ECML 2007 PKDD
WARSAW POLAND

THE 18TH EUROPEAN CONFERENCE ON MACHINE LEARNING
AND
THE 11TH EUROPEAN CONFERENCE ON PRINCIPLES AND PRACTICE
OF KNOWLEDGE DISCOVERY IN DATABASES

PROCEEDINGS OF THE GRAPH LABELLING WORKSHOP AND WEB SPAM CHALLENGE

GRAPHLAB'07

September 17, 2007

Warsaw, Poland

Editors:
Carlos Castillo
Yahoo! Research, Spain
Brian D. Davison
Lehigh University, USA
Ludovic Denoyer and Patrick Gallinari
LIP6 - University Pierre et Marie Curie, France

Preface

Topics

The workshop focus is on the **Graph labelling problem**. The goal of the graph labelling task is to automatically label the nodes of a graph (with or without content information on the nodes of the graph). The generic task has a lot of applications in many different domains: Web spam detection, Social networks,.....

The scope of this workshop is the development of new models for graph labelling and all the applications where the data can be represented as a graph :

- Generic graph labelling models
- Tree annotation models
- Models for large graph
- Web spam detection
- XML annotation
- Wiki, Blog, Web retrieval
- Web classification and clustering
- Social networks

The workshop particularly focuses on a key application which is **Web Spam detection** where the goal is to label the nodes (the Web pages or Web hosts) of a graph as spam or not spam. This workshop presents the results obtained by different research teams on the second phase of the PASCAL WebSpam Challenge[1].

The workshop has been opened to any submission concerning theoretical models or large size applications of graph labelling with a particular focus on internet graphs.

Graph Labelling

Many domains and applications are concerned with complex data composed of elementary components linked according to some structural or logical organization. These data are often described as a graph where nodes and links hold information. In Computer Vision for example, a picture can be described by a graph which corresponds to the organization of the different regions of the pictures – each node of the graph corresponding to a region. In the text domain, the diffusion of new data formats like XML and HTML has considerably changed the domains of Information Retrieval. On the Web, documents are organized according to a graph structure where nodes correspond to the Web page and edges correspond to the hyper links between the pages. Moreover, Web pages are also structured documents containing both a content information and a logical information encoded by the HTML tags, and can be viewed as labelled trees. Other application domains concerned with graph data include image processing, multimedia (video), natural language processing, social networks, biology, etc. Handling structured

[1] http://webspam.lip6.fr

data has become a main challenge for these domains and different communities have been developing for some years their own methods for dealing with structured data. The ML community should be a major actor in this area. Graph labelling which consists in labelling all the vertices of a graph from a partial labelling of the graph vertices has been identified as a generic ML problem with many fields of application.

Web Spam detection

Web spam detection is becoming a major target application for web search providers. The Web contains numerous profit-seeking ventures that are attracted by the prospect of reaching millions of users at a very low cost. There is an economic incentive for manipulating search engine's listings by creating pages that score high independently of their real merit. In practice such manipulation is widespread, and in many cases, successful.

Traditional IR methods assumed a controlled collection in which the authors of the documents being indexed and retrieved had no knowledge of the IR system and no intention of manipulating its behaviour. On Web-IR, these assumptions are no longer valid, specially when searching at global scale.

Almost every IR algorithm is prone to manipulation in its pure form. A ranking based purely on the vector space model, for instance, can be easily manipulated by inserting many keywords in the document; a ranking based purely on counting citations can be manipulated by creating many meaningless pages pointing to a target page, and so on.

Of course, ideally the search engine administrators want to stay ahead of the spammers in terms of ranking algorithms and detection methods. Fortunately, from the point of view of the search engine, the goal is just to alter the economic balance for the would-be spammer, not necessarily detecting 100% of the Web spam. If the search engine can maintain the costs for the spammers consistently above their expected gain from manipulating the ranking, it can really keep Web spam low.

The adversarial Information Retrieval on the Web (AIRWeb) series of workshops was started in 2005 by the academic community. Many existing heuristics for detection are often specific to a specific type of spam and can not be used if a new Web spam technique appears. We need to propose new models able to learn to detect any type of Web Spam and that can be adapted quickly to new unknown spam techniques. **Machine learning methods are the key to achieve this goal.**

Carlos Castillo
Brian D. Davison
Ludovic Denoyer
Patrick Gallinari

Workshop Organization

Workshop Chairs

Carlos Castillo (Yahoo! Research)
Brian D. Davison (Lehigh University)
Ludovic Denoyer (LIP6 - University Pierre et Marie Curie)
Patrick Gallinari (LIP6 - University Pierre et Marie Curie)

ECML/PKDD Workshop Chair

Marzena Kryszkiewicz (Warsaw University of Technology)

Workshop Program Committee

Kumar Chellapilla	Mark Herbster
Brian D. Davison	Massimiliano Pontil
Ludovic Denoyer	Juho Rousu
Dennis Fetterly	John Shawe Taylor
Patrick Gallinari	Alessandro Sperduti
Remi Gilleron	Tanguy Urvoy
Marco Gori	

Table of Contents

Semi-Supervised Learning: A Comparative Study for Web Spam and Telephone User Churn 1
András A. Benczúr and Károly Csalogány and László Lukács and Dávid Siklósi

A Fast Method to Predict the Labeling of a Tree 9
Sergio Rojas Galeano and Mark Herbster

A Semi-Supervised Approach for Web Spam Detection using Combinatorial Feature-Fusion 16
Ye Tian and Gary M. Weiss and Qiang Ma

Web Spam Challenge 2007 Track II - Secure Computing Corporation Research . . 24
Yuchun Tang and Yuanchen He and Sven Krasser and Paul Judge

Semi-supervised classification with hyperlinks 32
Jacob Abernethy and Olivier Chapelle

Webspam detection via Semi-Supervised Graph Partitioning 33
Chris Biemann and Hans Friedrich Witschel

SpamChallenge 2007 - Track II: France Telecom RD Submissions 35
Pascal Filoche and Tanguy Urvoy and Marc Boullé

Author Index 37

Semi-Supervised Learning: A Comparative Study for Web Spam and Telephone User Churn[⋆]

András A. Benczúr Károly Csalogány László Lukács Dávid Siklósi

Informatics Laboratory
Computer and Automation Research Institute
Hungarian Academy of Sciences
11 Lagymanyosi u, H-1111 Budapest
and
Eötvös University, Budapest
{benczur, cskaresz, lacko, sdavid}@ilab.sztaki.hu
http://datamining.sztaki.hu/

Abstract. We compare a wide range of semi-supervised learning techniques both for Web spam filtering and for telephone user churn classification. Semi-supervised learning has the assumption that the label of a node in a graph is similar to those of its neighbors. In this paper we measure this phenomenon both for Web spam and telco churn. We conclude that spam is often linked to spam while honest pages are linked to honest ones; similarly churn occurs in bursts in groups of a social network.

1 Introduction

Semi-supervised learning, a new field of machine learning surveyed e.g. in [27] also exploits information from unlabeled data for learning. We focus on the applicability of classifying Web spam and telephone churn, i.e. users who cancel their telephone line. Our assumption is that the label (spam and churned, respectively) of a node in a graph is similar to those of its neighbors.

We compare various means of stacked graphical learning, a meta-learning scheme in which a base learner is augmented by expanding the features of one node with predictions on other related nodes in a graph is introduced recently by Kou and Cohen [15]. The methodology is used with success for Web spam detection in [5]: they use the average label of the neighbors as a new feature for the classifier.

We run our tests on the Web Spam Challenge datasets. The baseline decision tree utilized all graph based features related to a node (i.e. features related to the "home page" or the "maximum PageRank node within site" are not computed) [5] and a Naive Bayes classifier of the machine learning toolkit Weka [24] over the content based features of the Web Spam Challenge Phases I and II data. Depending on the data set the best forms of graph stacking improve the F-measure by 1-10% as shown in Section 3.2.

The other data set we use for evaluating and comparing graph labeling methods is a telephone call graph, a data type that appears less in the publications of the data

[⋆] Support from a Yahoo! Faculty Research Grant, projects NKFP-2/0024/2005 and NKFP-2004 Language Miner http://nyelvbanyasz.sztaki.hu.

mining community. Closely related to our work are the churn prediction results by machine learning methods on real data [23, 1, etc.]; these results however do not exploit neighborhood information embedded in the call graph.

The telephone call graph is formed from the call detail record, a log of all calls within a time period including caller and callee id, duration, cost and time stamp. The vertex set consists of all nodes that appear at least once as caller or calle; over this set calls form directed edges from caller to callee.

Churn classification uses customer information (price package, time since in service etc.) as well as traffic aggregates in various call zones and directions. We use one year call detail record and all customer information up to a given time; the classification target consists of users who leave service in the fourth month "in future" (in a time period with no information available for the classifier). Due to the sparsity of positive instances (below 1% churn in a month) and a large amount of churn explained by external reasons such as the customer moves churn classification is a hard task; baseline reaches $F = 0.08$ and this is improved to 0.1 by stacked graphical learning. In the industrial practice the goodness of the churn classification is measured by the recall of the top list of 10% of the customers, i.e. they are willing to involve a maximum of 10% of their customers in direct marketing campaigns and want to maximize the potential churn reached. In this sense our baseline classification has a recall of 40.8%, improved to 47% by stacked graphical learning.

In this paper we concentrate on spreading trust (or no churn) and distrust (churn) information from known nodes with the help of hyperlink based similarity measures. Our main goal is to identify features based on similarities to known honest and spam pages that can be used to classify unknown pages. We propose a set of spam and churn classification methods that combine graph based similarity to labeled nodes [2] with trust and distrust propagation both backward and forward. For example given a link farm alliance [10] with one known target labeled as spam, similarity based features will automatically label other targets as spam as well.

Our stacked graphical learning algorithms generate features by averaging known and predicted labels for similar nodes of the graph by the measures in Section 2.1. We compare various similarity measures, including simple and multi-step neighborhood, co-citation, cosine and Jaccard similarity of the neighborhood as well as their multi-step variants [8] described in detail in Section 2. For the purposes of evaluation we consider these algorithms separately, by performing one classification experiment for each feature.

1.1 Related results

Identifying and preventing spam is cited as one of the top challenges in web search engines in [13]. major search engines incorporate anchor text and link analysis algorithms into their ranking schemes, Web spam appears in sophisticated forms that manipulate content as well as linkage [11]. Spam hunters use a variety of both content [7, 19] and link [12, 6, 25, 3, 2] based features to detect Web spam; a recent measurement of their combination appears in [5].

Recently several results has appeared that apply rank propagation to extend initial trust or distrust judgments over a small set of seed pages or sites to the entire web, such as trust [12, 26], distrust [20, 6] propagation in the neighborhood or their combination [25] as well as graph based similarity measures [2]. These methods are either based on propagating trust forward or distrust backwards along the hyperlinks based on the idea that honest pages predominantly point to honest ones, or, stated the other way, spam pages are backlinked only by spam pages. Trust and distrust propagation in trust networks originates in Guha et al. [9] for trust networks; Wu et al. [25] is the first to show its applicability for Web spam classification.

Trust and distrust propagation are in fact forms of semi-supervised learning surveyed by Zhu [27], a methodology to exploit unlabeled instances in supervised classification. Stacked graphical learning introduced by Kou and Cohen [15] is a simple implementation that outperforms the computationally expensive variants [15, 5].

Identifying spam pages is somewhat analogous to classifying web documents into multiple topics. Several results [21, and the references therein] demonstrate that classification accuracy can be significantly increased by taking into account the class labels assigned to neighboring nodes. In accordance with [2], Qi and Davison [21] found that most of the improvement comes from the neighborhood defined by co-citation.

Several link-based algorithms were designed to evaluate node-to-node similarities in networks that can be used to give alternate, similarity based weights to node pairs. We refer to [16] for an exhaustive list of the available methods ranging from co-citation to more complex measures such as max-flow/min-cut-based similarities of [17] in the vicinity graph of the query. Co-citation is in fact used in [9] as an elementary step of trust propagation. Another method [18] penalizes the biconnected component of a spam page in a subgraph obtained by backward distrust propagation.

2 The stacked graphical learning framework

2.1 Feature generation

For a given unknown node u and edge weight function w (that may be in or out-degree, cocitation, PageRank etc.), our algorithm selects the k largest weight neighbors of u to generate a new feature based on the known spam and honest hosts in this set. As in [2] we extract four different features from this set of size k or possibly less if u has less than k neighbors. Each element v is either classified as spam with weight $p(v)$ or else labeled spam or nonspam; in these cases we let $p(v)$ be 0 and 1, respectively. Let s and h be the sum of $p(v)$ and $1 - p(v)$ in the set; remember $s + h < k$ is possible. We define a weighted version s^* and h^* as the sum of $w(uv) \cdot p(v)$ and $w(uv) \cdot (1 - p(v))$.

We define our features as follows.

- Spam Ratio (SR): fraction of the number of spam within labeled spam and honest pages, $s/(s + h)$.
- Spam over Non-spam (SON): number of spam divided by number of honest pages in the top list, s/h.
- Spam Value Ratio (SVR): sum of the similarity values of spam pages divided by the total similarity value of labeled spam and honest pages under the appropriate similarity function, $s^*/(s^* + h^*)$.

– Spam Value over Non-spam Value (SVONV): similarity value sum for spam divided by same for honest, s^*/h^*.

In most of the experiments we use SVR that also performed best in [2]; a small comparison is made in Section 3.2.

We add the new feature defined by either of the above to the existing ones and repeat the classification process with the extended feature set. Since the features are unstable if the neighborhood $N(u)$ is small, we also define versions SR', SON', SVR', SVONV' by regressing towards the undecided 1/2 or 1 value:

$$\text{SR}' = 1/2+(\text{SR}-1/2)\cdot(1-1/\sqrt{|N(u)|}); \quad \text{SON}' = 1+(\text{SON}-1)\cdot(1-1/\sqrt{|N(u)|}).$$

2.2 Direction of propagation

We may use both the input directed graph, its transpose by changing the direction of each edge, or the undirected version arising as the union of the previous two graphs. We will refer to the three variants as *directed*, *reversed* and *undirected* versions. For an edge weight function $d : V \times V \to \mathbf{R}$ we use $d^-(u,v) = d(v,u)$ for the reversed and $d^\pm = d + d^-$ for the undirected version. We extend this notion for an arbitrary similarity measure sim(u,v) computed over edge weights d and compute sim$^-(u,v)$ over d^- and sim$^\pm(u,v)$ over $d^\pm$.

Performance of directed, reversed or undirected varies problem by problem: the templatic nature of a Web spam farm is best characterized by similarity of out-links (directed), honest pages have incoming links from honest ones (reversed) and finally similarity in a telephone call graph is best characterized by the undirected graph since communication is typically bidirectional regardless of the actual caller–callee direction.

2.3 Multi-step propagation

There are several variants of weighting neighbors at distance k. We may consider reachability and exact reachability as $d^k(u,v)_{\text{reach}} = 1$ if v is reachable from u by a walk over k edges, 0 otherwise, respectively $d^k_{\text{exact}}(u,v) = 1$ if v is reachable from u in exactly k steps and over no shorter paths, 0 otherwise. We may take the number and the weighted number of such walks: $d^k_{\text{num}}(u,v)$ is the number of walks over k edges that reach from u to v and $d^k_{\text{wnum}}(u,v)$ is the probability of reaching v when starting at u and at each step choosing a random neighbor with probability proportional to the outgoing edge weights. The main multi-step feature we use is PPR(u), PageRank personalized to $p(v)$, the estimated spamicity of node v as in Section 2.1:

$$\text{PPR}(u) = \sum_k c(1-c)^k \sum_v p(v) \cdot d^k_{\text{wnum}}(u,v).$$

2.4 Cocitation, Jaccard and cosine

The cocitation coc(u,v) is defined as the number of common in-neighbors of u and v. This measure turned out most effective for Web spam classification [2]. By the notation of Section 2.2 coc$^-(u,v)$ denotes bibliographic coupling (nodes pointed to by

F-measure	graph stacking																	
$\times$1000	none	d	coc		coc$^-$		coc$^\pm$		Jac		Jac$^-$		Jac$^\pm$		cosine		PPR	
iterations		1	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Web Spam I	689	695	707	709	669	677	722	**724**	715	703	689	690	679	680	698	699	715	719
Web Spam II small, text	592	589	601	**605**	598	599	599	601	590	590	592	594	593	595	600	601	599	600
Web Spam II small, link	762	752	788	**793**	774	765	748	738	756	762	782	777	766	756	760	760	731	737
Web Spam II large, link	939	962	983	984	987	988	983	984	984	985	975	976	961	953	982	982	958	960
Churn	086	**102**	063	052	079	088	**102**	083	067	065	059	066	097	089	084	065	092	087
Churn, nonchurn sampled	161	155	141	142	197	200	114	121	254	265	153	147	175	158	267	**280**	277	257

Table 1. 1000 times the F-measure shown for different data sets and edge weights.

both u and v) and coc$^\pm(u, v)$ is the undirected cocitation. We may also use cocitation downweighted by degree, coc$(u, v)/d(u) \cdot d(v)$.

The Jaccard and cosine similarity coefficients are useful for finding important connections and ignoring "accidental" unimportant connections. The Jaccard coefficient Jac(u, v) is the ratio of common neighbors within all neighbors. The coefficient has variants that use the reversed or undirected graphs. For a weighted graph we may divide the total weight to common neighbors by the total weight of edges from u and v. This measure performs poor if for example edges ux and vy have low weight while uy and vx have very high since the Jaccard coefficient is high while the actual similarity is very low.

Cosine similarity fixes the above problem of the Jaccard coefficient. We consider the row of the adjacency matrix corresponding to node u as vector $\overline{u}$. The cosine similarity of nodes u and v is simply $\cos(u, v) = \overline{u}^T \overline{v}$. We may similarly define $\cos^-(u, v)$ over the transpose matrix and $\cos^\pm(u, v)$ over the sum.

Since filling a quadratic size matrix is infeasible, we calculate Jaccard and cosine only for existing edges. The resulting scheme downweights unimportant edges but is unable to add "uncaught contacts" to the network. It is possible to find all pairs with weight above a given threshold by fingerprinting techniques; we leave performance tests for future work.

3 Experiments

3.1 Data sets

For Web spam classification we follow the same methodology as Castillo et al. [5]. We use the Web Spam Challenge Phase I dataset WEBSPAM-UK-2006 [4] that consists of 71% of the hosts classified as normal, 25% as spam and the remainder 4% as undecided as well as the Phase II data set WEBSPAM-LIP6-2006. In this preliminary experiment we consider three tasks. First we use Phase I data (the *Domain Or Two Humans* classification that introduces additional nonspam domains and gives 10% spam among the 5622 labeled sites) with the publicly available features of [5] and then classify by the

	d	coc	coc$^{\pm}$	Jac	Jac$^{\pm}$	cosine	PPR
SON'	602	615	602	600	599	599	599
SON	600	614	602	599	599	599	598
SR'	**603**	618	609	**611**	**606**	**610**	**608**
SR	601	**619**	**611**	610	605	**610**	603
SVONV'	602	607	602	598	596	597	599
SVONV	600	606	602	598	596	598	599
SVR'	**603**	618	609	603	**606**	604	600
SVR	601	**619**	**611**	600	604	601	600

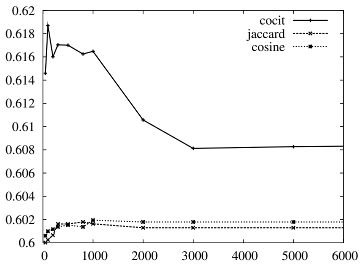


Table 2. Left: 1000 times the F-measure shown for different data weights and feature generation methods. **Right:** the effect of the top list size for SVR.

cost sensitive C4.5 implementation of the machine learning toolkit Weka [24] with bagging. Then we use the Phase II data set features and use the Naive Bayes classifier of Weka. Finally we compute all graph based features of [5] for the Phase II data graph and classify by C4.5 again. We combined the text and graph classifiers by SVM.

For churn classification we use data from a small Hungarian landline telephone service provider. We form features based on aggregated call cost duration in different cost segments, including daytime and off-peak, weekday and weekend as well as local and different long-distance call volumes. Part of the users perform calls via an alternate provider by dialing a prefix; these calls are aggregated similarly for each user. We also use the pricing package information that also includes a distinction of company and family lines as well as the start date of the service usage. For a time range of 12 months, after aggregating calls between the same pairs of callers we obtained a graph with $n = 66,000$ nodes and $m = 1,360,000$ directed edges.

We use the cost sensitive C4.5 implementation of the machine learning toolkit Weka [24] with bagging. Since the running times on the full data set were over 10 hours we also compiled a smaller data set where a random sample of non-churned users were dropped, resulting in $7,151$ users but we kept the entire graph.

3.2 Classification results

Table 1 shows the first three digits of the F-measure for the best selected settings, with the best result in bold. For the Web spam data we measure over the testing labels while for churn we use 10-fold crossvalidation. Since the text and link SVM-combined Web Spam II experiment is computationally very expensive, we only computed the base and the simple neighbor methods that give 0.738 and improve to 0.748 for the small and 0.338 vs. 0.449 for the large graph.

In Table 2 we can see that the difference between the feature generation methods of Section 2.1 are minor and the length of the top list has little effect in the range of k between 100 and 1000, although for cocitation the very long and for others the very short lists deteriorate the performance. Results are shown for the text features of the small Phase II graph and single-iteration stacked graphical classification.

4 Conclusion and Future Work

We presented Web spam and landline telephone churn classification measurements over the Web Spam Challenge Phase II and a small Hungarian landline telephone provider year 2005 datasets. Our experiments demonstrated that stacked graphical learning in combination with graph node similarity methods improve classification accuracy in both cases. Due to the large number of possible feature generation methods the results are by no means complete but show a very good performance of co-citation and little actual use of the neighborhood beyond two steps in the graph.

For future work we plan testing more complex multi-step variants of cocitation and the Jaccard coefficient. Jeh and Widom [14] define SimRank as a multi-step generalization of downweighted cocitation. In an alternate formulation [22] the k-step SimRank $\mathrm{Sim}^{(k)}_{v_1,v_2}$ equals the total weight of pairs of walks with length $k' \leq k$ that both end at u and one of them comes from v_1 while the other one from v_2. The weight of the pair of walks is the *expected* $(1-c)$ *meeting distance* as defined in [14]; notice we get downweighted cocitation if $k = 1$. Computing the full SimRank matrix requires quadratic space; we may use the algorithm of [22] instead. Finally Fogaras and Rácz [8] describe XJaccard as the weighted sum of Jaccard coefficients of the distance k neighborhoods and give an efficient randomized approximation algorithm to compute it.

References

1. W.-H. Au, K. C. C. Chan, and X. Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Trans. Evolutionary Computation*, 7(6):532–545, 2003.
2. A. A. Benczúr, K. Csalogány, and T. Sarlós. Link-based similarity search to fight web spam. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), held in conjunction with SIGIR2006*, 2006.
3. A. A. Benczúr, K. Csalogány, T. Sarlós, and M. Uher. SpamRank – Fully automatic link spam detection. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), held in conjunction with WWW2005*, 2005. To appear in *Information Retrieval*.
4. C. Castillo, D. Donato, L. Becchetti, P. Boldi, S. Leonardi, M. Santini, and S. Vigna. A reference collection for web spam. *SIGIR Forum*, 40(2):11–24, December 2006.
5. C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: Web spam detection using the web topology. Technical report, DELIS – Dynamically Evolving, Large-Scale Information Systems, 2006.
6. I. Drost and T. Scheffer. Thwarting the nigritude ultramarine: Learning to identify link spam. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, volume 3720 of *Lecture Notes in Artificial Intelligence*, pages 233–243, Porto, Portugal, 2005.
7. D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics – Using statistical analysis to locate spam web pages. In *Proceedings of the 7th International Workshop on the Web and Databases (WebDB)*, pages 1–6, Paris, France, 2004.
8. D. Fogaras and B. Rácz. Scaling link-based similarity search. In *Proceedings of the 14th World Wide Web Conference (WWW)*, pages 641–650, Chiba, Japan, 2005.
9. R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 403–412, 2004.

10. Z. Gyöngyi and H. Garcia-Molina. Link spam alliances. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, Trondheim, Norway, 2005.
11. Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, Chiba, Japan, 2005.
12. Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 576–587, Toronto, Canada, 2004.
13. M. R. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2):11–22, 2002.
14. G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 538–543, 2002.
15. Z. Kou and W. W. Cohen. Stacked graphical models for efficient inference in markov random fields. In *SDM 07*, 2007.
16. D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the 12th Conference on Information and Knowledge Management (CIKM)*, pages 556–559, 2003.
17. W. Lu, J. Janssen, E. Milios, and N. Japkowicz. Node similarity in networked information spaces. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative research*, page 11, 2001.
18. P. T. Metaxas and J. Destefano. Web spam, propaganda and trust. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web*, 2005.
19. A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, pages 83–92, Edinburgh, Scotland, 2006.
20. PR10.info. BadRank as the opposite of PageRank, 2004. `http://en.pr10.info/pagerank0-badrank/` (visited June 27th, 2005).
21. X. Qi and B. D. Davison. Knowing a web page by the company it keeps. In *Proceedings of the 15th Conference on Information and Knowledge Management (CIKM)*, 2006.
22. T. Sarlós, A. A. Benczúr, K. Csalogány, D. Fogaras, and B. Rácz. To randomize or not to randomize: Space optimal summaries for hyperlink analysis. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, pages 297–306, 2006. Full version available at http://www.ilab.sztaki.hu/websearch/Publications/.
23. C.-P. Wei and I.-T. Chiu. Turning telecommunications call details to churn prediction: a data mining approach. *Expert Syst. Appl.*, 23(2):103–112, 2002.
24. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.
25. B. Wu, V. Goel, and B. D. Davison. Propagating trust and distrust to demote web spam. In *Workshop on Models of Trust for the Web*, Edinburgh, Scotland, 2006.
26. B. Wu, V. Goel, and B. D. Davison. Topical TrustRank: Using topicality to combat web spam. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, Edinburgh, Scotland, 2006.
27. X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

A Fast Method to Predict the Labeling of a Tree

Sergio Rojas Galeano and Mark Herbster

[1] Department of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
{M.Herbster
[2] S.Rojas}@cs.ucl.ac.uk

Abstract. Given an n vertex weighted tree with (structural) diameter $S_{\mathbf{G}}$ and a set of ℓ vertices we give a method to compute the corresponding $\ell \times \ell$ Gram matrix of the pseudoinverse of the graph Laplacian in $O(n + \ell^2 S_{\mathbf{G}})$ time. We discuss the application of this method to predicting the labeling of a graph. Preliminary experimental results on a digit classification task are given.

1 Introduction

Classification methods which rely upon the graph Laplacian (see [2, 10, 5] and references therein), have proven to be useful for transductive and semi-supervised learning. A key insight of these methods is that unlabeled data can be used to improve the performance of supervised learners. These methods reduce to the problem of labeling a graph whose vertices are associated to the data points and the edges to the similarity between pairs of data points. The labeling of the graph can be achieved either in a batch [2, 10] or in a online manner [5]. These methods can all be interpreted as different kernel methods: ridge regression in the case of [2], minimal semi-norm interpolation in [10] or the perceptron algorithm in [5]. Therefore, in all cases an important preprocessing step is the computation of the kernel[3], which is the pseudoinverse of the graph Laplacian or a matrix related to it [5]. This computation usually scales cubically with the quantity of unlabeled data, which may prevent the use of these methods on large graphs. In the case of unbalanced bipartite graphs [7] presents a method which significantly improves the computation time of the pseudoinverse. Finally [3] presents a non-Laplacian-based method for predicting the labeling of tree based on computing the exact probabilities of a Markov random field.

In this paper, we are concerned with the practical scenario in which only a relatively small number of vertices of a large graph need to be labeled. Specifically, we divide the vertices into three sets, $V^{(\ell)}, V^{(p)}$, and $V^{(u)}$, which are the set of labeled vertices, the set of unlabeled vertices for which we will give predictions, and the set of remaining unlabeled vertices (no predictions is required on them), respectively. Therefore, if $V^{(n)}$ denotes the set of n vertices of the graph,

[3] Computation in the primal is also possible with a comparable time expenditure.

we have that $n = \ell + u + p$, where ℓ is number of labeled vertices, p is the number of *predictive* unlabeled vertices and u is the number of nonpredictive unlabeled vertices. Typically ℓ is much smaller than both p and u, and p is much smaller than u.

In this paper, we propose a technique to improve the computational complexity of Laplacian-based learning methods. The method is based on approximating the original graph with a tree. Computationally our method requires an $O(n)$ initialization step and after that any element of the pseudoinverse of the Laplacian of a tree may be computed in $O(S_{\mathbf{G}})$ time, where $S_{\mathbf{G}}$ is the structural diameter of the tree $\mathbf{G}$. The pseudoinverse of the Laplacian may then be used as a kernel [6, 5] for a variety label prediction methods. We consider as specific examples the kernel perceptron and ridge regression (see e.g., [9]). Thus if we are given an n vertex tree and assume that the time to compute the inverse of a matrix is cubic in the dimension then with ℓ labeled vertices and p predictive vertices the total time required to predict with the kernel perceptron is $O(n + \ell^2 S_{\mathbf{G}} + p\ell S_{\mathbf{G}})$ and with kernel ridge regression is $O(n + \ell^2 S_{\mathbf{G}} + \ell^3 + p\ell S_{\mathbf{G}})$. The promise of our technique is that if $(l + p + S_{\mathbf{G}}) \ll n$ and a tree is given, our method requires $O(n)$ time versus $O(n^3)$ for standard methods and when only a similarity function on the data is given then our method requires $O(n^2)$ time and $O(n)$ space.

2 Preliminairies

In this paper any graph $\mathcal{G}$ is assumed connected, to have n vertices, and to have edge weights. The set of vertices of $\mathcal{G}$ is denoted $V = \{1, \ldots, n\}$. Let $\mathbf{A}$ denote the $n \times n$ symmetric nonnegative weight matrix of the graph such that $A_{ij} \geq 0$, and define the edge set $E(\mathcal{G}) := \{(i, j) : 0 < A_{ij},\, i < j\}$. We say that $\mathcal{G}$ is a *tree* if it is connected and has $n - 1$ edges. The graph Laplacian $\mathbf{G}$ is the $n \times n$ matrix defined as

$$\mathbf{G} := \mathbf{D} - \mathbf{A},$$

where $\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_n)$ and d_i is the *weighted* degree of vertex i, $d_i = \sum_{j=1}^{n} A_{ij}$. The Laplacian is positive semidefinite and induces the semi-norm

$$\|\mathbf{w}\|_{\mathbf{G}}^2 := \mathbf{w}^\top \mathbf{G} \mathbf{w} = \sum_{(i,j) \in E(\mathbf{G})} A_{ij} (w_i - w_j)^2. \tag{1}$$

The reproducing kernel [1] associated with the above semi-norm is $\mathbf{G}^+$, where "+" denotes pseudoinverse (see [6, 5] for further details). As the graph is connected, it follows from equation (1) that the null space of $\mathbf{G}$ is spanned by the constant vector $\mathbf{1}$ only. The weighted graph may be seen as a network of resistors where edge (i, j) is a resistor with resistance $\pi_{ij} = A_{ij}^{-1}$. Then the *effective resistance* $r_{\mathbf{G}}(i, j)$ may be defined as the resistance measured between vertex i and j in this network and may be calculated using Kirchoff's circuit laws or directly from $\mathbf{G}^+$ using [8]

$$r_{\mathbf{G}}(i, j) = \mathbf{G}^+_{ii} + \mathbf{G}^+_{jj} - 2\mathbf{G}^+_{ij}\,. \tag{2}$$

The effective resistance is a metric distance on the graph [8] as well as the geodesic $d_{\mathbf{G}}$ and structural $s_{\mathbf{G}}$ distances. The structural (geodesic) distance between vertices $i, j \in V$ is

$$\begin{aligned} s_{\mathbf{G}}(i,j) &:= \min\{|P(i,j)| : P(i,j) \in \mathcal{P}\}\,, \\ d_{\mathbf{G}}(i,j) &:= \min\{ \sum_{(p,q)\in\mathcal{P}(i,j)} \pi_{pq} : P(i,j) \in \mathcal{P}\}\,, \end{aligned}$$

where $\mathcal{P}$ is the set of all paths in $\mathcal{G}$ and $P(i,j)$ is the set of edges in a particular path from i to j. The diameter is the maximum distance between any two points on the graph hence the resistance, structural, and, geodesic diameter are

$$R_{\mathbf{G}} = \max_{i,j\in V} r_{\mathbf{G}}(i,j)\,,\quad S_{\mathbf{G}} = \max_{i,j\in V} s_{\mathbf{G}}(i,j)\,,\ \text{and}\ \ D_{\mathbf{G}} = \max_{i,j\in V} d_{\mathbf{G}}(i,j)\,,$$

respectively.

3 Computing the Pseudoinverse of a Tree Quickly

In the following we give our method to compute the pseudoinverse of a tree. The principle of the method is that when a graph is a tree, there is a unique path between any two vertices hence the *effective resistance* is simply the sum of resistances along that path (see for example [8, 5]) and hence on trees the geodesic distance is equivalent to the resistance distance. We now additionally assume that $\mathcal{G}$ is a tree, the root vertex of the tree is indexed as 1. The parent of vertex i is denoted $\uparrow(i)$ while the children of i are $\downarrow(i)$ and the descendants of i are

$$\downarrow^*(i) := \begin{cases} \downarrow(i) \cup \sum_{j\in\downarrow(i)} \downarrow^*(j) & \text{if } \downarrow(i) \neq \emptyset \\ \emptyset & \text{if } \downarrow(i) = \emptyset \end{cases}.$$

We define $Z := \sum_{i=1}^{n} \mathbf{G}^+_{ii}$, $R(i) := \sum_{j\neq i} r_{\mathbf{G}}(i,j)$ and $R := \sum_{i=1}^{n} R(i)$. In equations (3) and (5) we give the formulas which we use to compute $\mathbf{G}^+$.The off-diagonal elements are computed with

$$\mathbf{G}^+_{ij} = \frac{\mathbf{G}^+_{ii} + \mathbf{G}^+_{jj} - r_{\mathbf{G}}(i,j)}{2}. \tag{3}$$

as follows from (2). Observe that

$$\mathbf{G}^+_{ii} = -\sum_{j\neq i} \mathbf{G}^+_{ij} \tag{4}$$

since the null space of $\mathbf{G}$ is spanned by the constant vector $\mathbf{1}$. Thus we may substitute (3) into (4) to obtain

$$\mathbf{G}^+_{ii} = -\frac{1}{2}\left[(n-1)\mathbf{G}^+_{ii} + \sum_{j\neq i}\mathbf{G}^+_{jj} - \sum_{j\neq i} r_{\mathbf{G}}(i,j)\right],$$

thus

$$\mathbf{G}^+_{ii} = \frac{R(i) - Z}{n} \text{ and } Z = \frac{R}{2n}. \tag{5}$$

We now describe a method to initially compute $\mathbf{G}^+_{ii}, R(i)$, $i = 1, \ldots, n, R$ and Z in $O(n)$ time and then with these precomputed values we may compute entries as needed in $\mathbf{G}^+_{ij}$ from equation (3) by computing $r_{\mathbf{G}}(i,j)$ in $O(S_{\mathbf{G}})$ time. The number of descendents of i (including i) is $\kappa(i) = 1 + |\downarrow^*(i)|$, while $T(i)$ (resp. $S(i)$) is the sum of the resistances of vertex i to each descendant (resp. non-descendent), hence,

$$T(i) = \sum_{j \in \downarrow^*(i)} r_{\mathbf{G}}(i,j)\,, \quad S(i) = \sum_{j \notin \downarrow^*(i)} r_{\mathbf{G}}(i,j)\,.$$

We compute $\kappa(i)$ and $T(i)$ (for $i = 1, \ldots, n$) with leaves-to-root recursions while we compute $S(i)$ with a root-to-leaves recursion. These $3n$ quantities are computed with the following recursions,

$$\kappa(i) := \begin{cases} 1 + \sum_{j \in \downarrow(i)} \kappa(j) & \downarrow(i) \neq \emptyset \\ 1 & \downarrow(i) = \emptyset \end{cases},$$

and

$$T(i) := \begin{cases} \sum_{j \in \downarrow(i)} (T(j) + \pi_{ij}\kappa(j)) & \downarrow(i) \neq \emptyset \\ 0 & \downarrow(i) = \emptyset \end{cases}$$

by computing $\kappa(1)$ then $T(1)$ and caching the intermediate values. We observe that $R(1) = T(1)$. We now descend the tree caching each calculated

$$S(i) := \begin{cases} S(\uparrow(i)) + T(\uparrow(i)) - T(i) + (n - 2\kappa(i))\pi_{i\uparrow(i)} & i \neq 1 \\ 0 & i = 1 \end{cases}.$$

Now as $R(i) = S(i) + T(i)$, the diagonal of $\mathbf{G}^+$ is calculated from (5) with a cumulative computation time of $O(n)$. We now observe to compute $\mathbf{G}^+_{ij}$ (see Equation (3)) we need $r_{\mathbf{G}}(i,j)$ which is simply the sum of resistances along the path from i to j, this path may be computing by separately computing the path from i–to–1 and j–to–1 ($O(S_{\mathbf{G}})$ time) and summing the resistances along each edge that is either in i–to–1 or j–to–1 but not both. In the full paper we will show that if we need to calculate an $\ell \times \ell$ submatrix of $\mathbf{G}^+$ this may be accomplished in $O(n + \ell^2 + \ell S_{\mathbf{G}})$ time.

4 Tree Construction

In the previous discussion, we have considered that a tree has already been given. In the following, we assume that a graph $\mathcal{G}$ or a similarity function is given and the aim is to construct an approximating tree. We will consider both the *minimum spanning tree* (`MST`) as a “best” in norm approximation; and the *shortest* *path tree* (SPT) as an approximation which maintains a mistake bound [6, 5] guarantee. Moreover, we comment on the time and space complexity of constructing such trees. Given a graph with a "cost" on each edge the MST is a subgraph with $n-1$ edges such that the total cost is minimized. A SPT(i) at vertex i is a subgraph such that geodesic distance in "costs" is minimized from i to every other vertex. In our set-up the cost of edge (i,j) is π_{ij} therefore,

$$\texttt{MST}(\mathbf{G}) = \arg\min_{\mathbf{T}\subseteq\mathbf{G}}\{\sum_{(i,j)\in E(\mathbf{T})} \pi_{ij} : |\mathbf{T}| = n-1, \mathbf{T} \text{ is a tree}\},$$

$$\texttt{SPT}(\mathbf{G}, i) = \arg\min_{\mathbf{T}\subseteq\mathbf{G}}\{\sum_{j=1}^{n} r_{\mathbf{T}}(i,j) : |\mathbf{T}| = n-1, \mathbf{T} \text{ is a tree}\}.$$

Observe that for a graph with unit costs every tree is a MST but not necessarily a SPT. A MST is also the tree whose Laplacian best approximates the Laplacian of the given graph according to the trace norm, that is,

$$\texttt{MST}(\mathbf{G}) = \arg\min_{\mathbf{T}\subseteq\mathbf{G}}\{\|\mathbf{G}-\mathbf{T}\|_{\mathbf{tr}} : |\mathbf{T}| = n-1, \mathbf{T} \text{ is a tree}\}.$$

We now provide a justification for approximating the given graph by a SPT. It relies upon the analysis in [5, Theorem 4.2], where the cumulative number of mistakes of the kernel perceptron with the kernel $\mathbf{K} = \mathbf{G}^{+} + \mathbf{1}\mathbf{1}^{\top}$ was upper bounded by

$$|\mathcal{M}_A| \leq (\|\mathbf{u}\|_{\mathbf{G}}^2 + 1)(R_{\mathbf{G}} + 1), \tag{6}$$

for consistent labelings $\mathbf{u} \in \{-1,1\}^n$. To explain our argument, first we note that when we approximate the graph with a tree $\mathbf{T}$ the term $\|\mathbf{u}\|_{\mathbf{G}}^2$ is always decreasing, while the term $R_{\mathbf{G}}$ is always increasing by Rayleigh's monotonicity law (see for example [5, Corollary 3.1]). The resistance diameter $R_{\mathbf{T}}$ of an SPT subgraph is bounded by twice the geodesic diameter of the original graph,

$$R_{\mathbf{T}} \leq 2D_{\mathbf{G}}, \tag{7}$$

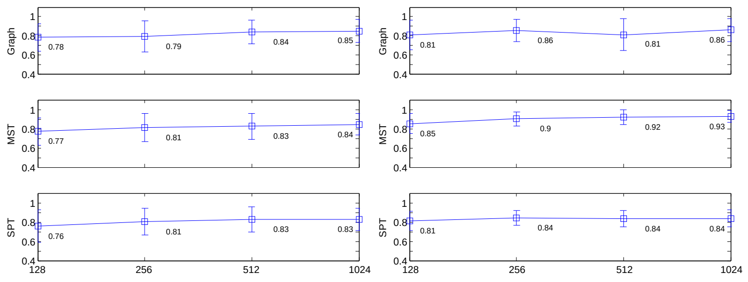
since for any path between p and q in the graph $\mathbf{G}$ there is in the SPT a path from p to the root and then to q which can be no longer than $2D_{\mathbf{G}}$. Thus, if the original graph was unweighted and consisted of a few dense clusters each uniquely labeled and with only a few cross-cluster edges, the tree built with a SPT would still have a non-vacuous mistake bound. This fact follows from equations (6) and (7). No such bound as (7) holds for a MST subgraph. For example, consider a bicycle wheel graph whose edge set is the union of n *spoke* edges $\{(0,i) : i = 1,\ldots,n\}$ and n *rim* edges $\{(i, i+1 \mod n) : i = 1,\ldots,n\}$ with costs on the spoke edges of 2 and on the rim edges of 1, the MST diameter is then $n+1$ while an SPT diameter is ≤ 8.

In general, we may not be given a tree rather we may be given a graph or a *similarity function* which may be then used to compute a tree. The MST and SPT trees may be constructed with Prim and Dijikstra algorithms [4] respectively in $O(n \log n + |E(\mathcal{G})|)$ time if implemented with a Fibonacci Heap. In the general case of a non-sparse graph or similarity function the time complexity is $\Theta(n^2)$, however as both Prim and Dijikstra are “greedy” algorithms their space complexity is $O(n)$ which may be dominant consideration in a large graph.

5 Experiments

We provide results on preliminary experiments to study the feasibility of our methods. In this exploratory study we have not yet implemented the fast computation technique described. Rather the aim of these experiments is to see if there is a significant performance decrease in using a tree subgraph rather than the original graph. The initial results are promising as we find (see Figure 1) that the accuracy of the predictor with an `MST` approximation is competitive with the original graph.

We performed experiments using the USPS digits dataset, with the aim of classifying ‘1’ vs. ‘2’ and then ‘3’ vs. ‘8’. Each OCR’d digit (a 16×16 array with 256 levels of gray) is represented as a vector $\mathbf{x} \in [-1, 1]^{256}$. The adjacency matrix for the graph was set to $A_{ij} = \exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2)$. The parameter a was then selected by grid search as the value which optimized generalization performance of the complete weighted graph ($n = 1024$) for both digit recognition tasks (‘1’ vs. ‘2’ : $a = 0.1$; ‘3’ vs. ‘8’ : $a = 0.05$). `MST` and `SPT` weighted subgraphs were then constructed. For each of the three graphs we computed the graph kernel $\mathbf{K} = \mathbf{G}^{+} + \mathbf{1}\mathbf{1}^{\top} + \mathbf{I}$ (where $\mathbf{I}$ is the identity matrix) as discussed in [5, equation (10)]. A training set $V^{(\ell)}$ of 8 labeled points of 4 positives and 4 negatives was randomly selected. A kernel perceptron was then trained for 3 epochs on $V^{(\ell)}$. The accuracy of each of the three classifiers was then measured on a randomly selected predictive set $V^{(p)}$ of 50 points. This protocol was performed for $n = 128, 256, 512, 1024$. Each such experiment was repeated 50 times, the mean accuracy is accordingly reported in Figure 1.



Digits ‘1’ vs. ‘2’ Digits ‘3’ vs. ‘8’

Fig. 1. Digit Classification (**Left:** Accuracy; **Bottom:** Vertex Set Size $|V^{(n)}|$)

The motivation for our methodology is the idea that accuracy can be improved by increasing the quantity of unlabeled data even if the quantity of labeled data is fixed at only a linear increase in computation cost. Experimentally this trend of increasing accuracy is seen for the graph and both the `MST` and the `SPT`; however this result is not conclusive as it may be an artifact of the tuning procedure for a and hence deserves further study. Interestingly the performance of the `MST` versus the complete weighted graph was found to be better in 5 out of 8 experiments even though the parameter a was optimally tuned for the complete graph. The `SPT` though competitive with the complete graph consistently under performed relative to the `MST`. We believe that these initial results show promise for our technique and we plan to implement the methods of Section 3 to compute the graph Laplacian pseudoinverse in order to scale our experiments to larger datasets.

Acknowledgements: We would like to thank José Luis Balcázar for valuable discussions and Massimiliano Pontil for valuable discussions and detailed comments on this manuscript. Finally, we would also like to thank Dimitrios Athanasakis and Sudhir Shankar Raman for useful preliminary experimentation. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778 and Sergio Rojas Galeano is supported under the DHPA Research Councils UK Scheme.

References

1. N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.
2. M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Machine Learning*, 56:209–239, 2004.
3. A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 13, New York, NY, USA, 2004. ACM Press.
4. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.
5. M. Herbster and M. Pontil. Prediction on a graph with a perceptron. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 577–584. MIT Press, Cambridge, MA, 2007.
6. M. Herbster, M. Pontil, and L. Wainer. Online learning over graphs. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 305–312, New York, NY, USA, 2005. ACM Press.
7. N.-D. Ho and P. V. Dooren. On the pseudo-inverse of the laplacian of a bipartite graph. *Appl. Math. Lett.*, 18(8):917–922, 2005.
8. D. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95, 1993.
9. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.
10. X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *In Proc. of the ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.

A Semi-Supervised Approach for Web Spam Detection using Combinatorial Feature-Fusion

Ye Tian, Gary M. Weiss, Qiang Ma

Department of Computer and Information Science
Fordham University
441 East Fordham Road
Bronx, NY 10458
{tian,gweiss,ma}@cis.fordham.edu

Abstract: This paper describes a machine learning approach for detecting web spam. Each example in this classification task corresponds to 100 web pages from a host and the task is to predict whether this collection of pages represents spam or not. This task is part of the 2007 ECML/PKDD Graph Labeling Workshop's Web Spam Challenge (track 2). Our approach begins by adding several human-engineered features constructed from the raw data. We then construct a rough classifier and use semi-supervised learning to classify the unlabelled examples provided to us. We then construct additional link-based features and incorporate them into the training process. We also employ a *combinatorial feature-fusion method* for "compressing" the enormous number of word-based features that are available, so that conventional machine learning algorithms can be used. Our results demonstrate the effectiveness of semi-supervised learning and the combinatorial feature-fusion method.

Keywords: feature construction, classification, link mining, information fusion, class imbalance.

1 Introduction

Search engines perform a vital role in permitting users to quickly and efficiently find information on the World Wide Web. Because web pages that are pointed to by many other web pages are favored by most search engines, spam web pages, which are created for the sole purpose of influencing search engine results, have become quite common. If the quality of search engine results is to be preserved, automated means of identifying these spam pages must be developed. The ECML/PKDD Graph Labeling Workshop's Web Spam Challenge supports the development of these methods, by providing a forum for researchers to develop, evaluate, and compare the effectiveness of a variety of methods. The second track of this challenge focuses on machine learning methods for identifying web spam, and in this paper we describe our machine learning entry for that challenge.

Our machine learning approach starts by supplementing the raw data provided by the Web Spam Challenge with additional, human-engineered content-based [1] and link-based [2] features. We then build a rough classifier using the raw data and these supplemental features, and use this classifier to label all of the unlabeled graph nodes provided by the Web Spam Challenge. Thus, we perform semi-supervised learning. We use the actual labels for the training data and the predicted labels for the other nodes to generate additional link-based features. In addition to the use of semi-supervised learning, we also use a combinatorial feature-fusion method, described in previous work [3,4], to "compress" the enormous number of content-based features, so that conventional classifier induction algorithms, which do not handle sparse features, can be used. Our results indicate that both semi-supervised learning and combinatorial feature fusion are effective at improving web spam detection.

This paper is organized as follows. In Section 2 we provide background on the combinatorial feature-fusion method. Then in Section 3 we describe the Web Spam Challenge data and the features that we manually engineer in order to improve the performance of our system. Section 4 describes our experimental methodology, including the feature-fusion method and use of semi-supervised learning. Our results from the Web Spam Challenge are then presented in Section 5. Finally, we describe our conclusions and areas for future work in Section 6.

2 Background on the Combinatorial Feature Fusion Method

In this section we describe the terminology and concepts associated with the combinatorial feature-fusion method, described in detail in previous work [3,4]. We use a simple example to describe the method. Our feature fusion method currently only handles numeric features, but this is not an issue for the Web Spam Challenge.

A data set is made up of examples, or records, each of which has a fixed number of features. Consistent with previous work on information fusion [3], we view the value of a feature as a *score*. Table 1 introduces a sample data set, with the score values in Table 1a replaced by rank values in Table 1b. The ranks in Table 1b have been computed in the straightforward manner, where a low score yields a low rank. This will not always be the case. As we will shortly see, the higher scores will receive lower ranks if this ranking scheme yields improved predictive performance.

Table 1. A sample data set with the data (a) unmodified and (b) with score values replaced by rank. The original data set contains eight examples, labeled A-H, with five numeric features and a binary class variable. In this example class 1 is the minority class and accounts for 3/8, or 37.5%, of the examples.

	F1	**F2**	**F3**	**F4**	**F5**	**Class**
A	1	4	3	2	8	1
B	3	3	5	5	4	0
C	5	5	2	6	7	1
D	7	6	15	3	2	0
E	11	13	16	7	14	0
F	15	16	4	13	11	0
G	9	7	14	1	18	1
H	17	15	9	8	3	0

(a)

	F1	**F2**	**F3**	**F4**	**F5**
A	1	2	2	2	5
B	2	1	4	4	3
C	3	3	1	5	4
D	4	4	7	3	1
E	6	6	8	6	7
F	7	8	3	8	6
G	5	5	6	1	8
H	8	7	5	7	2

(b)

Next we show how to compute the performance of a feature, using feature F2 from the sample data set as an example. For our purposes, the performance of a feature indicates how well its rank performs at predicting minority-class examples. The records in the data set are sorted by the rank value of F2 and the results are shown in Table 2a. The performance of F2 is then computed as the fraction of the records at the "top" of the table that belong to the minority class. The number of records considered is based on the percentage of minority-class examples in the training data, so in this case we look at the top 37.5%, or 3, records. In this case the performance of F2 is 2/3. Table 2b shows the performance values for all of the features.

Table 2. The results of ordering the examples by the rank of F2 is shown in Table 2a. The top three records are then used to compute the performance of F2, which in this case is 2/3, since 2 of the 3 records are associated with the minority class. The performance of all five feataures is provided in Table 2b.

	F2 Rank	**Class**
B	1	0
A	2	**1**
C	3	**1**
D	4	0
G	5	1
E	6	0
H	7	0
F	8	0

(a)

Feature	**Performance**
F1	0.67
F2	0.67
F3	0.67
F4	0.67
F5	0

(b)

This method is also used to compute the performance of combined (i.e., fused) features. However, to do this we need to determine the rank of a fused feature, so we can sort the examples by this rank. We compute this using a *rank combination function*, which averages the ranks of the features to be combined. This is done for each record. As an example, if we want to fuse features F1–F5 and create a new feature, then the rank of this fused feature for record A is computed as: (rank(F1) + rank(F2) + rank(F3) + rank(F4) + rank(F5))/5 = (1+2+2+2+5)/5 = 2.4. Once the rank values are computed, the performance value can be computed as before.

3 Data and Data Engineering

In this section we describe the raw data provided to us as part of the Web Spam Challenge and then describe the features that we design/engineer based on this raw data. We do not discuss the features automatically generated by our feature-fusion method or the link-based features generated as part of semi-supervised learning until Section 4.

The data utilized in this paper was provided as part of the Web Spam Challenge (track II, Corpus #1), held in conjunction with the 2007 ECML/PKDD Graph Labeling workshop. The data describes the content and link structure of about 9,000 examples, where each example describes 100 web pages from an Internet host. Each example can be viewed as a node in a graph, where nodes are connected if there are hyperlinks between them. The data is physically distributed over the following four data sets:

1. Feature Vectors: these vectors correspond to the TF-IDF vectors over the 100 web pages for a host. Thus this data set contains word frequency information. These are sparse vectors in that if a word does not occur, then it is not represented in the vector.
2. Link Matrix: each non-zero entry in this data set represents an edge in the graph and thus determines which nodes are connected by hyperlinks.
3. Training Labels: identifies the class label (spam or normal) for each node in the training set.
4. Validation Labels: identifies the class label (spam or normal) for each node in the "validation" set.

The raw data from the first two data sets are used to generate the examples for our classifier (we discuss this shortly). The training labels data set determines which examples are used for training while the validation labels data set determines which examples are used to evaluate the classifier to generate our preliminary results. The training data consists of 907 examples (hosts) and the validation data set contains 1800 examples. The test set labels are maintained by the Web Spam Challenge organizers, who use these labels to score the classifier results that are submitted by the challenge competitors. The class distribution of data, for both the training and validation set, is approximately 80% normal and 20% spam. This data is skewed, which can lead to some difficulties when learning a predictive model [5]. Since the training and validation data sets contain a total of 2,707 examples, 6,365 of the 9,072 examples are left unclassified. These examples could be ignored, but we use semi-supervised learning [6] to exploit them and improve our predictive model.

We manually engineered (i.e., constructed) seven features from the raw Web Spam Challenge data. Each feature is associated with a node/host and is computed based on information associated with that node/host. Table 3 summarizes the engineered features used in this study. Note hotwords are the words (i.e., content-based features) that appear in the greatest percentage of the 9,072 nodes provided in the Web Challenge Data. In order to limit the number of features for consideration we track only the top 500 hotwords (this may be too restrictive and should be increased in future work).

Table 3. Summary of Engineered Features. The first three features are content-based features and the remaining featurs are link-based features.

1	%HotwordsCovered	Percentage of the 500 hotwords found in the node
2	%Hotwords	Percentage of unique words in a node that are hotwords
3	TFIDF-Above-0.2	1 if any of the TF-IDF values is above 0.2; 0 otherwise
4	InboundLinks	The number of inbound links to this host
5	OutboundLinks	The number of outbound links from this host
6	InboundFraction	The fraction of total links that are inbound links
7	OutboundFraction	The fraction of total links that are outbound links

4 Experimental Methodology

We describe our experiments in this section. In Section 4.1 we describe the learning algorithms that we employ. Then in Section 4.2 we describe the basic procedure for encoding the examples for our learning problem. In Section 4.3 we describe how we generate new link-based features by using semi-supervised learning and in Section 4.4 we describe how we use combinatorial feature-fusion to construct new features.

4.1 Learning Algorithm

In this paper we evaluate three classifier induction algorithms, which are part of the Weka data mining package [7]. These algorithms are ADTree, an alternating decision tree algorithm [8], SMO, an implementation of a support vector machine, and Bayes, an implementation of a naïve Bayes classifier. Note that for alternating decision trees the ultimate classification is determined by multiple paths through the tree rather than a single path through the tree. Since our preliminary results showed that ADTree performed best, only classifiers induced using this algorithm were used as part of the official Web Spam Challenge.

4.2 Example Generation

The examples for the training and validation data sets start with the features included in the feature vector data set described in Section 3. These include the TF-IDF values for the content-based features. Since this information is provided using a sparse representation and our machine learning methods do not handle sparse representations of features, we insert null entries for the missing features. Because our learning algorithms can not handle the enormous number of resulting content-based features, we prune all but the top 200 such features based on the performance values produced by our feature-fusion method on the training data, as described in Section 2. This should keep the features that are best able to predict web spam, since the performance metric is based on the ability to predict the minority-class examples. Next, we join these 200 features with the seven engineered features listed in Table 3. We then add the class labels for the training and validation data using the class information provided as part of the Web Spam Challenge.

4.3 Semi-Supervised Learning

Once the examples are generated as described in Section 4.2, we use the training data to build a "rough" classifier. This is then used to classify the nodes *not* in the training set. We then use the predicted class labels for these non-training set nodes and the actual class labels for the training set nodes to determine, for each link in the link matrix, whether the inbound or outbound side is spam or normal. From this we construct four new link-based features, which indicate the prevalence of spam for the neighbors of a node. These features represent, for each node, the number of inbound (outbound) spam links and the percentage of inbound (outbound) links that are spam.

We could use these four newly constructed features to build our final model, but instead choose to include an additional round of semi-supervised learning, on the assumption that it will yield superior results. That is, we train a classifier using all previous features plus these four new ones and then *again* predict the class labels for the nodes that are not in the training set. We then recompute these four link-based features using the updated labels. These link-based feature values are then included for use in building our final classification model, once the combinatorial feature-fusion method described in Section 4.4 is used to construct some additional features.

4.4 Use of Combinatorial Feature-Fusion

Next we use our combinatorial feature-fusion strategy to introduce new "fused" features. Section 2 described how to fuse features and evaluate their performance, but did not discuss how we decide which features to fuse and how to determine which of these fused features to keep. In previous work we tried a variety of fusion strategies [4], but in this paper we simply fuse all pairs of features. Given that we have a total of 211 features, this leads to C(211,2), or 22,155, possible pairings. While restricting the fusion to only two features is limiting, we will see that it still improves classifier performance.

We next decide which of the fused features to keep. We order the fused features by their performance value and then tentatively add them in one by one. After adding each feature, we regenerate the classifier from the training data and compare the performance of the classifier on a hold-out set (selected from the training data) with the performance prior to adding the feature. If the feature yields an improvement in performance with a t-test confidence of at least .9, then we keep the feature; otherwise we discard it. Since we measure classifier performance using AUC and the F-measure, for the purpose of deciding whether to keep a feature or not we measure performance using the average value of AUC and the F-measure.

5 Results

In this section we report our preliminary results on the validation data provided to us, using three classifiers, and then present the official Web Spam Challenge results, calculated by the competition organizers. Our preliminary results are summarized in Table 4. These results are reported with and without the various enhancements so the impact of these enhancements can be evaluated. In all cases the seven basic engineered features from Table 3 are included. The results in Table 4 show that when neither semi-supervised learning nor feature-fusion is used, the results are not very good, and that both of these enhancements yield consistent improvements in all measures of classifier performance. The results also demonstrate that the classifier generated by ADTree yields the best overall performance, and hence we submit only the classifiers induced using this algorithm for the Web Spam Challenge.

Table 4. Summary of Preliminary Classifier Performance Results. The table shows the results without any enhancements, then using semi-supervised learning, then using semi-supervised learning and the feature-fusion method.

Classifier	Metric	Enhancements		
		Initial	Semi-supervised learning	Semi-supervised learning + fusion
ADTree	AUC	.753	.824	.931
	F-Measure	.291	.523	.716
	Precision	.553	.611	.797
	Recall	.370	.457	.649
SMO	AUC	.581	.613	.628
	F-Measure	.301	.373	.410
	Precision	.320	.368	.392
	Recall	.285	.247	.284
Bayes	AUC	.703	.762	.772
	F-Measure	.451	.495	.517
	Precision	.345	.368	.392
	Recall	.651	.758	.763

The results for the official Web Spam Challenge are provided in Table 5. These results are presented separately for the validation data and the test data even though the validation data was not used to train the classifier (the class labels for the test data were not provided to the competitors). Two sets of results are reported since the competition permitted two classifiers to be submitted. In our case the differences were minor in that a few features were ommitted for the second classifier.

Table 5. Summary of our Classifier Perrformance for the Official Competition. All results are based on the ADTree classifier, using both the semi-supervised and fusion enhancements.

Evaluated Data	AUC	Precision	Recall	Accuracy (thresh=0.5)	Accuracy (thresh=optimal)
Validation	.889	.766	.438	.839	.850
Validation	.884	.794	.338	.821	.838
Test	.864	.725	.406	.825	.842
Test	.854	.738	.318	.803	.832

The results in Table 5 show reasonable values for AUC, but show relatively low values for recall. In particular, the values for the validation data in Table 5 are worse than those for the same data in Table 4. Although some last minute changes that were made for the competition may have negatively impacted our results, the only change that we are aware of that should have had a negative impact is that for our preliminary results the validation data was inadvertently used to calculate the performance values for the feature fusion method—but *not* used to train the actual classifier. It is not clear to us that this should cause a substantial difference in overall performance.

6 Conclusion and Future Work

In this paper we describe a machine learning approach for identifying web spam. Our approach involves adding human-engineered features and then using semi-supervised learning to exploit the unlabeled examples that are provided as part of the Web Spam Challenge data. We also use our combinatorial feature-fusion method in order to reduce the number of TF-IDF content-based features and to construct new features that are combinations of these features. We feel that our final results are reasonable and, perhaps more significantly, we show that both our feature-fusion strategy and use of semi-supervised learning lead to dramatically improved classification performance.

We see many opportunities for future work. Many of these opportunities relate to our feature fusion method. First, we would like to employ a heuristic version of the feature fusion method so that we can handle more than 200 features and also generate more complex feature combinations. Also, the rank combination used in our feature fusion method assigns equal weight to each feature and we would like to learn the optimal weights for combining these ranks. Since we assign a rank based only on the positive or negative magnitude of the score value, our method will not handle the case well where the most predictive value occurs in the middle. We could address this by binning the numerical values and then ranking the bins based on their predictive value. Using a similar idea we could also use the feature fusion method to handle non-numerical values.

We would also like to further study the semi-supervised learning method that we employed. We would like to try to generate more sophisticated link-based features, determine how the number of iterations of semi-supervised learning impacts classifier performance, and determine if the values of the link-based features that we construct converge after several iterations of semi-supervised learning.

References

1. Ntoulas, A., Najork, M., Manasse, M., Fetterly, D. Detecting spam web pages through content analysis, Proceedings of the 15th International World Wide Web Conference (2006)
2. Becchetti, L., Castillo, C., Donato, D., Leonardi, S., Baeza-Yates, R. Link Based Characterization and Detection of Web Spam, Workshop on Adversarial Information Retrieval on the Web (2006)
3. Hsu, D.F., Chung, Y., Kristal, B. Combinatorial fusion analysis: methods and practices of combining multiple scoring systems. Advanced Data Mining Technologies in Bioinformatics. Hershey, PA: Idea Group Publishing; 32–62 (2006)
4. Tian, Y., Weiss, G., Hsu, D.F., Ma, Q. A Combinatorial Fusion Method for Feature Mining, Proceedings of KDD'07 Workshop on Mining Multiple Information Sources (2007)
5. Weiss, G. M. Mining with rarity: A unifying framework. SIGKDD Explorations, 6(1): 7-19.
6. Chapelle, O., Scholkopf, B., Zien, A. Semi-Supervised Learning. MIT Press, Cambridge, MA. (2006).
7. Freund, Y., Mason, L. The alternating decision tree learning algorithm. Proceedings of the Sixteenth International Conference on Machine Learning, 124-133 (1999).
8. Markov, Z., Russell, I. An introduction to the WEKA data mining system. In Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education. 367–368 (2006)

Web Spam Challenge 2007 Track II Secure Computing Corporation Research

Yuchun Tang, Yuanchen He, Sven Krasser, and Paul Judge

Secure Computing Corporation
4800 North Point Parkway, Suite 300
Alpharetta, GA 30022, USA
{ytang,yhe,skrasser,pjudge}@securecomputing.com
http://www.trustedsource.org

Abstract. To discriminate spam Web hosts/pages from normal ones, text-based and link-based data are provided for Web Spam Challenge Track II. Given a small part of labeled nodes (about 10%) in a Web linkage graph, the challenge is to predict other nodes' class to be spam or normal. We extract features from link-based data, and then combine them with text-based features. After feature scaling, Support Vector Machines (SVM) and Random Forests (RF) are modeled in the extremely high dimensional space with about 5 million features. Stratified 3-fold cross validation for SVM and out-of-bag estimation for RF are used to tune the modeling parameters and estimate the generalization capability. On the small corpus for Web host classification, the best F-Measure value is 75.46% and the best AUC value is 95.11%. On the large corpus for Web page classification, the best F-Measure value is 90.20% and the best AUC value is 98.92%.

Key words: Web Spam, Web Linkage Graph, Support Vector Machine, Random Forest

1 Introduction

It is important to detect deliberate actions of deception to increase the ranking of targeted Web pages or Web hosts on search engines for internet search providers. Web can be naturally represented by a graph, where each node corresponds to a Web page/host and a directed edge from node A to node B denotes the number of hyper links from A to B. The goal of the Web Spam Challenge is to utilize machine learning methods for automatically labeling nodes to be "spam" or "normal" in such a graph. The challenge is labeling all nodes of a graph from a partial labeling of them. In the given datasets, only 10% nodes are manually labeled by human experts. For the Track II of the challenge, a single standard set of features has been provided for each node. Readers are suggested to refer `http://webspam.lip6.fr/` for more details.

2 Experiments

The experiments are conducted with several steps including text-based feature preprocessing, link-based feature preprocessing, SVM modeling, dimensionality reduction, and RF modeling.

2.1 Text-based Feature Preprocessing

On the small corpus, the features are normalized Term Frequency-Inverse Document Frequency (TF-IDF) values, and we directly use them for classification modeling.

On the large corpus, the features are TF values. The maximal TF value in the corpus is 39041, so we simply divide each value by 39401 to scale each feature into [0,1].

2.2 Link-based Feature Preprocessing

Given a node A in a Web linkage graph, 9 features are extracted and listed in Table 1.

Table 1. Link-based Feature Extraction

id	name	meaning
l01	Od	the number of links from A to other nodes
l02	Odn	the number of links from A to other known normal nodes
l03	Ods	the number of links from A to other known spam nodes
l04	Id	the number of links from other nodes to A
l05	Idn	the number of links from other known normal nodes to A
l06	Ids	the number of links from other known spam nodes to A
l07	Bd	the number of other nodes that are connected with A in both directions
l08	Bdn	the number of other known normal nodes that are connected with A in both directions
l09	Bds	the number of other known spam nodes that are connected with A in both directions

Notice that we remove self-connection links, which otherwise induce noise when calculating the number of links to and/or from known nodes.

We also extract other 25 features as shown in Table 2.

The value of these 25 features is defined to be 0 if the corresponding denominator is 0.

These 34 link-based features are standardized into [0,1] before being fed into classification modeling. The standardization formula is $(x - min)/(max - min)$.

2.3 SVM Modeling

Table 3 lists characteristics of the datasets after preprocessing. The small dataset is for Web host classification while the large one for Web page classification.

Table 2. More Link-based Feature Extraction

id	meaning
l10	Odn/Od
l11	Ods/Od
l12	Idn/Id
l13	Ids/Id
l14	$Odn/(Odn + Ods)$
l15	$Ods/(Odn + Ods)$
l16	$Idn/(Idn + Ids)$
l17	$Ids/(Idn + Ids)$
l18	$Od + Id$
l19	$Odn + Idn$
l20	$Ods + Ids$
l21	$(Odn + Idn)/(Od + Id)$
l22	$(Ods + Ids)/(Od + Id)$
l23	$(Odn + Idn)/(Odn + Idn + Ods + Ids)$
l24	$(Ods + Ids)/(Odn + Idn + Ods + Ids)$
l25	Bd/Od
l26	Bd/Id
l27	Bdn/Odn
l28	Bdn/Idn
l29	Bds/Ods
l30	Bds/Ids
l31	Bdn/Bd
l32	Bds/Bd
l33	$Bdn/(Bdn + Bds)$
l34	$Bds/(Bds + Bds)$

Table 3. Characteristics Of Datasets

dataset		#features	#samples	#normal : #spam
small	training	4,924,007	907	701:206
	validation		1,800	1,412:388
large	training	4,924,007	40,000	32,083:7,917
	validation		80,000	63,874:16,126

We select Support Vector Machines (SVM) [1] for classification because it demonstrates good performance on high dimensional datasets [2]. The LIBSVM software package, which is available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`), is used for SVM modeling.

We conduct SVM modeling with both linear and RBF kernels. 3-fold cross validation is conducted on the training dataset for parameter tuning. We tune the cost parameter c, and the gamma parameter γ if RBF kernel is used. We also tune the weight parameter w_1 while $w_0 = 1$ because there are much more normal nodes than spam nodes in the training datasets. A value of 0 indicates the normal class while a value of 1 denotes spam. All other parameters are fixed at default values in LIBSVM.

On the small corpus for Web host classification, the best F-Measure value is 73.22% and the best AUC value is 93.26%, which are achieved by modeling a RBF SVM with $c = 2^{10}$, $\gamma = 0.05$ and $w_1 = 2$.

On the large corpus for Web page classification, the best F-Measure value is 90.20% and the best AUC value is 98.92%, which are achieved by modeling a linear SVM with $c = 2^{14}$ and $w_1 = 4$. SVM modeling with RBF kernel cannot achieve better performance in our experiments.

The ROC curves are shown in Figures 1-2. The dotted and dashed curves denote 3-fold cross validation performance on the given training datsets while the solid curves denote prediction performance on the given validation datasets.

It is interesting to observe that linear SVMs have almost the same or even better performance than RBF SVMs. The reason is that the number of features is already much higher than the number of samples. Hence, it is not very helpful to conduct classification modeling in a even higher feature space with RBF transformation.

2.4 Dimensionality Reduction

With a linear SVM, features can be ranked based on their contribution to SVM classification. This is the basic idea of the Support Vector Machine - Recursive Feature Elimination (SVM-RFE) algorithm [3]. By applying SVM-RFE on the small dataset, 28,051 features are selected from the original 4,924,007 features for Web host classification. Most of link-based features contribute to classification as demonstrated in Table 4. Almost the same accuracy can be achieved before and after feature selection. It seems that dimensionality reduction can improve efficiency and may also generate more insights for Web spam detection. This is an interesting future work.

2.5 RF Modeling

We also implement another learning process similar to Random Forests modeling [4] on the small dataset with the 28,051 features. Firstly, we generate 100 datasets with 100% bootstrapping. Secondly, we randomly select 1000 features and remove all other features for each dataset. Thirdly, a C4.5 decision tree

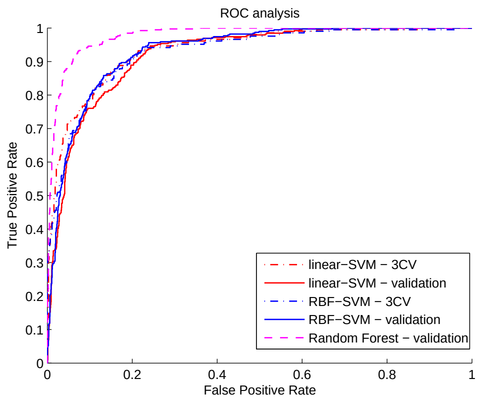


Fig. 1. ROC curves for Web host classification on the small dataset

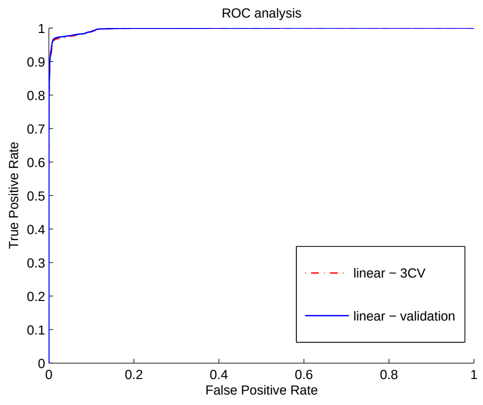


Fig. 2. ROC curves for Web page classification on the large dataset

Table 4. Link-based Features are highly ranked in SVM-RFE

id	ranking	id	ranking	id	ranking
l05	5	l34	490	l31	14020
l04	26	l01	869	l22	16252
l02	88	l06	950	l13	16706
l30	108	l21	1855	l28	17954
l19	116	l12	1876	l26	18122
l20	131	l27	3593	l23	21806
l03	151	l33	3604	l17	21911
l32	239	l15	5380	l14	24149
l11	242	l10	8017	l24	removed
l29	323	l09	10818	l08	removed
l07	326	l25	13970	l16	removed
l18	428				

[5] is modeled on each dataset in Weka with default values. Weka is available at `http://www.cs.waikato.ac.nz/ml/weka/`. Lastly, the decision values of the 100 decision trees are summed up and averaged for the final decision. Instead of 3-fold cross validation for SVM modeling, out-of-bag estimation on the training dataset is used to estimate generalization capability. This modeling method generates a even better performance with 75.46% F-Measure value and 95.11% AUC value. In Fig 1, this method demonstrates higher ROC curve than SVM counterparts. Due to limited computing power, we cannot run SVM-RFE feature selection and random forest modeling on the large dataset. But a quick look on the linear SVM shows that only 83,926 features contribute to the SVM classification.

Table 5 summarizes experiment results. For SVM methods, both 3-fold cross validation performance on the training dataset and prediction performance on the validation dataset are reported. For tree and forest methods, we report out-of-bag performance on the training dataset and prediction performance on the validation dataset.

Table 5. Experiment Results

Task	Method	F-Measure			AUC		
		CV/OB	validation	testing	CV/OB	validation	testing
host classification	RF with C4.5	78.74%	75.71%	75.46%	96.24%	95.84%	**95.11%**
	RBF SVM	72.50%	72.95%	73.22%	92.74%	93.12%	**93.26%**
page classification	Linear SVM	94.53%	94.96%	90.20%	99.64%	99.66%	**98.92%**

3 Conclusion

The data preprocessing and machine learning methods described in this paper demonstrate high accuracy on Web Spam Challenge corpora. On the small corpus for Web host classification, the best F-Measure value is 75.46% and the best AUC value is 95.11%. On the large corpus for Web page classification, the best F-Measure value is 90.20% and the best AUC value is 98.92%.

References

1. V. N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, New York, 1998.
2. C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
3. I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.
4. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
5. R. J. Quinlan, *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann, 1993.

Semi-supervised classification with hyperlinks

J. Abernethy[1] and O. Chapelle[2]

[1] Department of Computer Science, UC Berkeley
[2] Yahoo! Research

1 Notations

Our method takes as input:

- a set of l labeled examples $(\mathbf{x}_1, y), \ldots, (\mathbf{x}_l, y_l)$, where $\mathbf{x}$ denotes the feature vector and y the $\pm$ label.
- a set of u unlabeled examples, $(\mathbf{x}_{l+1}, y_{l+1}), \ldots, (\mathbf{x}_n, y_n)$, with $n = l + u$.
- a weighted directed graph whose nodes are $\mathbf{x}_1, \ldots, \mathbf{x}_n$. a_{ij} is the weight of the link from $\mathbf{x}_i$ to $\mathbf{x}_j$.

The goal is to learn a linear classifier $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ in an *inductive* mode, i.e. such that it can be evaluated on a unseen test example not belonging to the graph. The graph is thus used only during training and its semantic is problem dependent. In the case of this competition, the information we have from the graph is that there is usually no link from a non-spam host to a spam host.

We thus minimize:

$$\lambda \mathbf{w} \cdot \mathbf{w} + \sum_{i=1}^{l} \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i))^2 + \gamma \sum_{i \leftarrow j} a_{ij} \Phi(\mathbf{w} \cdot \mathbf{x}_i, \mathbf{w} \cdot \mathbf{x}_j).$$

The first two terms correspond to a standard linear SVM, while the third one takes the graph information into account. The function Φ is chosen such that it penalizes predictions with links from non-spam hosts to spam hosts; we took it to be:

$$\Phi(a, b) = \max(0, b - a)^2.$$

The objective function can be efficiently minimized in the primal, see for instance [1].

The parameters λ and γ are selected on a validation set.

References

[1] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 2006.

Webspam detection via Semi-Supervised Graph Partitioning

Chris Biemann, Hans Friedrich Witschel

{biem|witschel}@informatik.uni-leipzig.de

Abstract. The aim of our experiments for the WebSpam challenge was twofold: first to explore a mixture of a link graph and a document similarity graph; and second to adapt an efficient graph clustering algorithm to a semi-supervised functionality. The results on the validation sets suggest that page content can be ignored and that the semi-supervised partitioning works very well, especially on the large set.

1 Graph building

In order to build a mixed content-link graph, we first turned the directed link graph into an undirected one. Then, a document similarity graph was constructed in the following way: for each rare term t that occurred $n_t < n$ times in the whole collection of web pages, a list of all documents $d_1, ..., d_{n_t}$ containing t was constructed. The free parameter n defines the notion of "rare term" and depends on the size of the collection.

The lists $d_1, ..., d_{n_t}$ were treated as *sentences* of natural language and fed to the corpus production engine *tinyCC*[1] that is usually used for analysis of large text corpora. TinyCC efficiently computes - for all pairs of words that co-occur in sentences - whether the number of joint occurrences deviates significantly from statistical independence.

Applied to the documents of the WebSpam challenge, this produces a list of pairs (d_i, d_j) of documents that co-occur more often than expected in "document sentences" $d_1, ..., d_{n_t}$, which means that they share many rare terms. For each pair, there is also a significance value $w_{content}(d_i, d_j)$, which can be used as an edge weight when interpreting the list of pairs as a document similarity graph.

Mixing of the two graphs was performed by linearly combining edge weights: $w(d_i, d_j) = \alpha w_{link}(d_i, d_j) + (1 - \alpha) w_{content}(d_i, d_j)$. Since the link graph is originally unweighted, $w_{link}(d_i, d_j)$ was set to the average weight of all edges in the content graph for all document pairs (d_i, d_j) in order to make edge weights comparable among the two graphs. The parameter α was then varied in order to determine how much influence should be given to content and links, respectively.

2 Semi-supervised graph partitioning

Orginally, Chinese Whispers [1] is a parameter-free, randomised graph partitioning algorithm that has linear run-time in the number of edges, allowing the

[1] http://wortschatz.uni-leipzig.de/˜cbiemann/software/TinyCC2.html

processing of very large graphs. For the purpose of web spam detection, the we employed a yet unpublished semi-supervised version of this algorithm, which is outlined in the following algorithm on graph $G(V, E)$, training T.

```
for all $v_i \in V$ do
    $class(v_i) = -1$
end for
for all $v_i \in T$ do
    $class(v_i)$ =training class
end for
for it=1 to number-of-iterations do
    for all $v \in V \setminus T$, randomised order do
        $class(v)$=predominant class in $neigh(v)$
    end for
end for
return partition $P$ induced by class labels
```

The algorithm starts by initialising all nodes according to their training classification, all other nodes get label -1. Then, for a couple of iterations (we chose 10 in the experiments), all nodes get updated in random order and inherit the predominant class in the neighbourhood. The dominance per class a for node v is computed locally in the neighbourhood $neigh(v)$ by:

$$dominance(a, v) = \frac{\sum_{w \in neigh(v), class(w)=a} ew(v, w) \cdot nw(w)}{\sum_{w \in neigh(v)} ew(v, w) \cdot nw(w)}.$$

The initialisation class -1 has always dominance 0. Here, $ew(v, w)$ denotes the edge weight between nodes v and w as given in the graph, $nw(w)$ is the node weight. In preliminary experiments, we determined $nw(w) = \frac{1}{degree(w)}$, i.e. the influence of nodes is weighted down linearily with the number of edges to other nodes. This weighting scheme is motivated by the following: pages that have many outgoing or ingoing links should be less important when propagating classifications w.r.t. spamicity.

3 Results

The results on the two validation sets given for the challenge suggest that:

- Content can be ignored: on the small set results were somewhat inconclusive as to the optimal value of α, but $\alpha = 1$ was near-optimal for various values of n. On the large set, $\alpha = 1$ was always optimal.
- The semi-supervised graph partitioning seems to work very well, especially on large data sets: the best precision obtained was 88.72% on the small set and 99.58% (184/63814 errors for non-spam, 151/16126 errors for spam) on the large set. Spamicity grading is given by $dominance(spam, v)$.

References

1. Chris Biemann. Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of the HLT-NAACL-06 Workshop on Textgraphs-06*, 2006.

SpamChallenge 2007 - Track II: France Telecom R&D Submissions

Pascal Filoche and Tanguy Urvoy and Marc Boullé

France Telecom R&D, Lannion, France

1 Overview

We submitted 2 predictions for each graph:

- First prediction uses a selective bayesian classifier ;
- Second prediction uses graph-based smoothing on first prediction scores.

2 Selective Bayesian classification

We use a naive Bayes classifier with optimal discretization, variable selection and model averaging. The univariate class conditional probabilities are estimated using the *MODL* discretization method [1], which enables to retrieve the most probable discretization model given the data.

The naive Bayes classifier is based on the assumption that the variables are independant within each output label. To leverage this asumption, we use the compression-based averaging method described in [2]. A Bayesian regularization technique is applied to select the most probable subset of variables compliant with the naive Bayes assumption. Furthermore, an averaging method is used to exploit the posterior distribution of the selective naive Bayes models. Experiments show these techniques, optimal preprocessing, variable selection and model averaging, allow to consistently improve the performance of the naive Bayes classifier.

This classifier is trained on the 10000 most frequent attributes from TF.IDF feature vectors and the result of a standard PageRank [4] computation on the graph. A *spamicity* score is then computed by the classifier for each node of the graph.

On the small (*hosts*) graph data, 1480 features are selected by the classifier (PageRank is ranked 24th). Evaluation of the score on validation data returns an optimal F1 measure for spam of 0.723. Evaluation on challenge test data gives an optimal F1 measure for spam of 0.738.

On the large (*URLs*) graph data, 5700 features are selected (PageRank is ranked 32nd). Evaluation of the classifier on validation data returns an optimal

F1 measure for spam of 0.937 Evaluation on challenge test data returns an optimal F1 measure for spam of 0.935.

3 Graph-based smoothing of classification

During a second phase, the nodes computed spamicities is iteratively smoothed by neighbourhood in the graph: $x \rightarrow y$ edges are randomly picked and values of connected vertices are gradually changed to reduce distance between their scores:

$$S_{n+1}(x) = (1-\delta)S_n(x) + \delta S_n(y),\ and$$
$$S_{n+1}(y) = (1-\delta')S_n(y) + \delta' S_n(x).$$

As in [3], the intuition for this algorithm is that edges between nodes of the same class outnumber edges between nodes of different classes. As described in Fig. 1, this property is well verified for the big graph but not for the small one.
Using this method with $\delta = \delta' = 0.001$, optimal F1 on validation reaches 0.75 for the *small* graph and 0.97 for the *large* one.

$x \rightarrow y$	y normal	y spam
x normal	**3761**	27
x spam	**576**	**501**

$x \rightarrow y$	y normal	y spam
x normal	**99703**	128
x spam	569	**31572**

Fig. 1. Edge distribution in training data for small and large graph (ignoring self-loops). For the small graph the spamicity of the source is less informative: ideally the smoothing coefficients δ and δ' should be tuned accordingly.

References

1. Marc Boullé. MODL: a Bayes optimal discretization method for continuous attributes. *Machine Learning*, 65(1):131–165, 2006.
2. Marc Boullé. Compression-based averaging of selective naive Bayes classifiers. *Journal of Machine Learning Research*, 8:1659–1685, 2007.
3. Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: Web spam detection using the web topology. In *Proceedings of ACM SIGIR (to appear)*, Amsterdam, Netherlands, 2007.
4. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

Author Index

Abernethy, Jacob, 32

Benczúr, András A., 1
Biemann, Chris, 33
Boullé, Marc, 35

Chapelle, Olivier, 32
Csalogány, Károly, 1

Filoche, Pascal, 35

He, Yuanchen, 24
Herbster, Mark, 9

Judge, Paul, 24

Krasser, Sven, 24

Lukács, László, 1

Ma, Qiang, 16

Rojas Galeano, Sergio, 9

Siklósi, Dávid, 1

Tang, Yunchun, 24
Tian, Ye, 16

Urvoy, Tanguy, 35

Weiss, Gary M., 16
Witschel, Hans Friedrich, 33