Graph Neural Networks for Ranking Web Pages

Franco Scarselli
University of Siena
Siena, Italy
franco@ing.unisi.it

Sweah Liang Yong
University of Wollongong
Wollongong, Australia
sly56@uow.edu.au

Marco Gori
University of Siena
Siena, Italy
marco@ing.unisi.it

Markus Hagenbuchner
University of Wollongong
Wollongong, Australia
markus@uow.edu.au

Ah Chung Tsoi
Australian Research Council
Canberra, Australia
ahchung.tsoi@arc.gov.au

Marco Maggini
University of Siena
Siena, Italy
maggini@ing.unisi.it

Abstract

An artificial neural network model, capable of processing general types of graph structured data, has recently been proposed. This paper applies the new model to the computation of customised page ranks problem in the World Wide Web. The class of customised page ranks that can be implemented in this way is very general and easy because the neural network model is learned by examples. Some preliminary experimental findings show that the model generalizes well over unseen web pages, and hence, may be suitable for the task of page rank computation on a large web graph.

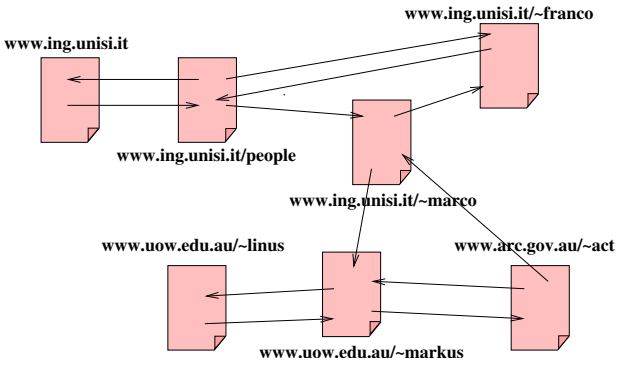

Figure 1. A subset of the web represented as a graph. The arrows denote hyperlinks.

1. Introduction

The World Wide Web is naturally represented as a graph where the nodes represent web pages and the arcs represent hyperlinks (see Fig. 1). The web connectivity can be exploited for the design of efficient web page ranking algorithms. In fact, web page ranking algorithms [3, 14] are a fundamental component of search engines. Their goal is to provide each web page a rank, i.e. a measure that predicts how important and authoritative the page will be considered by users. Such rank is exploited by search engines, together with other features, to sort the web pages (documents) returned in response to user queries.

Google's PageRank [3] is a well known example in this class which exploits the Web connectivity to measure the "authority" of documents. In PageRank, a page is considered "authoritative" if it is referred by many other pages and if the referring pages are, in turn, authoritative. Formally, the PageRank x_n of a page n is computed by

$$x_n = d \sum_{u \in \mathrm{pa}[n]} \frac{x_u}{h_n} + (1-d)\,, \qquad (1)$$

where h_n is the outdegree of n, $\mathrm{pa}[n]$ is the set of parents of n, and $d \in (0,1)$ is a damping factor [3].

While PageRank and most of the other page ranking algorithms are designed to serve general purpose search engines, some recent approaches provide specialized rankings which are suited for particular requirements [6, 12, 13, 17, 20]. The main underlying idea of these approaches is that the concept of "page importance" is not absolute but depends on the particular needs of a search engine or a user. For example, the homepage of a large directory may be authoritative for a general purpose search engine, but it may not be important for a search engine specialized on a topic such as "Wine". The research on this issue is particularly ac-

Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)
0-7695-2415-X/05 $20.00 © 2005 **IEEE**

tive, since the related technology may be useful for a number of applications. For example, it may allow to build user–centric interfaces which can adapt, according to the user preferences, the sorting of the documents returned in response to queries. Similarly, a search engine may use specialized ranks to provide or to sell a service customized for particular communities or companies. In the simplest case, the administrator of a small search engine (e.g. the search engine installed in the Intranet of a company or the search engine used in a portal specialized on a given topic) may wish to customize the ranking algorithm to the particular environment. In fact, approaches have been proposed to build page ranks specialized on a topic [6, 20], a user [13], or a query [12].

Notice that a ranking algorithm can be represented by a function τ that takes as input the web graph $\boldsymbol{G}$ and a page n and returns a score $\tau(\boldsymbol{G}, n) \in \mathbb{R}$. Very recently, a new neural network model called Graph Neural Network (GNN) [9] has been proposed which is able to directly process graphs. The approach extends and unifies some previous methods [8, 10, 11, 20] of graph processing using neural networks. A GNN can realize general ranking algorithms: in [18], it is proved that GNNs can approximate in probability most of the useful functions $\tau : \mathcal{G} \times \mathcal{N} \rightarrow \mathbb{R}^m$, where $\mathcal{G}$ is a set of graphs and $\mathcal{N}$ the corresponding set of nodes.

Moreover, a GNN can be trained by examples, i.e. τ can be constructed using a small set of patterns $\{(n_1, t_1), \ldots, (n_q, t_q))\}$, where n_i is a page and t_i is its desired rank. Training examples can also be provided by constraints as $t_i \geq t_j$, which express the fact that "the rank of page i must be larger than rank of page j".

In this paper, we adapt the GNN model to make it particularly suited for the web page ranking problem. The presented approach allows the automatic customization, based on training examples, of the rank to the particular needs of a search engine or Internet user, thus, simplifying the design of the page ranking algorithm. On the contrary, most of the previous approaches were based on a semi-manual design of a specialized rank which is a difficult task. To the best of our knowledge, there are only three methods which adapt the rank [4, 7, 20]. However, these approaches exploit simple models based on parameterized versions of the original PageRank equation (1). Thus, due the general approximation capabilities of GNNs, the proposed solution implements a more general class of ranking functions. Moreover, no other methods except for [20] allow the use of training sets in the form of constraints on page rank values [1]. Note that in this paper we do not address the issue of how training samples can be acquired. This is a matter of future research as the solution depends heavily on the environment where the proposed ranking algorithm is used.

1 Defining constraints between pages is a more natural task for the user than the task of providing the exact rank corresponding to each page.

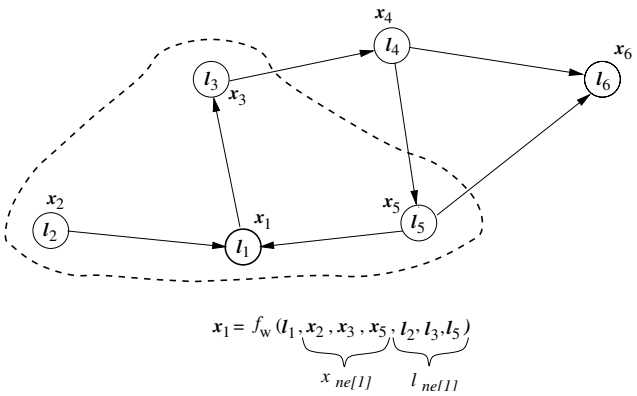

Figure 2. The dependence of state x_1 on neighborhood information.

The structure of this paper is as follows: Graph Neural Networks are introduced in Section 2. A specialized GNN model for web page ranking is introduced in Section 3. Some experimental results are presented in Section 4, and conclusions are drawn in Section 5.

2. Graph neural networks

In the following, a graph $\boldsymbol{G}$ is a pair $(\boldsymbol{N}, \boldsymbol{E})$, where $\boldsymbol{N}$ is a set of nodes and $\boldsymbol{E}$ a set of edges. The set ne$[n]$ are the nodes connected to n by arcs. Nodes may have labels, which are assumed to be real vectors. The labels attached to n will be represented by $\boldsymbol{l}_n \in \mathbb{R}^{l_N}$. Labels usually include features of the object corresponding to the node. For example, in the case of the World Wide Web, node labels may represent properties of the page, e.g. the list of words contained in the document, the URL, size and location of multimedia content, the topic discussed in the page, etc.

The intuitive idea underlining GNNs is that nodes in a graph represent objects or concepts and edges represent their relationships. Thus, to each node n a vector $\boldsymbol{x}_n \in \mathbb{R}^s$, called *state*, can be attached which collects a representation of the object denoted by n. $\boldsymbol{x}_n$ is naturally specified using the information contained in the neighborhood of n (see Figure 2).

$$\boldsymbol{x}_n = \sum_{u \in \text{ne}[n]} h_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{x}_u, \boldsymbol{l}_u), \; n \in \boldsymbol{N}. \qquad (2)$$

More precisely, let $h_{\boldsymbol{w}}$ be a feedforward neural network, called *transition network*, that expresses the dependence of a node on its neighborhood and is parameterized by a set of parameters $\boldsymbol{w}$. The states $\boldsymbol{x}_n$ are defined as the solution of the following system of equations: For each node n, an output $\boldsymbol{o}_n \in \mathbb{R}$ is defined which depends on the state $\boldsymbol{x}_n$ and la-

bel $\boldsymbol{l}_n$. The dependence is described by a parameterized *output network* $g_{\boldsymbol{w}}$

$$\boldsymbol{o}_n = g_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{l}_n),\ n \in \boldsymbol{N}. \tag{3}$$

Thus, Eqs. (2) and (3) define a method to produce an output (e.g. a rank) $\boldsymbol{o}_n$ for each node, i.e. a parameterized function $\varphi_{\boldsymbol{w}}(\boldsymbol{G}, n) = \boldsymbol{o}_n$ which operates on graphs. The corresponding machine learning problem consists of adapting the parameters $\boldsymbol{w}$ such that $\varphi_{\boldsymbol{w}}$ approximates the data in the learning data set $\mathcal{L} = \{(n_i, \boldsymbol{t}_i) | 1 \leq i \leq q\}$, where each pair $(n_i, \boldsymbol{t}_i)$ denotes a node n_i and the corresponding desired output $\boldsymbol{t}_i$. In practice, the learning problem is implemented by the minimization of a quadratic error function

$$J_{\boldsymbol{w}} = \sum_{i=1}^{q} (\boldsymbol{t}_i - \varphi_{\boldsymbol{w}}(\boldsymbol{G}, n_i))^2 . \tag{4}$$

In fact, to the best of our knowledge, all the existing ranking algorithms based on web connectivity are a particular case of GNNs. For example, note that Eqs. (2) and (3) extends Eq. (1) for the following reasons:

- In Eq. (2), the dependence between $\boldsymbol{x}_n$ and the states of the neighbors of n is defined by a generic nonlinear function, whereas in Eq. (1) the PageRank depends linearly on neighbors' PageRanks.
- In Eq. (2), the dependency of state $\boldsymbol{x}_n$ is extended to the children of n, while PageRank depends only on the parents. Such an extension was already used in [6, 14].
- GNN uses an output function, PageRank does not.

In order to implement the GNN formally defined by Eqs. (2) and (3), the following items must be provided:

(1) A method to solve Eqs. (2) and (3). This is addressed in Section 2.1

(2) A learning algorithm to learn the parameters of $h_{\boldsymbol{w}}$ and $g_{\boldsymbol{w}}$ from data from the training set. This is addressed in Section 2.2.

2.1. Computing the states

The solutions $\boldsymbol{x}_n$, $\boldsymbol{o}_n$ of system (2), (3) can be obtained by iterating:

$$\boldsymbol{x}_n(t+1) = \sum_{u \in \mathrm{ne}[n]} h_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{x}_u(t), \boldsymbol{l}_u)), \tag{5}$$

$$\boldsymbol{o}_n(t+1) = g_{\boldsymbol{w}}(\boldsymbol{x}_n(t+1), \boldsymbol{l}_n), \qquad n \in \boldsymbol{N}. \tag{6}$$

In [9, 18], it is proved that, under simple conditions on the transition network h_w, the sequences $\boldsymbol{x}_n(t)$ and $\boldsymbol{o}_n(t)$ converge exponentially to $\boldsymbol{x}_n$ and $\boldsymbol{o}_n$, respectively, i.e. $\lim_{t\to\infty} \boldsymbol{x}_n(t) = \boldsymbol{x}_n$ and $\lim_{t\to\infty} \boldsymbol{o}_n(t) = \boldsymbol{o}_n$. This method, which is just the Jacobi Algorithm for solving nonlinear systems, is the same one used by Google to compute PageRank. The approach is quite efficient in practice, because even if the web is represented by a very large graph, the exponential convergence ensures that the solution can be computed in few tens of iterations [2, 3].

Let us consider two computing units $\sum h_{\boldsymbol{w}}$ and $g_{\boldsymbol{w}}$ that calculate the terms on the right hand side of Eqs. (5) and (6), respectively. Note that the computation described in Eqs. (5) and (6) can be interpreted as the representation of a neural network consisting of instances of $\sum h_{\boldsymbol{w}}$ and $g_{\boldsymbol{w}}$. Such a neural network is called an *encoding network*.

In order to build the encoding network, each node of the graph can be replaced by a unit computing the function $\sum h_{\boldsymbol{w}}$ (see Figure 3). Each unit stores the current state $\boldsymbol{x}_n(t)$ of the corresponding node n, and, when activated, it calculates the state $\boldsymbol{x}_n(t+1)$ using the labels and the states stored in its neighborhood. The simultaneous and repeated activation of the units produces the behavior described by Eq. (5). In the encoding network, the output of node n is produced by another unit which implements $g_{\boldsymbol{w}}$.

The encoding network is a recurrent network, the transition and the output function can be implemented as a multilayered feedforward neural network.

2.2. A learning algorithm

The learning algorithm consists of two phases:

(a) the states $\boldsymbol{x}_n(t)$ are iteratively updated, using Eqs. (5) and (6) until they reach a stable fixed point $\boldsymbol{x}_n(T) = \boldsymbol{x}_n$ at time T;

(b) the gradient $\frac{\partial J_{\boldsymbol{w}}(T)}{\partial \boldsymbol{w}}$ is computed and the weights $\boldsymbol{w}$ are updated according to a gradient descent strategy.

Thus, while phase (a) moves the system to a stable point, phase (b) adapts the weights to change the outputs towards the desired target. These two phases are repeated until a given stopping criterion is reached. This algorithm, which was introduced in [9], is a combination of the backpropagation through structure algorithm, which is adopted for training recursive neural networks [19], and the Almeida–Pineda algorithm [1, 16]. The latter is a particular version of the backpropagation through time algorithm which can be used to train recurrent networks with cycles [19].

3. A GNN model for page rank computation

A specialized implementation of $h_{\boldsymbol{w}}$ and $g_{\boldsymbol{w}}$ is required in order to reach a better performance on the web page ranking problem. In this paper the GNN is defined as:

$$x_n = \sum_{u \in \mathrm{pa}[n]} h_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{x}_u, \boldsymbol{l}_u) = \sum_{u \in \mathrm{pa}[n]} \boldsymbol{A}_{n,u} \boldsymbol{x}_u + \boldsymbol{b}_n \tag{7}$$

$$\boldsymbol{o}_n = g_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{l}_n) = \boldsymbol{x}'_n \cdot \pi_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{l}_n) . \tag{8}$$

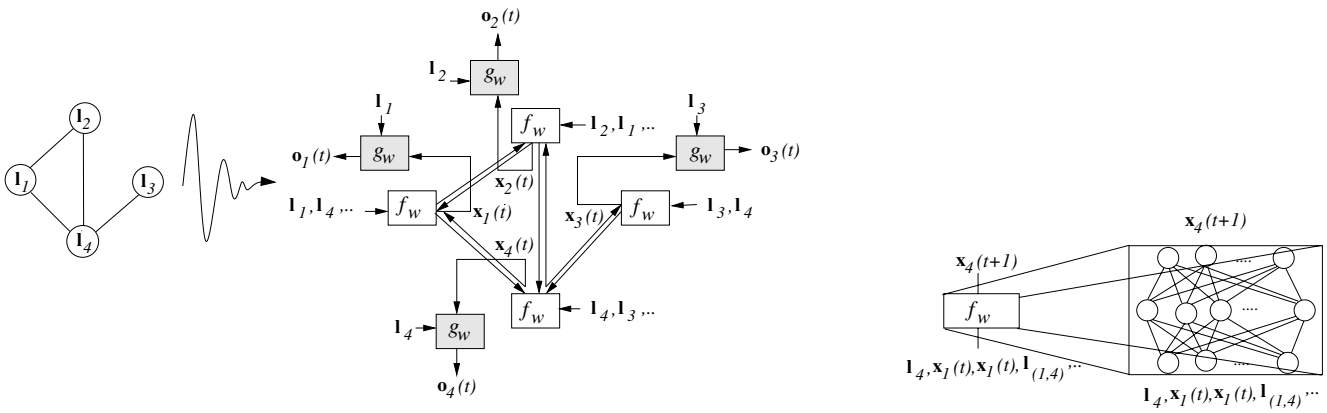

Figure 3. A graph and its corresponding encoding network (left), and the transition function implemented as a multi-layered feedforward Neural Network (right).

where $\prime$ is the transpose operator and $\pi_{\boldsymbol{w}} : \mathbf{R}^{s+\boldsymbol{l}_N} \rightarrow \mathbf{R}^s$ is the function implemented by a feedforward neural network with s outputs. Moreover, also $\boldsymbol{b}_n \in \mathbf{R}^s$ and the matrix $\boldsymbol{A}_{n,u} \in \mathbf{R}^{s \times s}$ are defined by the outputs of two feedforward neural networks respectively, whose parameters correspond to the parameters of the GNN. More precisely, let $\phi_{\boldsymbol{w}} : \mathbf{R}^{2l_N} \rightarrow \mathbf{R}^{s^2}$ and $\rho_{\boldsymbol{w}} : \mathbf{R}^{l_N} \rightarrow \mathbf{R}^s$ be the functions implemented by two feedforward networks. Then, we define

$$\begin{aligned} \boldsymbol{A}_{n,u} &= \frac{\mu}{s|\mathrm{ne}[u]|} \cdot \mathrm{Resize}\left(\phi_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_u)\right) \\ \boldsymbol{b}_n &= \rho_{\boldsymbol{w}}(\boldsymbol{l}_n)\,, \end{aligned}$$

where $\mu \in (0, 1)$ and $\mathrm{Resize}(\cdot)$ is the operator that allocates the elements of s^2-dimensional vector into a $s \times s$ matrix.

A formal explanation of the reasons that support the above GNN model is given in [9, 18]. Here, we provide the following intuitive explanation. The GNN defined by (7) and (8) is a sort of linear system as the one used to specify PageRank (see Eq. (1)) and most of the other ranking algorithms. However, whereas in other approaches the parameters of the linear system are predefined or are the same for each page, in our case $\boldsymbol{A}_{n,u}$ and $\boldsymbol{b}_n$ are defined by the neural networks $\phi_{\boldsymbol{w}}$ and $\rho_{\boldsymbol{w}}$ on the basis of page content and features. Thus, in our case those parameters are learned by examples. Moreover, due to the approximation capabilities of feedforward neural networks the class of page rank applications that can be realized in this way is wider [18].

Finally, it is worth noting that it is shown in [9, 18] that if $\|\phi_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_u)\|_1 \leq s^2$ holds, where $\|\cdot\|_1$ denotes the one norm for vectors and matrices, then the Jacobi Algorithm described in Section 2.1 always converges to the unique solution of system (7), (8). The above condition can be straightforwardly verified if the output neurons of the transition network uses an appropriately bounded activation function, e.g. a hyperbolic tangent.

4. Experimental results

The model was tested on the WT10G dataset distributed by CSIRO, Australia, and was used as a benchmark at the Text Retrieval Conference (TREC). The WT10G is a snapshot of a portion of the World Wide Web. The collection pays special attention to the connectivity among the web pages. Hence this set of collection is particularly suitable for evaluations of Internet search engine applications, e.g., page ranking algorithms. There are 1,692,096 documents contained in the dataset. Thus, the graph $\boldsymbol{G}$ used for our experiments was the WT10G connectivity graph. The label $\boldsymbol{l}_n$ attached to page n consisted of a vector representing the topics discussed in n. The pages were classified into eleven topics by a set of naive Bayes classifiers [15]. The topics included classes such as "Windows", "Linux", "Wine", "Sports", "Surgery", etc. We used Bayes classifiers as these are well established and popular methods in the area of text classification.

In a real life application, usually only few examples are available for training, since these examples must be manually labelled or collected by an interaction with the user. In order to simulate such an environment, the training dataset consisted of a small subgraph $\bar{\boldsymbol{G}}$ of WT10G. Moreover, only few pages of $\bar{\boldsymbol{G}}$, which we will call the supervised pages, were used as examples and were involved in the error function (4). On the other hand, the test set consisted of the whole WT10G graph $\boldsymbol{G}$.

For each page n_i, a desired target t_i was given, which was used for learning purpose if it is in the training dataset, and for evaluating the performance of the model if it is a page in the test set. Moreover, in some experiments, Google's PageRank was used to generate the targets t_i. In those cases, the PageRanks pr_i are computed using Eq. (1) on the training set $\bar{\boldsymbol{G}}$, if the page belonged to the supervised pages, and on the whole dataset, if the page belonged

to the test set.

Three simple experiments were devised in order to evaluate the properties of GNNs. The main goal of the experimentation was to verify whether the proposed model can learn a simple ranking function on a small set of training examples and whether it can generalize such a function to the whole WT10G dataset.

For sake of simplicity, all the feedforward neural networks involved in the GNN model, i.e. the $\tau_{\boldsymbol{w}}, \phi_{\boldsymbol{w}}$ and $\rho_{\boldsymbol{w}}$ of Section 3, were three layer (one hidden layer) feedforward neural networks with 5 hidden neurons. Experiments were carried out using the Matlab development environment on a 2GHz, Intel based PC. The GNN was trained for 2500 epochs which required less then 2 minutes of computation time when the training graph $\bar{\boldsymbol{G}}$ contained 4000 nodes. We also observed that training times increased linearly with the size of the training set. The test on the whole WT10G ($1,692,096$ pages) required around 4 minutes.

4.1. Increasing the focus on a selected topic

This first experiment simulates the case where the rank is specialized for a user (or a specialized search engine) interested in a selected topic. This is demonstrated by training a GNN such that pages addressing the selected topic "Sport" receive twice Google's PageRank while all other pages are to remain at Google's PageRank. Formally, we have $t_i = 2pr_i$, if page n_i discussed the topic "Sport" and $t_i = pr_i$, otherwise.

The experiment used just 20 random supervised pages equally subdivided in pages which address this topic and pages outside the topic of interest. In addition, 3980 random pages were used to form the training set graph $\bar{\boldsymbol{G}}$ [2]. For evaluating the performance of the model, we considered the pages with an error percentage smaller than 5% on the testing data [3]. The experiment was repeated several times and consistently exceeded 99% accuracy. The result shown in figure 4 presents the absolute page position when sorted by page rank value in descending order. More precisely, pages are sorted both according to PageRank and the rank computed by our model. Each page n_i is assigned a pair (a_i, b_i), where a_i is the position of the page in the sorting established by PageRank and b_i is the position according to GNN. Finally, the page is represented by a symbol at the coordinate (a_i, b_i). Thus, a page above (below) the diagonal is ranked higher (lower) by GNN than by PageRank. It can be observed that pages addressing "Sport" consistently rank higher. This is the expected consequence when doubling the page rank value of selected pages.

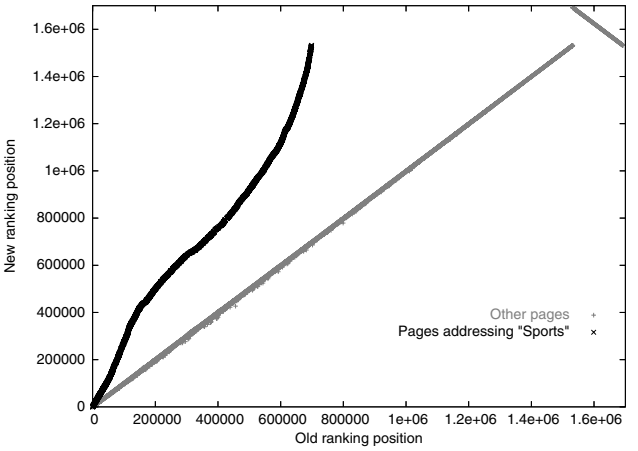

Figure 4. The results achieved when doubling the page rank of pages on "Sport".

4.2. An Exclusive OR-like problem

The second experiment considered two topics: "Sports" and "Surgery". The rank function to be learned is ($t_i = 2pr_i$) for pages classified as either "Sports" or "Surgery", and ($t_i = pr_i$) for pages which addressed both topics, and for pages which did not address either topic. This experiment simulates the situation when a user is interested in a set of topics, but does not like general documents that describe many topics.

Experiments were conducted using a training set consisting of 10 pages which are about "Sports", 10 pages on "Surgery", 10 about "Sport" and "Surgery", and 10 pages which addressed neither "Sport" nor "Surgery". Other 20 supervised pages were randomly selected from the children and parents of the above pages [4]. The network performed often at 98% accuracy, and produced a result as illustrated in Figure 5. It can be seen that the pages addressing either of the topics increased significanly in their ranking, while pages addressing both topics remain close to the original (Google's) page rank.

When conducting the experiments we observed a poor performance in a few cases. This may be attributed to a local minimum issue. In general, neural network learning rules are based on gradient descend methods which can get caught in a local minimum on the error surface. In practice, one way in which this can be overcome is by retrain-

2 Those pages belonged to the training set but were not supervised.

3 We consider a node is on target when the computed rank differs from the target rank by $\pm 5\%$. For example, a computed rank of 10 is considered on target if the target rank is larger than 9.6 and smaller than 10.4.

4 The parent pages and children pages may not necessarily be of the selected topics, but they are useful since they ensure that the training set contains connected pages.

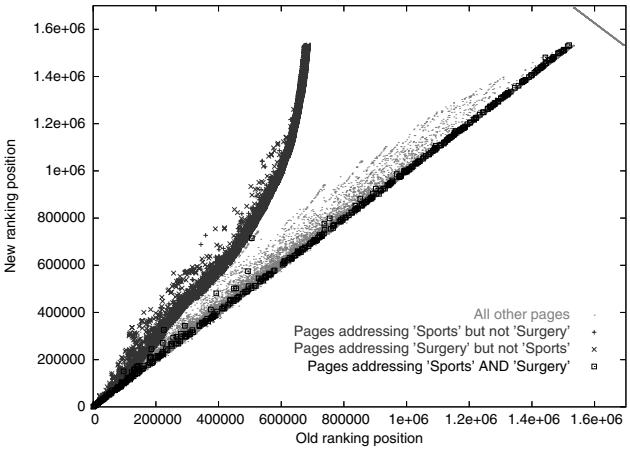

Figure 5. The results achieved on the XOR problem.

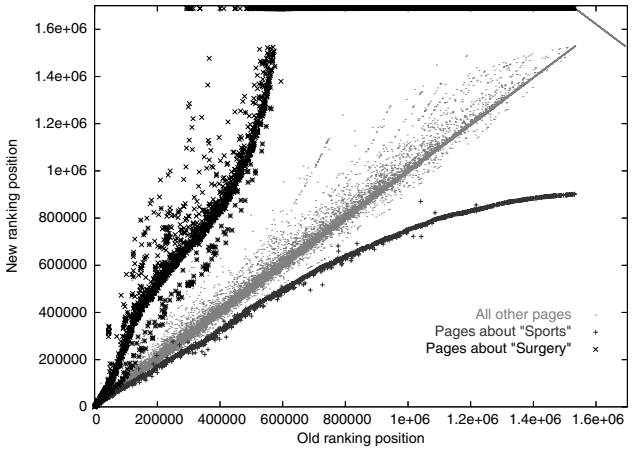

Figure 6. The results achieved on the rank constraint problem.

ing the neural network with different initial conditions, and then to chose the best performing one.

4.3. Rank constraints

We have used explicit targets so far, i.e. for each page n_i a desired rank t_i was given. Even if these experiments were useful to study the properties of the proposed model, a user may not be able to define an exact target for a given page. On the other hand, a more convenient approach consists of allowing the users to give examples as constraints between the ranks of pages. For instance, a set of examples can be collected by asking the user to sort a set of pages according to the user's own preferences.

In this experiments, we assume that the examples are constraints $o_{n_1} > o_{u_1}, \ldots, o_{n_c} > o_{u_c}$, where o_n is the GNN output for node n, and $o_{n_i} > o_{u_i}$ states that the rank of page n_i must be higher than the rank of page u_i.

Note that in general there may be an infinite number of ranking functions that satisfy a set of constraints, so we add the further rule: the produced rank should be close to Google's PageRank. The idea supporting this approach is that we should satisfy user's constraints, but we should also avoid producing a rank that is completely different from PageRank. In practice, the following error function will implement this goal:

$$J_{\boldsymbol{w}} = \sum_{n \notin S} (o_n - pr_n)^2 + \alpha \sum_{i=1}^{c} L(o_{n_i} - o_{u_i}) \,,$$

where $S = \{n_1, \ldots, n_c, u_1, \ldots, u_c\}$ is the set of constrained pages, L is a penalty function such that $L(y) = y^2$ if $y < 0$ and $L(y) = 0$, otherwise, and α is a parameter balancing the importance of the constraints w.r.t. the importance of keeping the pages close to PageRank.

The experiment used 10 constraints which involved 10 pages on "Surgery" and 10 pages on "Sport". Each constraint was in the form of $\boldsymbol{o}_{n_i} \geq \boldsymbol{o}_{u_i}$, where n_i was a page on "Sport" and u_i a page about "Surgery". Thus, the constraints forced the GNN to produce higher ranks for pages about "Sport" than pages about "Surgery". The pages of the constraints were randomly selected. A further 3980 random selected pages completed the training graph $\bar{\boldsymbol{G}}$.

Figure 6 shows that most of the pages about "Sport" are above the diagonal, whereas most of the pages about "Surgery" are below the diagonal. Moreover, pages which do not address either "Sport" or "Surgery" are mainly found close to the diagonal. Thus, the experiments achieved the expected result. The experimentation also proves that the proposed model is able to generalize the lesson learned on a small set of examples to a large dataset. In fact, even if the training set contained only 20 constraints and dealt with few pages, the GNN has learned that documents about "Sport" are more important than documents about "Surgery". Figure 6 clearly demonstrates that the GNN had applied this learned rule also to pages which do not belong to the training set.

It is also interesting to note that there are many pages with positions close to the horizontal axis for the topic "Surgery". This is perfectly normal as most of these pages have a very small number of incoming links. Thus, their ranks are mainly determined by the constant b_n of Eq. (7). Moreover, since b_n depends only on the page content, the ranks of all these pages are about the same. In practice, the GNN assigned to these pages the smallest ranks of the whole dataset so that they occupy the lowest positions of the rank produced by GNN.

5. Conclusions

It was shown that the GNN model can be adapted to the task of computing customized web page rank values. GNNs can learn a ranking function by examples and they are capable of generalizing over unseen data. The proposed model extends the previous approaches both for its learning capability and for the ability to produce very general ranking functions. On the other hand, GNNs maintain useful properties of the previous approaches such as the fast convergence of the iterative mechanism used to compute the rank. Thus, even if learning may be a slower computational process than the solution of linear equation like in PageRank's case, the computation of the rank after the training phase is a fast process that can be carried also on very large Web graphs.

Moreover, the experiments have shown promising results confirming an outstanding generalization capability of the proposed model.

Future directions of research include the experimentation of GNNs on a considerable larger dataset which is currently being prepared by [5]. Moreover, it will be useful to analyze the capabilities of GNNs on handling more complex learning tasks such as: contradicting constraints – some constraints cannot be fulfilled without violating other constraints; or weighted constraint – a weight is associated to each constraint measuring its importance or cost.

Finally, an important matter of future research is the study of the approaches that can be exploited to collect from users examples used in the training set.

Acknowledgement

The 2nd and 4th author wish to acknowledge financial support from the Australian Research Council in the form of a Discovery Project grant. In addition, the 1st, 2nd and 6th author were supported by an Australian Research Council Linkage International Grant.

References

[1] L. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In M. Caudill and C. Butler, editors, *IEEE International Conference on Neural Networks*, volume 2, pages 609–618, San Diego, 1987.

[2] M. Bianchini, M. Gori, and F. Scarselli. Inside pagerank. *ACM Trasanctions on Internet Technology*, 2005.

[3] S. Brin and L. Page. The anatomy of a large–scale hypertextual Web search engine. In *Proceedings of the 7th World Wide Web Conference*, Apr. 1998.

[4] H. Chang, D. Cohn, and M. A.K. Learning to create customized authority lists. In *Proceedings of the 17th International Conference on Machine Learning*, pages 127–134. Morgan Kaufmann, 2000.

[5] W. Chiang, M. Hagenbuchner, and A. Tsoi. The wt10g dataset and the evolution of the web. In *14th International World Wide Web conference*, Alternate track papers and posters, Chiba city, Japan, May 2005 (to appear).

[6] M. Diligenti, M. Gori, and M. Maggini. A unified probabilistic framework for web page scoring systems. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):4–16, January 2004.

[7] M. Diligenti, M. Gori, and M. Maggini. Learning web page scores by error back-propagation. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2005.

[8] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9(5):768–786, September 1998.

[9] M. Gori, M. Hagenbuchner, F. Scarselli, and A. C. Tsoi. Graphical based learning environment for pattern recognition. In *Proceeding of Syntatic and Structural Pattern Recognition*, August 2004. (Invited Paper).

[10] M. Gori, M. Maggini, E. Martinelli, and F. Scarselli. Learning user profiles in NAUTILUS. In *International Conference on Adaptive Hypermedia and Adaptive Web–based Systems*, Trento (Italy), August 2000.

[11] M. Hagenbuchner, A. Sperduti, and A. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 2003.

[12] T. H. Haveliwala. Topic sensitive pagerank. In *Proceedings of the 11th World Wide Web Conference (WWW11)*, 2002. Available on the Internet at http://dbpubs.stanford.edu:8090/pub/2002-6.

[13] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th World Wide Web Conference*, 20–24May 2003.

[14] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[15] P. Langley, P. Iba, and P. Thompson. An analysis of bayesian classifiers. In *The Tenth National Conference on Artificial Intelligence*, 1992.

[16] F. Pineda. Generalization of back–propagation to recurrent neural networks. *Physical Review Letters*, 59:2229–2232, 1987.

[17] M. Richardson and P. Domingos. The intellingent surfer: probabilistic combination of link and content information in pagerank. In *Advances in Neural Information Processing Systems, 14*, Cambridge, MA, 2002. MIT Press.

[18] F. Scarselli, A. C. Tsoi, M. Gori, and M. Hagenbuchner. A new neural network model for graph processing. Technical Report DII 01/05, Dept. of Information Engineering, University of Siena, 2005.

[19] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8:429–459, 1997.

[20] A. C. Tsoi, G. Morini, F. Scarselli, Hagenbuchner, and M. Maggini. Adaptive ranking of web pages. In *Proceedings of the 12th WWW Conference*, Budapest, Hungary, May 2003.