Web Spam Detection Using Transductive–Inductive Graph Neural Networks

Anas Belahcen[1,2], Monica Bianchini[1], and Franco Scarselli[1]

[1] Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche
Università degli Studi di Siena – Siena, Italy
[2] LeRMA – ENSIAS
Mohammed V Souissi University – Rabat, Morocco
{monica,franco}@diism.unisi.it,
belahcen.anas@gmail.com

Abstract. The Web spam detection problem has received a growing interest in the last few years, since it has a considerable impact on search engine reputations, being fundamental for the increase or the deterioration of the quality of their results. As a matter of fact, the World Wide Web is naturally represented as a graph, where nodes correspond to Web pages and edges stand for hyperlinks. In this paper, we address the Web spam detection problem by using the GNN architecture, a supervised neural network model capable of solving classification and regression problems on graphical domains. Interestingly, a GNN can act as a mixed transductive–inductive model that, during the test phase, is able to classify pages by using both the explicit memory of the classes assigned to the training examples, and the information stored in the network parameters. In this paper, this property of GNNs is evaluated on a well–known benchmark for Web spam detection, the WEBSPAM–UK2006 dataset. The obtained results are comparable to the state–of–the–art on this dataset. Moreover, the experiments show that performances of both the standard and the transductive–inductive GNNs are very similar, whereas the computation time required by the latter is significantly shorter.

1 Introduction

In several application areas, data are naturally represented as graphs or trees, e.g., in computer vision, molecular biology, software engineering and natural language processing. As a matter of fact, nodes in these structures are used to represent objects, while edges determine the relationships between them. For example, the World Wide Web is commonly described by a graph, where nodes represent Web pages and edges stand for hyperlinks. In the Web graph, nodes and edges may have vector labels, collecting the information available about the page contents and the hyperlinks, respectively.

Traditional machine learning approaches try to reduce graphical data into simple representations, as, e.g., a set of vectors. In this way, the topological information may be lost during the preprocessing step, which can deeply affect the achieved performance. On the other hand, the Graph Neural Network (GNN) model [1] is capable of processing graphs directly, without any preprocessing step. GNNs are supervised neural network models that extend the recursive paradigm, and can be applied on most of the practically useful kinds of graphs, including directed, undirected, labeled and cyclic graphs. GNNs have been successfully employed in several application domains, such as molecule classification, object localization in images, and Web page ranking.

In this paper, we apply GNNs to Web spam detection, i.e., the problem of classifying a Web page as a document containing spam or not. Such a problem has received a growing interest in the last few years, due to its importance for search engines [2–4]. In Web spam detection, the Web graph can be used both for learning and testing. During training, we use a small set of pages, for which the target is known, to learn the GNN parameters. Then, the trained GNN is applied on the whole Web graph, to classify the remaining Web pages. GNNs are well suited for Web spam detection, since they can learn to automatically classify pages, exploiting both the information available on the page contents and on the Web connectivity.

Interestingly, a GNN can operate using two modalities: it can act as a pure inductive model or as a mixed transductive–inductive model. In the former case, during the test phase, the GNN completely relies on its parameters to classify Web pages. The actual classification of a training page is not explicitly memorized, so that such page can be even misclassified by the GNN. With the transductive–inductive modality, the classification of the training pages is explicitly added to the Web graph. In this way, the GNN operates also as a transductive model, that classifies test pages by using the information already available for the training pages, and by diffusing such an information through the Web graph.

In order to evaluate our approach, we tested GNNs on a well–known benchmark for Web spam detection, the WEBSPAM–UK2006 dataset [5], finding promising preliminary results.

2 The Graph Neural Network Model

Graph Neural Networks (GNNs) are a supervised connectionist model capable of solving classification and regression problems on graphical domains. One of the major advantages of GNNs is their capacity of processing graphs directly (without preprocessing), which preserves the information collected into the graph topology.

In fact, graph nodes are used to represent concepts, while edges determine the relationships between them. Each concept or node in the graph is defined by its features, and also by the information contained in its neighborhood. Based on these two information sources, the GNN calculates a state x_n, for each node n, which contains the node representation (see Fig. 1). Then, using this state, the GNN produces an output that denotes the classification decision on that node. Formally, the output of a GNN is defined by the following equations:

$$\begin{aligned} x_n &= f_w\big(l_n, l_{co[n]}, x_{ne[n]}, l_{ne[n]}\big), \\ o_n &= g_w(x_n, l_n), \end{aligned} \tag{1}$$

where f_w and g_w are parametric functions, implemented by two feedforward neural networks, which express the dependence of the state at each node on the state of its neighborhood, and the dependence of the node output on its state, respectively.

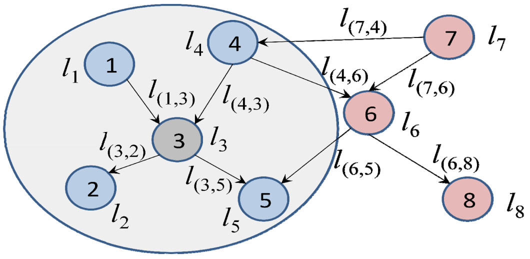

Fig. 1. A graph and, in evidence, the neighborhood of a node. The state x_3 of node 3 depends on the information contained in its neighborhood. The transition and the output functions are, respectively defined as $x_3 = f_w\left(l_3, l_{(1,3)}, l_{(3,2)}, l_{(4,3)}, l_{(3,5)}, x_1, x_2, x_4, x_5, l_1, l_2, l_4, l_5\right)$, $o_3 = g_w(x_3, l_3)$

Moreover, $l_n, l_{co[n]}, x_{ne[n]}, l_{ne[n]}$ represent the label of n, the labels of its attached edges, and the states and the labels of the nodes in its neighborhood, respectively.

In order to compute the output defined by Eq. (1), the Banach Fixed Point Theorem suggests the following classic iterative scheme:

$$\begin{aligned} x_n(t+1) &= f_w\big(l_n, l_{co[n]}, x_{ne[n]}(t), l_{ne[n]}\big), \\ o_n(t) &= g_w(x_n(t), l_n), \end{aligned} \tag{2}$$

for each node n. Intuitively, the computation described by Eq. (2) can be interpreted as the activity of a network consisting of units which compute f_w and g_w. Such a network, built by replacing each node of the graph with a unit computing f_w (see Fig. 2), will be called the encoding network. Each unit stores the current state $x_n(t)$ at node n and, when activated, it calculates $x_n(t+1)$ (Fig. 2). The output at n is produced by another unit which implements g_w.

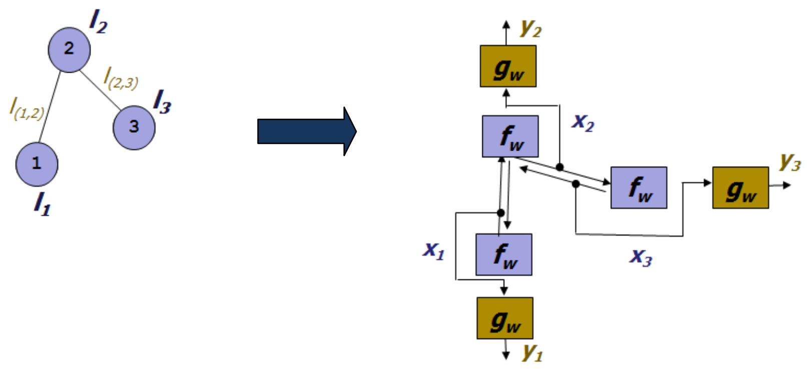

Fig. 2. The graph (on the left) and the corresponding encoding network (on the right). Graph nodes (circles) are replaced by ad hoc units computing f_w and g_w (squares). When f_w and g_w are implemented by feedforward neural networks, the encoding network is a recurrent network.

More details on the GNN training algorithm and output computation can be found in [1]. Here, it suffices to say that both training and test sets consist of a labelled graph which, in our application, is a portion of the Web graph. For the training set, also targets for some nodes are provided, which define the actual class of these nodes, i.e. whether corresponding pages are spam or not. The training procedure adapts the network parameters in order to produce the correct outputs on the supervised pages, while the test procedure uses the trained GNN to classify the remaining pages.

As mentioned in Section 1, GNNs can be exploited either as a common parameterized inductive model or as a mixed transductive–inductive model. In the inductive setting, the network is fed with the Web graph, using supervised pages to adapt the GNN parameters. Hence, in this way, the information contained in the training set is used to approximate a classification function that can be used to directly classify the nodes of the Web graph. On the other hand, in the transductive–inductive model, during training, a subset of the supervised pages is assigned a label enriched with their class membership, whereas the remaining (the class membership label is unset) are used for training – i.e. they contribute to the calculus/optimization of the error function. Instead, during testing, a component of the label of each training page explicitly specifies whether such a page is spam or not (the class membership label is unset for unsupervised pages), so that the information available on the training pages *is directly diffused* through the Web graph.

3 The WEBSPAM–UK2006 Dataset

In order to assess our approach, we evaluate the GNN model on the WEBSPAM–UK2006 dataset. Actually, the dataset was adopted in 2007 by the Web Spam Challenge, a competition held annually during the International Workshop on

Adversarial Information Retrieval on the Web. The Web graph is a crawl of the .uk domain that includes 77.9 million pages and over 3 billion links in 11,402 hosts. The labeling was at the host level, i.e., the assessors labeled the hosts as normal or spam. Such a benchmark is particularly suited for our purpose both because it has been used by several research groups and because it is sufficiently large to produce significant results and, at the same time, not too huge to prevent a wide experimentation.

3.1 Features

Data are represented by the following features: (1) link–based features, which include, f.i., the indegree and the outdegree of hosts and their neighbors, PageRank and TrustRank; (2) content–based features, which include, f.i., the fraction of anchor and visible text, the compression rate, the corpus precision (the fraction of words in a page that belong to the set of popular terms), and the corpus recall (the fraction of popular terms that appear in the page).

3.2 Feature Preprocessing

The WEBSPAM–UK2006 dataset includes 41 link–based and 96 content–based features, which are used as node labels. Due to the high number of features, we use a feedforward neural network in order to summarize and compress them into a single one. More precisely, different configurations were used for GNNs, as it follows.

- **Link and content–based features directly**: The most significant link and content–based features are selected, using a correlation–based feature selector [6], and integrated as the node label.
- **Link–based feature**: A feedforward neural network is employed in order to compress all the link–based features into a single output. This output is then used as the node label.
- **Content–based feature**: A feedforward neural network is employed in order to compress all the content–based features into a single output. This output is then used as the node label.
- **Compressed and uncompressed features:** Link and content–based features, already compressed by feedforward networks, in addition to some features directly selected (in particular, the PageRank and the TrustRank of the host and of its maximum scored page) are collected together and then used as the node label.

3.3 Teams Participating to the 2007 Web Spam Challenge

We compare our results with those gained by the six teams participating to the 2007 Web Spam Challenge. The competition attracted three teams from academic institutions (Hungarian Academy of Sciences, University of Waterloo, and Chinese Academy of Sciences) and three teams from industry research laboratories (Genie

Knows, Microsoft, and France Télécom). Results obtained by competing teams are shown in Table 3. Notice that, after the competition, other groups have worked on these benchmarks, but their results are difficult to be comparatively evaluated, due to the fact that, in most cases, the original splitting between training and test sets has not been used.

4 Experimental Results

In this section, we present the experimental results obtained by GNNs. The experiments are divided into two parts. The first one uses the original training/test splitting, already fixed in the challenge, whereas in the second one the splitting is randomly constructed from all the dataset pages. The choice of testing the approach on a random splitting is a common procedure and it is motivated by the presence of different the data distributions in the original training and test sets. Besides, for each splitting, an inductive learning model and a mixed transductive–inductive learning approach were used. As mentioned before, in the transductive–inductive setting, the training pages were divided into two groups, in order to define the error function and to simulate the transductive inference, respectively. In our experiments, two equal–size groups were randomly defined. Finally, the performance was measured by the area under the ROC curve, the F–measure, and the accuracy.

4.1 The Random Splitting

The WEBSPAM–UK2006 dataset was randomly split into training (2228 hosts), validation (1000 hosts), and test (2518 hosts) sets. The results are divided according to whether GNNs are used as an inductive or a mixed transductive–inductive learning model. Table 1 shows the performance obtained by different GNN configurations.

Each row represents a different simulation, as described below:

- In the first experiment, the most significant link and content–based features were chosen, using a correlation–based feature selector [6]. In fact, according to a preliminary experiment, ten link–based and two content–based features were selected. The performance was the lowest compared to the other configurations.
- In the second and in the third experiment, a feedforward neural network was used, in order to compress all the features into a single output (each type of features has its own output). Exploiting this idea, the performance of the model increases.
- In the last configuration, we combine link and content–based features, already compressed by feedforward networks, with directly selected features (i.e., PageRank and TrustRank of the host and of its maximum PageRank page). With this experiment, we obtain the highest performance.

For most of the experiments, the results obtained in the transductive–inductive learning framework are slightly better. In Table 3, we compare the results of our best GNN configuration with those produced by the other teams, proving that it gains very similar performance to the winner.

Table 1. Performances of different GNN configurations with random splitting

	Accuracy		F-Measure		ROC	
Configurations	Transd.-Ind.	Induc.	Transd.-Ind.	Induc.	Transd.-Ind..	Induc.
Link and content directly	89,48%	89,27%	0.7550	0.7528	0.9467	0.9446
Link based (FFNN)	90,30%	90,27%	0.7680	0.7763	0.9506	0.9516
Content based (FFNN)	90,50%	90,11%	0.7532	0.7531	0.9417	0.9387
Link and content (FFNN) and directly selected	**94,08%**	**93,96%**	**0.8534**	**0.8499**	**0.9681**	**0.9717**

4.2 The Original Splitting

The splitting adopted in the Challenge was also used for the experiments. In this case, the training set includes 8415 hosts (7472 normal, 767 spam, and 175 undecided), while the test set contains 2247 hosts (651 normal, 1346 spam, and 250 undecided). The architecture used to address this classification problem is shown in Fig. 3.

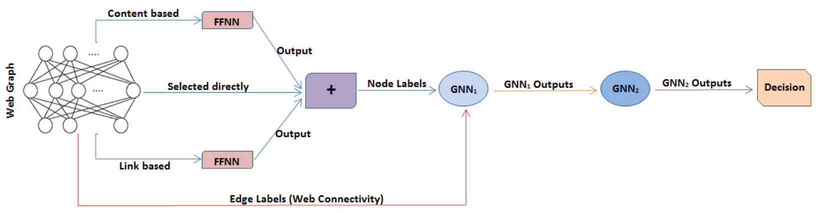

Fig. 3. The configuration adopted in the original splitting.

In this case, content–based and link–based features compressed into single features (by feedforward networks) are used, in addition to features directly selected, to construct the whole labels for the GNN processing. The produced output will be used in a second GNN. The output of the second GNN will be the decision on the hosts, classified as spam or normal. As in the random splitting experiments, this GNN configuration can be used as an inductive or a mixed transductive–inductive model. The obtained results are shown in Table 2.

Table 2. Performance comparison with the original splitting

	Accuracy		F-Measure		ROC	
Configurations	Transd.	Induc.	Transd.	Induc.	Transd.	Induc.
Link and content (FFNN) and directly selected	**89,53%**	**89,23%**	**0.9219**	**0.9215**	**0.9496**	**0.9502**

Based on the experiments, we observe that the standard inductive and the transductive–inductive models are comparable in terms of performance.

Nevertheless, some more experiments are worth carrying out in order to clearly establish the GNN ability in addressing the proposed problem. In the following, we compare the performance obtained in our experiments with respect to those of the other competing teams, and also evaluate training times of both the transductive–inductive and the inductive models, with respect to the random and the original splitting.

4.3 Performance Comparison

The experiments, based on both original and random splitting, show that our results are comparable to the best results obtained so far on the WEBSPAM–UK2006.

Table 3. Comparative results

Participants	F1	ROC
Abou *et al.* (Genie Knows)	0.81	0.80
Benczur *et al.* (Hungarian Academy of Sciences)	0.91	0.93
Cormack (University of Waterloo)	0.67	0.96
Fetterly *et al.* (Microsoft)	0.79	-
Filoche *et al.* (France Télécom)	0.88	0.93
Geng *et al.* (Chinese Academy of Sciences)	0.87	0.93
Random splitting	**0.85**	**0.97**
Original splitting	**0.92**	**0.95**

4.4 Training Time Comparison

Even if the two learning frameworks show comparable performances, they significantly differ in terms of training time (see Fig. 4).

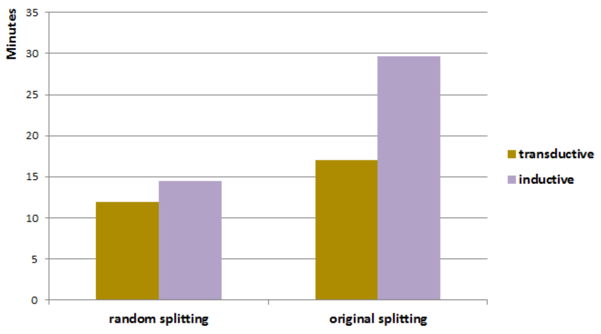

Fig. 4. Comparison between the transductive–inductive and the inductive configurations with respect to the training time

Actually, the training time for the inductive model is greater than that for the transductive–inductive approach, with the random splitting, of about 20%, while, with the original splitting, it increases of 76%, which means that the transductive–inductive model is as efficient as the inductive model, but it is certainly very less expensive from the computational point of view.

5 Conclusions

According to experiments conducted on the WEBSPAM−UK2006 dataset, the results obtained using the random and the original splitting can be compared, in terms of performance, to the state–of–the–art results. Besides, the experiments were realized based on both transductive–inductive and inductive frameworks, which show different training times, clearly assessing the advantages of the transductive–inductive approach from the computational point of view.

References

1. Scarselli, F., Gori, M., Tsoi, A.-C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. on Neural Networks 20(1), 61–80 (2009)
2. Di Noi, L., Hagenbuchner, M., Scarselli, F., Tsoi, A.-C.: Web Spam Detection by Probability Mapping GraphSOMS and Graph Neural Networks. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010, Part II. LNCS, vol. 6353, pp. 372–381. Springer, Heidelberg (2010)
3. Gyöngyi, Z., Garcia-Molina, H.: Web spam taxonomy. Adversarial information retrieval on the Web (2005)
4. Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M., Vigna, S.: A reference collection for Web spam. ACM SIGIR Forum 40(2), 11–24 (2006)
5. Web spam challenge (2007), http://webspam.lip6.fr/wiki/pmwiki.php?n=Main.PhaseIResults
6. Hall, M.A.: Correlation–based Feature Subset Selection for Machine Learning. Hamilton, New Zealand (1998)