Inference algorithm for the iteration-based propagation model. For each category c, g_{θ_c} is learned from the set of training nodes.

With such a loss function, the final scores $\hat{y}_i^{c*}$ are computed as:

$$(z_1^{c*}, ..., z_{|\mathcal{I}|}^{c*}) = argmin_{z_1^c, ..., z_{|\mathcal{I}|}^c} L^c.$$

Different optimization methods for minimizing the loss can be used. We propose here to use a coordinate gradient descend algorithm that proceeds by iteratively computing the score of a node from the scores of its neighbors. The algorithm is described in Figure 1.

2.4.2 *Iteration-based propagation*

The iteration-based propagation is a method presented in Figure 2 which relies on fewer assumptions than the regularization-based method presented previously. The idea is to learn a discriminant propagation scheme from the training nodes and then to apply it iteratively on the whole graph. Instead of considering the smoothness assumption, such a method aims at automatically deducing how labels propagate on the graph structure. It has two main advantages: first it is based on less constraints than the regularization model and is thus more expressive, second, it is able to handle complex propagation schemes that correspond to propagation of scores between different categories, capturing correlations between the different classes, while the previous model considers different categories as separate problems. The iteration-based approach has recently shown good performances on the problem of the annotation of nodes in a multi-relational graph [7].

For this model, we consider a regression function g_{θ_c} which corresponds to the propagation scheme of scores for category c. This function depends on a set of parameters θ_c that will be learnt from the training nodes. Basically, this model not only uses the scores of the neighboring nodes for category c, but also both:

- the scores of the neighboring nodes for other categories, trying to learn a propagation between different categories – e.g., *spam* scores propagates to *trust* scores
- the scores of the considered node for other categories,

trying to learn "rules" such as: *if I am spam, then my trust score is low.*

This information is handled through a vectorial representation of the node i denoted by $\Phi(i,c)$ which is composed of the previously defined information – $\hat{y}_i^c$ scores, current z_i^c scores of the node and its neighbors. Due to lack of space, Φ is not described in this paper and we refer to [7] for a detailed description of the method. The propagation function used here is a linear function $g_{\theta_c}(i) = \langle \theta_c ; \Phi(i,c) \rangle$ learned by minimizing a classical pairwise ranking loss.

3. EXPERIMENTS

We made a 5-fold cross validation using $4/5^{th}$ of the 2449 provided labels to train rankers and kept $1/5^{th}$ of labels to evaluate and calibrate the models.

3.1 Datasets

3.1.1 *Labels*

Training labels for Task 2 and 3 were extracted from the v2-en.labels-unified.csv file. In the experiments we didn't use any special handling of German and French languages, the model learned on English labels was used on two other languages (no quality labels were provided for other languages in the collection). A larger set of labels was also obtained by collecting all labels from $*$.useful-labels.csv and recalculating the quality with the formula provided by the organizers (was used with nPR submission, see section 3.1.4).

3.1.2 *Relational data*

The provided link-graph was weighted by the number of links between hosts. To enhance this graph, we reweighted the edges according to textual similarity:

$$w_{i,j} = n_{i,j} \cdot (\delta + (1-\delta) \cos_{tfidf}(\boldsymbol{x_i}, \boldsymbol{x_j})) \ , \qquad (1)$$

where $n_{i,j}$ is the number of links between two hosts $i, j \in \mathcal{I}$ and $\delta \in [0,1]$ is a parameter tuned to balance the relative importance of Web-links and textual content similarity.

3.1.3 *Instance-level data*

The instance-level features we used were of three types:

- 96 content-based synthetic features;
- 176 link-based synthetic features;
- a vocabulary of around $70K$ words weighted by $tfidf$.

As some hosts did not have features, we completed the features with a w-weight-aware version of the PageRank [2] for 10 different values of $\delta = 0, 0.1, \ldots, 0.9$ in (1) and 100 propagation steps on an unlabeled graph (www-hosts and nowww-hosts were not merged):

$$R^*(j) = \delta_f \sum_i \frac{R^*(j) w(i,j)}{w(i,\cdot)} + (1-\delta_f) R_0 \ , \qquad (2)$$

where $w(i,\cdot) = \sum_l w(i,l)$ is a weighted out-degree normalization factor, $R_0 = 1/|\mathcal{I}|$, and $\delta_f = 0.85$ as suggested in [2].

3.1.4 *Feature sets*

To train RankBoost we used several feature sets:

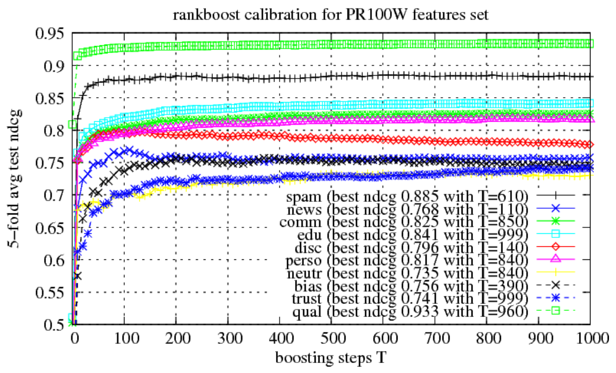

Figure 3: Boosting steps calibration: a rather stable performance.

1. "basic set": all content- and the link-based features provided by the organizers concatenated, amounting totally to 272 numerical features.
2. "PR set": basic set augmented with the 10 more features graph obtained using the formula (2) for values of $\delta = 0, \ldots, 0.9$.
3. "nPR set": PR set but labeled with the larger set of labels, as explained above.
4. "PR1K set": PR set with the set of $tfidf$ values for one thousand words having the largest $tfidf$ values.
5. "PR100W set": PR set with the set of $tfidf$ values for one hundred most frequent words for each host.
6. "PRall set": PR set with the set of $tfidf$ values for about $70K$ words, covering almost full vocabulary of the training set.

The number of training steps T was fixed by cross-validation (c.f. Fig. 3).

3.2 Results

3.2.1 *Selected features*

The sum of RankBoost weights for feature f is denoted α^f. It estimates the contribution of this feature to the global prediction. Table 2 gives the most important features according to this criterion for the *spam* subtask.

Feature `top_1000_query_prec_hp` is one of the spam features defined in [3], it evaluates the rate of top-query words in a Web-site: this is a strong Web-spam indicator because spammers often target the most frequent queries. On the other hand, `top_500_corpus_prec_avg` evaluates the rate of frequent words in a site: this is a good non-spam indicator as spammers often neglect to add frequent grammatical words in their forgery [6].

Features starting with `PR_D` are the weighted-PageRank features as detailed in section 3.1.2 for different values of δ. The traditional – mostly links-based – PageRank (like `PR_D0.80`) is a good spam indicator because this ranking criterion is often over optimized by spammers. On the other hand, the content sensitive PageRank (like `PR_D0.20`) remains a good quality indicator.

feature	α^f
top_1000_query_prec_hp	1.99
PR_D0.80	1.29
top_100_query_prec_hp	1.16
“http”	1.05
frac_visible_hp	1.03
...	
PR_D0.20	-0.66
“money”	-0.92
compress_rate_hp	-0.96
avg_length_avg	-1.25
top_500_corpus_prec_avg	-1.57

Table 2: Most informative features for spam according to RankBoost cumulative weighting. A negative α^f means a negative contribution.

3.2.2 Propagation models

The regularization-based model described in section 2.4.1 used the scores computed by the RankBoost algorithm using the PRall set of features. We have used different values of the λ and δ parameters. The best results presented here have been obtained with $\lambda = 0.01$ and $\delta = 0.9$. These results clearly show the good influence of the propagation model in comparison to the instance-based model. The iteration scheme also obtained its best performances on the $\delta = 0.9$ graph. We can see that it outperforms the regularization-based propagation model on the full set. This shows the ability of this model to handle complex propagation schemes that are not considered by the regularization model which is based on a too strong assumption. Particularly, this model is able to learn correlations between the different categories.

4. CONCLUSION

We have described the three models submitted to the Discovery Challenge 2010 by the MADSPAM consortium. The first model is a classical RankBoost method, while the two other methods are able to handle the dependencies between the predicted scores. Mainly, we have proposed a classical regularization-based propagation model and also an iterative algorithm able to handle more complex propagation schemes. The results presented show the effectiveness of the approaches. All these models are able to handle large scale datasets and many different categories. Moreover, all the selected features and keywords for the categories well correspond to the intuition (section 3.2.1) and show that these models can be used to understand different tasks. The quality of Web-pages was evaluated with NDCG. Because the quality is not query-dependent, it could be interesting, as in [8], to use a non-discounted metric. To go further with query-dependence, if we consider the quality of a Web-site as its ability to be frequently a good answer to popular queries, the best way to evaluate static ranking is probably to use average query-level NDCG. Static and dynamic ranking algorithms could then be evaluated on the same ground.

	method	**task 1**	**task 2** (en)	**task 3** (de)	**task 3** (fr)
Rank-Boost	basic	0.6387	0.9277	0.8378	0.8372
	PR	0.5744	0.9303	0.8504	0.8332
	nPR	0.6215	0.9188	0.8453	0.8252
	PR1K	0.6497	**0.9389**	0.8125	**0.8415**
	PR100W	0.6367	0.9302	0.8465	0.8296
Propa-gation	Reg/PRall	**0.7019**	0.8166	**0.8520**	0.7969
	Iter/PR1K	0.6592	0.9299	0.8347	0.8232

Table 3: Results on the validation subset.

	method	**task 1**	**task 2** (en)	**task 3** (de)	**task 3** (fr)
Rank-Boost	basic	0.6571	0.9048	0.7969	0.8212
	PR	0.6191	0.9164	0.8030	0.8184
	nPR	0.6318	0.9108	0.8157	0.8203
	PR1K	0.6489	0.9228	**0.8207**	**0.8455**
	PR100W	0.6319	0.9183	0.8049	0.8244
Propa-gation	Reg/PRall	0.6956	0.8350	0.8032	0.7940
	Iter/PR1K	**0.7010**	**0.9233**	0.8157	0.8362

Table 4: Results on the full test set.

Acknowledgments

We would like to thank the organizers for this challenge. This work was supported by MADSPAM: ANR project №2007TLOG01301.

5. REFERENCES

[1] A. A. Benczúr, C. Castillo, M. Erdélyi, Z. Gyöngyi, J. Masanes, and M. Matthews. ECML/PKDD 2010 Discovery Challenge Data Set. Crawled by the European Archive Foundation, 2010.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW '98*, 1998.

[3] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: web spam detection using the web topology. In *SIGIR '07*, 2007.

[4] L. Denoyer and P. Gallinari. A ranking based model for automatic annotation in a social network. In *ICWSM 2010 (Short Paper)*, 2010.

[5] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

method	**spam**	**news**	**comm**	**educ**	**disc**	**pers**	**neut**	**bias**	**trus**
basic	**0.8097**	0.6324	0.7500	0.7824	0.7635	0.7306	0.5111	0.4623	0.4714
PR	0.6817	0.5917	0.7578	0.8035	0.6826	0.6561	0.4064	0.5521	0.4404
nPR	0.7636	0.6342	0.7839	0.8091	0.7037	0.7415	0.4334	0.3632	0.4533
PR1K	0.6766	0.6846	0.8162	**0.8797**	0.7595	0.6556	0.4625	0.4962	0.4092
PR100W	0.7687	0.6333	0.8088	0.8227	0.6983	0.7366	0.4251	0.3503	0.4434
Reg/PRall	0.7859	0.6468	0.8254	0.8333	**0.7771**	0.7654	**0.5631**	**0.5682**	0.4955
Iter/PR1K	0.7761	**0.7369**	**0.8299**	0.8593	0.7390	**0.7676**	0.5472	0.5509	**0.5016**

Table 1: Per category results for task 1 on the full test set.

[6] T. Lavergne, T. Urvoy, and F. Yvon. Filtering artificial texts with statistical machine learning techniques. *Language Resources and Evaluation*, 2010.

[7] S. Peters, L. Denoyer, and P. Gallinari. Iterative annotation of multi-relational social networks. In *ASONAM 2010*, 2010.

[8] M. Richardson, A. Prakash, and E. Brill. Beyond pagerank: Machine learning for static ranking. In *In WWẂ06*, pages 707–715. ACM Press, 2006.