Improved Classification via Connectivity Information

Andrei Z. Broder[*] Robert Krauthgamer[†] Michael Mitzenmacher[‡]

Abstract

The motivation for our work is the observation that Web pages on a particular topic are often linked to other pages on the same topic. We model and analyze the problem of how to improve the classification of Web pages (that is, determining the topic of the page) by using link information. In our setting, an initial classifier examines the text of a Web page and assigns to it some classification, possibly mistaken. We investigate how to reduce the error probability using the observation above, thus building an improved classifier. We present a theoretical framework for this problem based on a random graph model and suggest two linear time algorithms, based on similar methods that have been proven effective in the setting of error-correcting codes. We provide simulation results to verify our analysis and to compare the performance of our suggested algorithms.

1 Introduction

Imagine a Web page classifier that examines the contents of each page and infers some classification for it. Such a classification could be a topical category (e.g. sport, society, etc.), a language (e.g. French, English, etc.), or a discourse style (e.g. scientific paper, news, etc.). A natural way to improve the page content classifier is to use link information, in so far as pages sharing a common classification are more likely to be linked than pages with different classifications. Here "linked" is taken in a broad sense: it can mean that there is a hyper-link from one page to another, or the fact that there is a third page that hyper-links to both pages (co-citation), or the fact that many users tend to view these pages in close sequence.

The question is: How can we use the link information along with the initial classifier to obtain an improved classifier? Other natural questions arise from this one: what are the limits on the improvement we can obtain? Can we trade off speed and accuracy in a useful way? For the Web context, Chakrabarti, Dom, and Indyk [5] showed experimentally that this approach is very successful. Their improved classifier uses a first order Markov Random Field as a model for the problem based on experimentally determined probabilities for the various types of links.

Clearly if the initial classification is correct the links are irrelevant. On the other hand if all links are guaranteed to go only between members of the same class and there are enough of them, then it suffices that the initial classification is slightly correlated with reality, since in this case each sufficiently large connected component can be identified by plurality vote. What about the more common case, where the initial classification can be wrong and links may span classes?

In this paper, we attempt to provide a theoretical framework for this problem. In our theoretical framework, we suggest some simple models and study the performance of various algorithms in these models. In particular, we focus on describing the case where there are just two possible classifications, a case which already provides a rich space to explore. We then describe how to extend our analysis to the case of multiple colors.

Our framework is motivated in part by recent developments in error-correcting codes. An error-correcting code can naturally be thought of as a classification problem: each bit should either be a 0 or a 1. A decoder has an initial classification of a bit, based on the value it received. The bits are "linked" via constraints that define the code; indeed, often these constraints can be represented naturally in a graph. The goal is to correctly classify all the bits, within some reasonable time constraint. In the realm of classifying Web pages, our goals will be much more modest. We merely hope that an improved classifier using link information will have substantially higher accuracy than a classifier that does not use link information.

The fact that links provide useful information on

[*]Compaq Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, USA. E-mail: broder@pa.dec.com.

[†]Weizmann Institute of Science, Rehovot 76100, Israel. Parts of this work were done while visiting Compaq Systems Research Center. E-mail: robi@wisdom.weizmann.ac.il

[‡]Harvard University, 33 Oxford St., Cambridge, MA 02138. Parts of this work were done while at Compaq Systems Research Center. E-mail: michaelm@eecs.harvard.edu.

the Web has most clearly been demonstrated by the HITS technique of Kleinberg (and various extensions thereof) for finding "hub" and "authority" web sites [2, 8, 4], and the PageRank algorithm used by Google for ranking web sites [3]. The specific idea of using link information to improve a classifier was suggested and tested through experiments by Chakrabarti, Dom, and Indyk in [5]. They showed that an approach based on iteratively relabeling pages using link information was successful using data from both Web pages and the U.S. Patent database.

The idea of iterative relabeling is also behind recent advances in decoding low density parity check codes [7, 12, 11, 13, 14, 16, 17] and the similarly motivated Turbo codes introduced by Berrou, Glavieux, and Thitimajshima [1, 15]. Hence it is natural to ask whether the approaches to analysis that have proven successful in this area can be applied to gain insight into the classification problem.

In this paper, we refer to the possible classifications as colors. We assume that an input graph $G = (V, E)$ with n nodes is drawn from a certain probability distribution known to the algorithm. In this distribution, each node $v \in V$ has a color $\kappa(v)$, and the existence of an edge between a pair of nodes is somehow correlated with their colors. An initial classifier proceeds independently for every node and assigns it a classification $\lambda(v)$. The probability distribution that the initial classifier is correct is also known. (In general, this probability is conditional on the real color of the node.) We begin with only the coloring information given by the classifier, as the input graph is given without its real coloring.

We want to build an improved classifier λ' so that $\lambda'(v)$ takes into account the link information. How much of the original coloring can we hope to restore under various assumptions on the distribution of G? In the interests of keeping the notation manageable and emphasizing the ideas we focus on a dramatic simplification of the problem, where there are only two possible classifications. We also describe a model with more than two colors.

A goal of introducing these models is to explore the accuracy of simple, efficient (linear time) algorithms, both through analysis and simulations. The algorithms we focus on have a simple form: each node holds some sort of state, representing its "belief" in its most likely color. Messages are passed between neighbors, causing the nodes to update their state for the next round; and after some number of rounds, the process stops, and nodes determine their final classification. Using our models, we demonstrate that dramatic improvements in classification by using link information is possible, corroborating the results of [5]. We also initiate an exploration of the tradeoffs between speed and accuracy, using the idea that algorithms may maintain, pass, and act on different state information.

We note that just before submitting the final version of this paper we became aware of recent work by Kleinberg and Tardos, who study a variation of this problem [9]. In their work, they consider the *cost* of a coloring. There is a cost associated with assigning a node a color, and a cost associated with each edge where the nodes for that edge are assigned different colors. These costs would correspond naturally to probabilities in our model, in a manner Kleinberg and Tardos describe. They use linear programming and randomized rounding to obtain approximation algorithms for the problem of finding the coloring with minimum cost.

Our work clearly differs from the above, in that we study a specific random graph model and the behavior of simpler algorithms in that model. In particular, our probabilistic approach does not yield (as far as we know) an approximation guarantee, and it must be suitably generalized to handle other graph models. Because we focus on very fast, simple algorithms, however, we believe the algorithms we consider may prove more useful for dealing with large data sets such as the Web.

2 The two color model

2.1 Informal description Initially we consider the case where all nodes take on only one of two colors, red or green, denoted r and g. In general, the input graph is directed (such as in the case where the edges represent links between Web pages). However, for the purposes of our algorithm, the direction of an edge is material only to the extent that it influences the conditional probability distribution of the color of its endpoints. Hence we will say that x and y are neighbors if there is an edge from x to y, or vice versa.

Let $\kappa(v)$ be the color of node v, and let $\lambda(v)$ be its classification from the initial classifier. Abusing notation, we let $\kappa(G)$ represent the coloring of the entire graph, and $\lambda(G)$ represent the results from the classifier.

An optimal classifier can be constructed for a graph with n vertices by computing for each of the 2^n possible colorings $\kappa(G)$

$$\mathbf{Pr}\left(\kappa(G) \mid G, \lambda(G)\right).$$

Once these terms are computed one can determine the probability that a particular node is, say, red, simply via

$$\mathbf{Pr}(\kappa(v) = r) = \sum_{\kappa(G) \text{ s.t. } \kappa(v)=r} \mathbf{Pr}\left(\kappa(G) \mid G, \lambda(G)\right).$$

Of course this is impractical for almost all graphs.

Hence we will attempt to find ways to achieve good accuracy with much less computation.

2.2 The tree model We describe an intermediary model, *the tree model*, where the computation can be carried much faster. The tree model forms the basis for our more general random graph model. The tree model is based on a random graph built recursively as follows. We shall call a tree "red" (resp. green) if its root is colored red (resp. green). The tree $T_0(v)$ consists of just one node, v. A red tree $T_i(v)$ consists of a red root node v that has $X(v)$ children, each one being a random T_{i-1} tree. The number of children $X(v)$ is an independent random variable Poisson distributed with parameter μ. The color of each child is chosen independently at random: a child is red with probability β_{rr} and green with probability $\beta_{rg} = 1 - \beta_{rr}$. A green $T_i(v)$ is built in an identical manner, with corresponding parameters (namely β_{gr} and β_{gg}).

The input to our algorithm is a random T_i tree, which is red with probability α_r and green with probability $\alpha_g = 1 - \alpha_r$. Every node v in the tree receives an initial classification, $\lambda(v)$. Let $\kappa(v)$ be the color of node v, and let $\gamma_{ab} = \mathbf{Pr}(\lambda(v) = b \mid \kappa(v) = a)$. Here γ_{ab} describe the accuracy of the classifier.

Consider a node x that has d children $y_1, \ldots, y_d$. Let $\lambda(T(x))$ be the entire set of initial classifications made at all descendants of x including x. Our next goal is to compute the probability of a particular initial classification $\lambda(T(x))$ conditional on the color of x. Decomposing according to all possible colorings for $y_1, \ldots, y_d$ we obtain that

$$\begin{aligned}
&\mathbf{Pr}\big(\lambda(T(x)) \mid \kappa(x)\big) \\
&= \sum_{\kappa(y_1),\ldots,\kappa(y_d)} \prod_{j=1}^{d} \mathbf{Pr}\big(\lambda(T(y_j)) \mid \kappa(y_j)\big)\beta_{\kappa(x),\kappa(y_j)} \\
&= \prod_{j=1}^{d}\Big(\mathbf{Pr}\big(\lambda(T(y_j)) \mid \kappa(y_j) = r\big)\beta_{\kappa(x),r} \\
&\qquad + \mathbf{Pr}\big(\lambda(T(y_j)) \mid \kappa(y_j) = g\big)\beta_{\kappa(x),g}\Big)
\end{aligned}$$

Furthermore, if x is a leaf, then

$$\mathbf{Pr}\big(\lambda(x) \mid \kappa(x)\big) = \gamma_{\kappa(x)\lambda(x)}.$$

Hence, for the root x_0 we can recursively compute $\mathbf{Pr}\big(\lambda(T(x_0)) \mid \kappa(x_0)\big)$, in time

$$O\Big(\sum_{v \in V} d(v)\Big) = O(m), \tag{2.1}$$

where $d(v)$ is the number of children of v, and m is the number of edges in the tree. This is of course much smaller than 2^n.

Finally,

$$\begin{aligned}
&\mathbf{Pr}\big(\kappa(x_0) = r \mid \lambda(T(x_0))\big) \\
&\quad = \mathbf{Pr}\big(\lambda(T(x_0)) \mid \kappa(x_0) = r\big)\, \alpha_r \\
&\qquad \Big/ \Big(\mathbf{Pr}\big(\lambda(T(x_0)) \mid \kappa(x_0) = r\big)\, \alpha_r \\
&\qquad\qquad + \mathbf{Pr}\big(\lambda(T(x_0)) \mid \kappa(x_0) = g\big)\, \alpha_g\Big)
\end{aligned}$$

and therefore we set

$$\lambda'(x_0) \leftarrow r$$

iff

$$\mathbf{Pr}\big(\lambda(T(x_0)) \mid \kappa(x_0) = r\big)\, \alpha_r > \mathbf{Pr}\big(\lambda(T(x_0)) \mid \kappa(x_0) = g\big)\, \alpha_g.$$

This is the optimum classifier for the root.

2.3 The tree model and belief propagation We now discuss belief propagation, which generally refers to an algorithm whereby messages concerning conditional probabilities are passed around a graph, in the specific context of the classification problem. Consider a random graph on n nodes, where each node is independently colored red with probability α_r and green with probability α_g. We assume that each edge slot between a pair of red nodes is filled independently with probability δ_{rr}, and similarly there exist parameters δ_{gg} and δ_{rg}. We also assume that these values are chosen so that the average degree of a node is constant. For convenience, we consider here the symmetric case where $\alpha_r = \alpha_g$, $\delta_{gg} = \delta_{rr}$ and $\delta_{rg} = \delta_{gr}$. This implies that the degree distribution for every node is the same.

The belief propagation algorithm proceeds over a number of rounds. For expository purposes, it is useful to assume that in each round each node makes an independent computation and sends a message to every one of its neighbors. The message passed from x to y reflects the current belief of node x regarding its color (the conditional probability that x has a particular color), based on its initial classification and all the information obtained so far from neighbors *other than* y. More precisely, consider a node x with neighbors $y, z_1, \ldots, z_d$. (See Figure 1.) Initially (in round 0), with no information from its neighbors, x sends to y the probability that it is, say, red, conditional on its initial classification. Once x receives information from $z_1, \ldots, z_d$, it calculates its new conditional probability of being red based on all the information received so far, and sends that on to y. (Notice that information obtained from y is ignored when calculating information

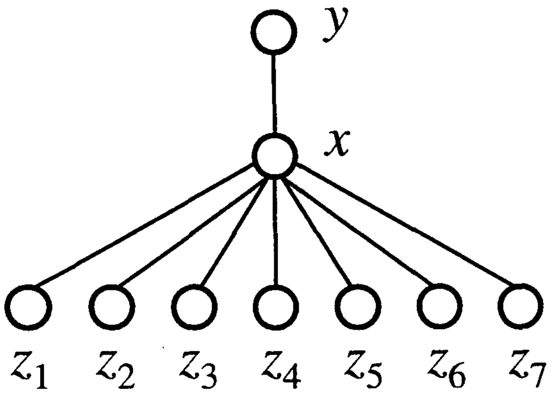

Figure 1: The message from x to y is based on information passed "up the tree" from children $z_1, \ldots, z_d$ in the previous round.

to send to y; the intuitive explanation for this is to prevent positive feedback.) This process continues for as many rounds as desired, where at each round, each node x performs such a computation for each of its neighbors, and sends them the (possibly different) results. Upon termination, each node computes its probability of being red given *all* the information received so far.

Note that the belief propagation algorithm essentially implements the tree model in parallel for all nodes in the graph, using only local message passing between neighboring nodes in the original graph. For sufficiently large graphs, the approximation of our random graph model with the assumption of a Poisson number of children is sufficiently accurate; the values δ_{gg},δ_{rg},δ_{rr}, and δ_{gr} yield corresponding values β_{rr}, and so on. The message passed from x to y in round k is obtained by computing $\lambda(T(x))$, where in the tree $T(x)$ the children of x excludes the neighbor y, and the tree $T(x)$ has depth k. Assuming that the neighborhood around a node x is cycle-free for a diameter k, after k rounds the node x will be able to compute the correct conditional probability of being red based on its neighborhood.

Hence, if a sufficiently large neighborhood of every node is cycle free, we can efficiently have each node make an optimal decision about its own color based on the information from a limited neighborhood of the node. Of course, having each node separately making its own optimal decision does not necessarily yield the globally optimal solution – the equivalent coloring might have probability 0. However, having each node choose its most likely color maximizes the expected number of correctly classified nodes, which is the important practical issue. Note also that it is usually the case that the most valuable information is located in the close neighborhood of a node, and so computations based on a limited neighborhood should give a good approximation to those based on the full graph information.

Of course cycles in the graph G bring into question whether belief propagation will be effective, since the assumption that the graph looks like a tree around the node will no longer be true. For a sufficiently large random graph, we expect few small cycles, and hence the problem may be limited to a small number of nodes. In practice we may also run the algorithm even if there are cycles. The nodes perform local computation at each round, simply pretending that the information they have obtained is based on a tree neighborhood. There is hope that the skew caused by cycles will be a lower order effect even over a large number of rounds. This technique has proven useful in other areas, such as coding theory [7, 12, 11, 13, 14, 16, 17].

Note that this model can be generalized to the cases where the parameters are not symmetric, or the degree at a node is not Poisson; these variations may simply make the conditional probability calculations at each step more complex. Assuming that we perform the belief propagation for a constant number of rounds, the total time taken by the algorithm will be proportional to the number of edges in the graph. Indeed, it is easy to see that all the computations of a node in a single round can be done in a number of operations which is proportional to its degree.

2.4 A simpler message passing algorithm

As a first step towards analyzing the effectiveness of belief propagation, we consider a simpler algorithm that passes less information. The price we pay for the simplicity is less accuracy. In the case of two colors we can easily analyze the performance of this algorithm in the tree model. Another reason for studying this algorithm is that it is faster than belief propagation, since the messages passed and the computations made throughout are simpler. Even though both belief propagation and this simplification require only linear time, because we are considering extremely large graphs, this faster approach may be worthwhile.

Again, we think of each node x passing messages to each of its neighbors in each round, but here the message will simply be colors instead of conditional probabilities. Again we assume the random graph model described above, and think of the node x passing a message to a node y as being the root of a tree with d children $z_1, \ldots, z_d$. In each round, the node x computes and sends to y a believed classification $\lambda'(x, y)$, such that $\lambda'(x, y)$ is only a function of the values $\lambda'(z_1, x), \ldots, \lambda'(z_d, x)$ from the previous round and $\lambda(x)$.

The algorithm is based on the following idea: $\lambda'(x,y) \leftarrow \lambda(x)$, unless sufficiently many of children of x suggest that $\lambda(x)$ is wrong, i.e. $\lambda(x) \neq \kappa(x)$. The proper threshold for how many children must suggest this (by having $\lambda'(z_i,x) \neq \lambda(x)$) can be determined by a probability calculation, that we now embark on. To simplify exposition we shall say that a child z_i "sends" its classification $\lambda'(z_i,x)$ to its parent x or "votes" on $\lambda'(x,y)$.

As before, for convenience we shall focus on the symmetric case, where each node is equally likely to be colored red or green, the probability of the initial classification being wrong is the same for both red and green nodes, and monochromatic edges of each color are equally likely (i.e., $\alpha_R = \alpha_G = 1/2$, $\beta_{rr} = \beta_{gg} = \beta$, and $\gamma_{rg} = \gamma_{gr} = \gamma$). We describe later the necessary modifications when there is no such symmetry. As we shall see, in the symmetric case, a proper threshold can be easily precomputed and has an interesting form.

Define p_i to be the probability that $\lambda'(x,y)$ is wrong in the ith round. Observe that due to the symmetry of our model, this probability is the same for all edges (x,y). Clearly in the symmetric case p_0 is just the probability that the initial classifier is incorrect, or $p_0 = \gamma$. Now we compute p_{i+1}. Assume that x has d children $y_1,\ldots,y_d$. As explained above, x will set $\lambda'(x,y) \leftarrow \lambda(x)$ unless at least a certain threshold, $b(i,d)$, of its children have sent it $\overline{\lambda(x)}$ as their classification.

First, observe that the probability that a child of x sends $\overline{\kappa(x)}$ as its classification $\lambda'(x,y)$ is exactly

$$q_i = \beta p_i + (1-\beta)(1-p_i).$$

(With probability βp_i the child has the same color as x but sends a mistaken classification, and with probability $(1-\beta)(1-p_i)$ the child has the opposite color and sends it correctly to x.) Note for future reference that, since $\beta > 1/2$ and $\gamma < 1/2$, we have $q_i < 1/2$.

Second, observe that x will send $\overline{\kappa(x)}$ iff either $\lambda(x) = \kappa(x)$ but at least $b(i,d)$ of its children sent it $\overline{\kappa(x)}$, or $\lambda(x) = \overline{\kappa(x)}$ but less than $b(i,d)$ of its children sent it $\kappa(x)$.

Let η_d be the probability that x has d children. The two observations above lead to the conclusion that

$$\begin{aligned} p_{i+1} &= \sum_{d=0}^{\infty} \eta_d \cdot \left[(1-\gamma)\sum_{j=b_{i,d}}^{d}\binom{d}{j} q_i^j (1-q_i)^{d-j} + \gamma \sum_{j=0}^{b_{i,d}-1}\binom{d}{j}(1-q_i)^j q_i^{d-j}\right] \\ &= \sum_{d=0}^{\infty} \eta_d \cdot \left[(1-\gamma)\sum_{j=b_{i,d}}^{d}\binom{d}{j} q_i^j (1-q_i)^{d-j} + \gamma\left(1 - \sum_{j=b_{i,d}}^{d}\binom{d}{j}(1-q_i)^j q_i^{d-j}\right)\right] \\ &= \sum_{d=0}^{\infty} \eta_d \cdot \left[\gamma - \sum_{j=b_{i,d}}^{d}\binom{d}{j}\left(\gamma(1-q_i)^j q_i^{d-j} - (1-\gamma) q_i^j (1-q_i)^{d-j}\right)\right] \end{aligned}$$

Hence the optimum value for $b_{i,d}$ should be such that

$$\sum_{j=b_{i,d}}^{d}\binom{d}{j}\left(\gamma(1-q_i)^j q_i^{d-j} - (1-\gamma) q_i^j (1-q_i)^{d-j}\right)$$

is maximized. Notice that the term corresponding to summing index j is positive iff

$$\text{(2.2)} \qquad \left(\frac{1-q_i}{q_i}\right)^{2j-d} > \frac{1-\gamma}{\gamma}.$$

Since, as we observed previously, $q_i < 1/2$, the left side of (2.2) is strictly increasing as j increases and therefore the optimum $b_{i,d}$ should be the smallest j for which (2.2) holds, that is, we set $b_{i,d}$ such that

$$\text{(2.3)} \qquad 2b_{i,d} - d > \ln\left(\frac{1-\gamma}{\gamma}\right) \Big/ \ln\left(\frac{1-q_i}{q_i}\right),$$

which ensures that $p_{i+1} \leq p_i$. In particular, our classifier is at least as good as the original classifier at each node.

Note that the right side of the inequality (2.3) does not depend on d. Denote it by Δ_i. It is interesting to note that the left hand side $2b_{i,d}-d$ equals $b_{i,d}-(d-b_{i,d})$. Hence, equation (2.3) states that the x should pass the initial classification unless the majority vote disagrees with it and the majority vote beats the minority vote by more than Δ_i votes. Our algorithm can therefore be stated as

$$\boxed{\text{Set } \lambda'(x,y) \leftarrow \lambda(x) \text{ unless } \overline{\lambda(x)} \text{ wins the children vote by more than } \Delta_i \text{ votes.}}$$

That our recursion for p_i is asymptotically correct as the size of the random graph G grows can be proven using a martingale argument, entirely similar to the proofs given in [14, 16] for the similar coding scenario. Consider the process up to some fixed constant number of rounds, j. There are two problems to be handled. First, some nodes might not have appropriate tree neighborhoods; it can be shown that the number of such nodes is small (though dependent on j). Second, it is not clear that the number of edges passing correct messages must be concentrated around its expectation. However, over j rounds the message passed along an edge is correlated with only a constant number of other edges, and so a martingale argument yields a concentration result.

The same method of analysis can be applied to the asymmetric case, although the equations are necessarily more complex. In particular, the recursive calculations require keeping track of the distinct probabilities that a green node and a red node pass the wrong color along an edge in round i. Similarly, the number of neighbors necessary to change the color a node x passes will depend on the initial classification $\lambda(x)$.

2.5 Extending the simple scheme Using the techniques described by Richardson and Urbanke to study belief propagation for error-correcting codes in [16], it appears that we can analyze belief propagation in the tree model in a similar manner to how we have analyzed the simple message-passing scheme. Instead of calculating the probability that the color red or green is passed in a message each round, we recursively calculate for each round the distribution of the value of the conditional probability passed in the appropriate message, based on the distribution of the messages obtained from the previous round. Of course computing the recursive behavior of distributions on $[0, 1]$ is much more complex than binary 0/1 random variables.

An interesting question is to come up with suitable intermediary schemes, that pass more detailed information than just the color but less information than the conditional probability. For example, we may consider sending a value from the set $\{-3, -2, -1, 0, 1, 2, 3\}$, with the interpretation that larger positive values suggest a stronger certainty of being green, and larger negative values suggest a stronger certainty of being red. As the number of possible values grows towards infinity, this method can increasingly approximate the behavior of belief propagation. The work of Richardson and Urbanke [16] suggests that an appropriate intermediary scheme may allow significant constant-factor speedups over belief propagation with minimal impact on performance.

3 Simulations

In this section, we compare the accuracy of belief propagation and the simpler, one-bit message passing algorithms using simulations of the symmetric case. We use simulations because of the difficulty of analyzing belief propagation and so that we may see the behavior of our algorithms on graphs of reasonable size. We focus on accuracy and not on speed, since the speed depends strongly on system details such as the size of the cache and the relative speed of integer and floating point computations. Recall that the simpler algorithm involves no floating point arithmetic, except for the precomputation of the thresholds Δ_i. In our experience, however, for small graphs (10,000 nodes), belief propagation takes almost twice as long. We expect that for larger graphs the smaller memory requirements of the simpler algorithm would yield even greater speedups.

Surprisingly, we find that the difference in accuracy between the two algorithms is usually small. Hence, in this case, we generally do not lose much information by collapsing each belief into a single bit.

We also compare the simulation results with our analysis of the simple message passing algorithm from Section 2.4. Our goal here is to check that the analysis, which holds only asymptotically, is accurate for graphs of a reasonable size. Moreover, we would hope to gain some understanding in situations where the analysis fails. We scrutinize the few places where there is a notable difference between the analysis and simulations, and give evidence these differences appears to be caused by small cycles in the graphs. We conclude that the analysis can indeed provide an excellent prediction for our model, if this pitfall is kept in mind.

3.1 Comparison between simulations of the two algorithms We measure the *final error* of each of the algorithms in terms of the fraction of edges passing the wrong value. For belief propagation, where probabilities (and not colors) are being passed in each round, during a round we associate with each edge (in each direction) the color with the larger probability. For the simpler message passing algorithm, during a round each edge (in each direction) is naturally associated with the color being passed.

It is worth noting that in our simulations, in some cases accuracy improves over a number of rounds, but then begins to deteriorate. This deterioration appears to be traceable to the cycles in the graph; if we run to the point where cycles start to impact results, performance can actually degrade. The final error results therefore represent the best performance obtained during the course of the algorithm. Note that the algorithm itself cannot tell what the best point to

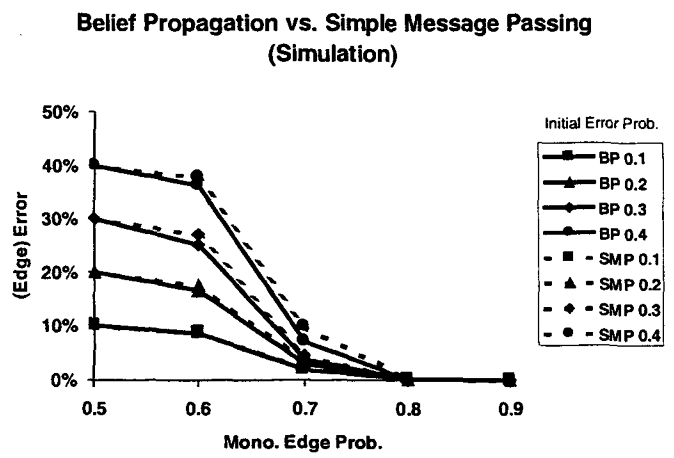

Figure 2: Simulation results of simple message passing and belief propagation.

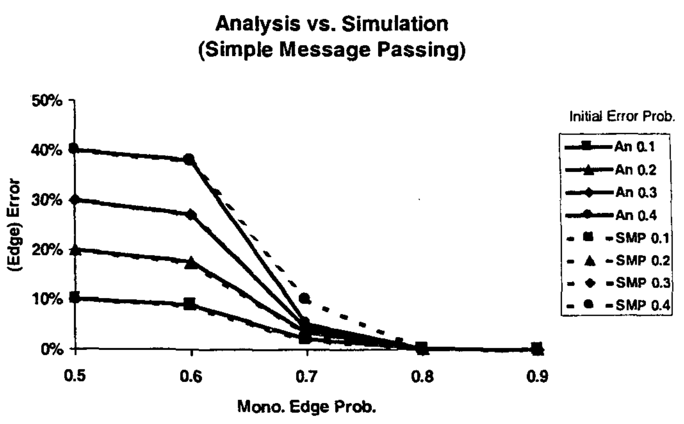

Figure 3: Analysis vs. simulation results for simple message passing

stop is, as this requires knowledge of the initial coloring. In practice, however, we believe the best stopping point can be determined accurately via simulation and knowledge of the graph structure; hence this comparison appears the most fair possible.

Also, we emphasize that our measurements present the final error in terms of the number of edges passing the wrong color, which is almost (but not exactly) the probability that a node takes on the wrong color using an improved classifier. We adopt this measure in order to be consistent with the analysis offered previously. In the interest of space, we leave data regarding the fraction of node errors for the final version.

The final error is a function of two parameters describing the quality of the input. The *monochromatic edge probability* is the probability that an edge in the graph is monochromatic. In our symmetric case, this is simply β. This, along with the average degree of a node, completely specifies the parameters defining the edges of the graph. For brevity, we consider here only the case where the average degree is 20. The *initial error* is the error rate incurred by the inaccuracy of the initial classification. In our symmetric case, this is simply γ. Our simulations are from graphs with 10,000 nodes. Each data point represents the average of 25 random inputs. Generally, unless otherwise specified, the deviations from the average are small, so we do not address them further.

The results of simulations comparing simple message passing and belief propagation are presented in Figure 2. Observe that, in many cases, the accuracy of the simple message passing algorithm is almost as good as that of belief propagation. In particular, when belief propagation does very well, reducing the final error to a very small number, then often the simpler algorithm suffices. In these cases, the information from the edges is so valuable that even modest use of it vastly improves performance. Not surprisingly, when this information is less valuable, belief propagation exploits the link information better. It is worth noting that the final error behaves monotonically in the two input parameters, exactly as one would expect. These results are generally representative of our more extensive simulations, in which we varied the average degree, the number of nodes, etc.

3.2 Simulations versus analysis The asymptotic analysis of the simple message passing algorithm from Section 2.4 can be used to predict the final error for small graphs by plugging the input parameters (average degree, initial error, etc.), into the formulas of Section 2.4. The analysis prediction for the final error is found by computing successive values of p_i until the values converge. The sequence of values p_i tends to converge numerically after at most ten rounds, and often it converges much more quickly.

The simulation results of the simple message passing algorithm are compared to the analysis prediction in Figure 3. In most cases the two yield similar results. However, in some cases, such as when the monochromatic edge probability is around 0.7 and the initial error is relatively high, there is a notable difference between analysis and simulation. We explain this discrepancy.

Consider the case where the monochromatic edge probability β is 0.7 and the initial error rate γ is 45%. Recall that p_i is the probability that an edge sends an erroneous message in the ith round according to the analysis. We show in Figure 4 how p_i changes with i for several values of β around 0.7, and also mark between

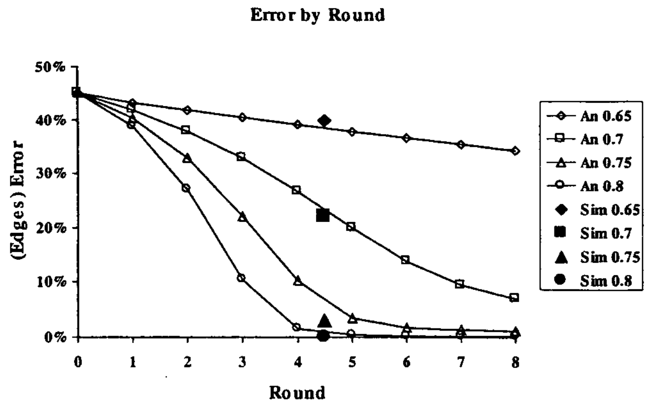

Figure 4: Analysis by round vs. simulation results for message passing.

rounds four and five the result from simulations. At $\beta = 0.7$, there is a sharp drop in p_i between rounds 4 and 6, from 27% to 14%. However, with 10,000 nodes and average degree 20, there are typically many cycles of length five and six. Since the analysis assumes that small cycles do not exist (or at least do not skew results significantly), it appears clear that the failure of the analysis is due to the failure of this assumption.

This idea is corroborated by the fact that when probability falls quickly within a few rounds, or falls very slowly after a few rounds, the analysis provides much more accurate predictions, as can be seen in Figure 4. Indeed, in all cases the results from the simulations are near the results given from the analysis from the fifth round. Further, we have found that by performing simulations on graphs with more nodes but the same parameter settings (e.g. average degree), the prediction from the analysis becomes better as the number of nodes increases. The improvement appears to correspond to an increase in the size of a typical small cycle in the graph. We note that also belief propagation does not elude this problem, as its final error (when $\beta = 0.7$, $\gamma = 0.45$) is only slightly better, and is much worse than the analysis would suggest. We conclude that in order to use the analysis to predict the behavior of the simple message passing algorithm, one must take into account the typical length of short cycles in the graph, and use the analysis prediction within a limited number of rounds.

4 Multiple Colors

For a richer model that more accurately represents the problem of classifying Web pages, it is necessary to consider cases with more than two colors. In this case, the number of possibilities for modeling the correlations between different types of pages is quite large. Indeed, in the work by Chakrabarti, Dom, and Indyk, the model for how pages of different categories tend to link was developed by using a subset of the data as a training set [5]. Rather than focus on developing an accurate model (a subject well worth further study), we examine a simple model that captures high level behavior we expect from objects such as Web pages.

4.1 A coloring model In developing a classifier for Web pages, we would expect certain errors to be more likely than others. For example, an article about a soccer match might understandably be classified under entertainment rather than sports, but it is unlikely that such an article would be mistakenly labeled as politics. We therefore adopt a color model that reflects the property that certain categories are more likely to be confused than others.

In a natural random graph model with colors $c_1, c_2, \ldots, c_k$, each node would be colored c_i with some probability α_{c_i}. The edge slots between colors c_i and c_j (where here i could equal j) would each contain an edge independently with probability $\delta_{c_i c_j}$. The initial classifier would mistakenly give a node of color c_i the color c_j with probability $\gamma_{c_i c_j}$.

We suggest a simple, symmetric model of this type as a first subject for analysis. Let there be $k = 2^t$ possible colors, where t should be thought of as a small integer (say 3). The *distance* $d(i,j)$ between colors c_i and c_j $(i \neq j)$ is $t - d$, where d is the number of digits from the left that the binary representations of i and j agree on. For example, the distance between color 000 and 001 is 1, while the distance between 111 and 011 is 3. It is clear that this distance measure is symmetric (satisfying $d(i,j) = d(j,i)$), and moreover each color has 2^{h-1} colors at distance $h \geq 1$ from it.

We assume that pairs of colors at the same distance behave similarly. That is, the probability the classifier mistakenly labels a node of one color with another depends only on the distance between the colors, and similarly for the probability of an edge existing between two nodes. So, for example, $\gamma_{c_i c_j}$ is the same for all pairs (i,j) of distance h. A further simplification is to make the $\delta_{c_i c_j}$ and $\gamma_{c_i c_j}$ be simple functions of the distance. For example, we experimented with the case where these values decrease geometrically with the distance. Here $\delta_{c_i c_j} \sim \delta^{d(i,j)}$ for some $\delta > 0$, and similarly $\gamma_{c_i c_j} \sim \gamma^{d(i,j)}$ for some $\gamma > 0$.

Intuitively, this model captures the idea that when classifying pages, certain categories are more likely to be confused than others, and pages from certain categories are more likely to be linked than others.

4.2 Multiple colors: belief propagation The belief propagation technique is generalized in a straightforward manner to the case of multiple colors. Indeed, this is similar to extensions of the belief propagation technique in coding to symbols other than bits, such as field elements over GF(q) [7, 6]. In the classification setting, each node begins with a vector of probabilities. Each entry in this vector corresponds to a color, and the values represent the initial belief that the node is the corresponding color, based upon the color assigned by the initial classifier. The messages passed each round are now vectors of probabilities, instead of single probabilities.

As a conditional probability must be determined for each color, the processing time for each node in each round is proportional to the product of the number of neighbors and the square of the number of colors. Hence, the processing time per round is $O(mk^2)$ for k colors when the graph has m edges. Note also that the memory requirements of belief propagation grow linearly with the number of colors. Hence, introducing multiple colors severely slows down belief propagation.

In the case of many colors and large graphs, which is the situation faced when computing on the Web, belief propagation may prove too slow or require too much memory to be effective. We therefore propose a simplification similar to that for the case of two colors. It is worth noting, however, that theoretically the performance of belief propagation in the random graph model can be analyzed, in a manner similar to the case for two colors.

4.3 A simpler algorithm In the two color case we suggested an algorithm that sends messages of a single bit across each edge. Similarly, in the multicolor case, we suggest an algorithm that sends a single color, rather than a vector of colors. Let mfc(x) be the most frequently appearing color in messages from the neighbors other than $\lambda(x)$. The natural generalization is then:

Set $\lambda'(x,y) \leftarrow \lambda(x)$ unless mfc(x) wins the children vote by more than Δ_i votes; in this case, $\lambda'(x) \leftarrow$ mfc(x) .

If there are n nodes, m edges, and k colors in this graph, this simpler algorithm requires time $O(m + nk)$ per round, which can be a substantial savings over the $O(mk^2)$ time per round of belief propagation.

Analyzing this simple scheme becomes complex in the case of multiple colors. Suppose that we seek to mimic the case of two colors, by calculating a value p_{i+1} representing the probability that $\lambda'(x,y)$ takes on the wrong value. Now we must consider all the possible colors it could send. Moreover, besides taking into account all other possibilities for the most frequent colors, we must consider the remaining distribution of the colors of all other votes.

We may attempt to approximate the performance by attempting to reduce it to the two color case, effectively throwing away lower order terms in the analysis. In the color model we have described, each color has exactly one nearest neighbor. Hence, *solely as a means of making the equations computationally tractable* we make the following simplifications: we assume that if a node x is correctly classified initially, then $\lambda'(x,y)$ will either be $\kappa(x)$, or it will take on the value of the color nearest to $\kappa(x)$ in our model. That is, we assume that when a node is classified correctly initially, it is quite unlikely that some color other than its nearest neighbor will appear frequently enough in the children votes to change $\lambda'(x,y)$. Intuitively this still captures the high order behavior of the algorithm. Similarly, we assume that if x is incorrect initially, then $\lambda'(x,y)$ will take on either the initially assigned color or the correct color of x.

In full generality, we may let $p_i(c_a, c_b)$ be the probability that in the ith round $\lambda'(x,y)$ is c_b conditioned on the fact that the color of x is c_a. There are then three different cases, depending on whether $c_b = c_a$, c_b and c_a are nearest neighbors, or c_b and c_a are neither equal nor nearest neighbors. Each case yields a separate equation.

In the interests of space, we leave a fuller discussion of this approximate analysis as well as simulation results for the multiple color model we have described for the full version of the paper. We summarize our basic findings here. As in the two-color case, our simplified algorithm often performs nearly as well as belief propagation. Moreover, in the multiple color setting, the speedup is substantial, as one would expect. Our analysis, although only approximate, predicts performance well in most situations. In some cases, however, the simplifying assumptions are too optimistic, leading to deviations between the analysis and observed performance.

5 Conclusions

A main thrust of this work is that the classification problem is inherently similar to the decoding problem for certain types of codes. Based on this connection, we have considered both belief propagation and message passing approximations of belief propagation for classifying pages on the Web, using random graphs models. By studying these models through both simulation and analysis, we have found that using link information, even in simple ways, can lead to dramatically improved accuracy.

There are many further directions to take this work. One important liability in the current analysis is that the random graph model we focus on is perhaps too simplistic. Recently, other random graph models for the Web have been suggested [10], although finding a random graph model that suitably approximates the Web is still an open question. It would be worthwhile to study the performance of these improved classification schemes in a more suitable random graph model, either analytically or through simulations.

Another important direction is to gain a greater understanding of the situation where there are many colors. Analyzing the case of many colors fully appears to be an inherently complex problem, although the work of Kleinberg and Tardos demonstrates that approximation algorithms are possible [9]. It would also be interesting to experimentally determine a reasonable model for how categories behave on the Web, in order to guide subsequent analysis.

References

[1] C. Berrou, A Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *Proceedings of IEEE International Communications Conference*, 1993.

[2] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 104-111, 1998.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, pages 107-117. Elsevier Science, April 1998.

[4] S. Chakrabarti, B. Dom, D. Gibson, S.R. Kumar, S. Rajagopalan, and P. Raghavan. Experiments in topic distillation. In *ACM SIGIR Workshop on Hypertext Information Retrieval on the Web*, 1998.

[5] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

[6] M. Davey and D. MacKay. Low density parity check codes over GF(q). *IEEE Communications Letters*, vol. 2, no. 6, June 1998.

[7] R. G. Gallager. *Low density parity check codes.* MIT Press, Cambridge, MA, 1963.

[8] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998. Also available as IBM Research Report RJ 10076, May 1997.

[9] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 14–23,October 1999.

[10] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the Web. In *Proceedings of the 25th VLDB Conference*, 1999.

[11] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proceedings of the 29th Annual Symposium on Theory of Computing*, pages 150–159, 1997.

[12] M. Luby, M. Mitzenmacher, and M. A. Shokrollahi. Analysis of random processes via And-Or trees. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 364–373, 1998.

[13] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman. Improved low-density parity-check codes using irregular graphs and belief propagation. In *Proceedings of the 1998 International Symposium on Information Theory.*

[14] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman. Analysis of low density codes and improved designs using irregular graphs. In *Proceedings of the 30th Annual Symposium on Theory of Computing*, pages 249–258, 1998.

[15] R.J. McEliece, D. MacKay, and J. Feng. Turbo decoding as an instance of Pearl's "Belief Propagation" algorithm. *IEEE Journal on Selected Areas in Communication*, vol. 16, no. 2, pages 140-152, February, 1998.

[16] T. Richardson and R. Urbanke. The capacity of low-density parity check codes under message-passing decoding. Preprint, available at `http://cm.bell-labs.com/who/tjr/pub.html`.

[17] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of provably good low-density parity check codes. Preprint, available at `http://cm.bell-labs.com/who/tjr/pub.html`.