Efficient and Accurate Label Propagation on Large Graphs and Label Sets

Michele Covell and Shumeet Baluja
Google Research
Google Inc., Mountain View CA, USA
covell@google.com shumeet@google.com

Abstract—**Many web-based application areas must infer label distributions starting from a small set of sparse, noisy labels. Examples include searching for, recommending, and advertising against image, audio, and video content. These labeling problems must handle millions of interconnected entities (users, domains, content segments) and thousands of competing labels (interests, tags, recommendations, topics). Previous work has shown that graph-based propagation can be very effective at finding the best label distribution across nodes, starting from partial information and a weighted-connection graph. In their work on video recommendations, Baluja et al. [1] showed high-quality results using *Adsorption*, a normalized propagation process. An important step in the original formulation of Adsorption was re-normalization of the label vectors associated with each node, between every propagation step. That interleaved normalization forced computation of all label distributions, in synchrony, in order to allow the normalization to be correctly determined. Interleaved normalization also prevented use of standard linear-algebra methods, like stabilized bi-conjugate gradient descent (*BiCGStab*) and Gaussian elimination. This paper presents a method that replaces the interleaved normalization with a single pre-normalization, done once before the main propagation process starts, allowing use of selective label computation *(label slicing)* as well as large-matrix-solution methods. As a result, much larger graphs and label sets can be handled than in the original formulation and more accurate solutions can be found in fewer propagation steps. We also report results from using pre-normalized Adsorption in topic labeling for web domains, using label slicing and BiCGStab.**

Keywords-graph propagation, large-scale labeling, stabilized bi-conjugate gradient descent, Gaussian elimination, topic discovery, web domains.

I. INTRODUCTION

Many different approaches have recently been proposed to label propagation across weighted graphs of nodes [1,2,3,4,5,6]. These applications share the characteristics of having a limited amount of label data, often of uneven quality, associated with a large graph of weighted connections between many nodes, some unlabeled and some partially labeled.

We build on the work done by Zhu and Ghahramani [2] and Baluja et al. [1]. The Baluja paper described *Adsorption*, a graph-based approach to estimating label distributions, which was applied to providing YouTube video recommendations. The resulting top-pick recommendation was more accurate than the next-best alternative algorithm for all users who had watched 3 or more previous videos, with accuracy improvements of up to 100% for the most frequent watchers. In Adsorption, each node (e.g., each video for which we are building a recommendation list) has a limited capacity for labels (e.g., the proposed recommendations for that video). Baluja et al. [1] enforce this constraint by interleaving a normalization step at each node, in between every propagation step. Without this normalization, the solution is not guaranteed to converge.

The interleaved normalization step is needed for convergence but prevents label slicing: under the original formulation, we cannot find the estimated distribution of a subset of labels without solving for the full set of labels first. Furthermore, the interleaved normalization prevents the use of most standard linear-algebra techniques, such as Gaussian elimination of nodes that are not of direct interest (though they still are needed for their effect on the remainder of the graph). Additionally, methods for rapid convergence to the final solution, such as stabilized bi-conjugate gradient descent (BiCGStab), cannot be used in the original formulation.

This paper presents a formula for pre-normalizing the Adsorption graph and label weights, such that there is no need for interleaved normalization (Section III). With this, we can use BiCGStab and Gaussian elimination. Our graph size contains more than 10 million nodes and 4 billion inter-connections (i.e., more than 10 million rows and more than 4 billion non-zero entries in the corresponding matrix), which is more than we can reasonably handle in straightforward implementations of these techniques. Instead, we use implementations of BiCGStab and Gaussian elimination in the MapReduce framework. We describe these implementations briefly, in Sections IV and V. Finally, in Section VI, we present our results on topic labeling of web domains, using a graph based on shared keywords between pages across the domains. We start the paper with a recap of the original Adsorption application and mathematical description, in Section II.

II. ADSORPTION (WITH INTERLEAVED NORMALIZATION)

The original formulation of Adsorption [1] can be described as an iteration using two systems of equations:

$$\tilde{\underline{\underline{X}}}_{n+1} = \sigma \underline{\underline{X}}_n + \beta \underline{\underline{W}} \underline{\underline{X}}_n + \left[\begin{array}{cc} \gamma \underline{L} & \delta \underline{1} \end{array} \right] \quad (1)$$

$$\left\{ \underline{\underline{X}}_{n+1} \right\}_{i^*} = \left\{ \tilde{\underline{\underline{X}}}_{n+1} \right\}_{i^*} \Big/ \left\| \left\{ \tilde{\underline{\underline{X}}}_{n+1} \right\}_{i^*} \right\|_1 \quad (2)$$

where double underlining indicates a matrix of values, a single underline is a vector, not-underlined values are

 ISBN: 978-1-61208-265-3

scalars, and the tilde indicates a not-normalized set of values. The matrix $\underline{\underline{W}}$ holds the connection weights with row i giving the incoming connections into the i'th node. This matrix often is symmetric, to start with, but this property is not required and will be given up later to allow for pre-normalization. The matrix $\underline{\underline{L}}$ holds the weights of the *injection label* information. These are often noisy or incomplete label sets based on some prior information, with the graph propagation as a way to improve and expand these label sets. In $\underline{\underline{L}}$, each label is associated with a column and the weights for the injection labels for the i'th node of the graph in the i'th row of the matrix. In addition to the true labels, in $\underline{\underline{L}}$, Baluja et al. [1] add an *abandonment label*, represented in (1) by the appended column $\delta\underline{1}$. The scalar δ can be thought of in many different ways: as the loss in certainty about any of the labels that are propagated for one hop in the graph; as the number of random walks through the graph that end with "abandonment", giving no final label set; as the regularization margin in the system of equations. The other scalars (σ, β, and γ) allow graph-wide balancing of the previous (same-node) labels, of the propagated neighbors' labels, and of the injection labels. Finally, the matrix $\underline{\underline{X}}_n$ is the label distribution estimate, with the i'th row containing the estimated labels for the i'th node, including as the last column the abandonment label. In this context, the node's abandonment weight provides a measure, at that node, of the label uncertainty.

Equation (1) creates a new *un-normalized* estimate of the steady-state label distribution across all the nodes using a weighted combination of the previous normalized estimate for the distribution ($\underline{\underline{X}}_n$), of a graph-weighted propagated version of that same distribution ($\underline{\underline{W}}\underline{\underline{X}}_n$), of injection labels ($\underline{\underline{L}}$), and of the abandonment label (δ). Equation (2) provides a normalized estimate of the label distribution, by dividing each row of the estimate from (1) by the L_1 norm of the full label set, including the abandonment label.

Iterating over (1) and (2) together is guaranteed to converge to a stable steady-state solution, as long as δ is greater than 0. Baluja et al. [1] used this algorithm to successfully provide video recommendations that, using a top-pick-accuracy measure, outperformed alternative approaches. Our goal is to provide a formulation for the same Adsorption algorithm that does not require per-propagation-step normalization, allowing us to use label slicing and standard linear-algebra tools.

III. PRE-NORMALIZED ADSORPTION

We achieve our goal of pre-normalized Adsorption by first assuming that all associations in our graph and in our label injection are non-negative. Specifically: $sign\left(\{\underline{\underline{X}}_n\}_{ij}\right) \geq 0$, $sign\left(\{\underline{\underline{W}}\}_{ij}\right) \geq 0$, and $sign\left(\{\underline{\underline{L}}\}_{ij}\right) \geq 0$.

This non-negative assumption works well with the partial-information applications that are the most common ones in large-graph labeling formulations: for example, in video recommendation, we can say that two videos are often watched together, within a single viewing session, but it is much more difficult to say that two videos are negatively associated (that watching one means you are significantly less likely to watch the other), since we seldom have enough training data to make such an assertion with any confidence.

For those applications where we do have confidence in negative label-to-node associations (negative values in $\underline{\underline{L}}$), we can handle these by introducing a negated label column and using positive associations with the negated label where we would have otherwise used negative associations with the positive label. Handling negative node-to-node connections (negative values in $\underline{\underline{W}}$) is also possible but we omit it here, since it is an uncommon use case (and is much more complicated).

Assuming we have non-negative values in our component matrices, we can consider the denominator of (2) in more detail:

$$\left\|\{\tilde{\underline{\underline{X}}}_{n+1}\}_{i*}\right\|_1 = \left\|\left\{\left(\sigma\underline{\underline{I}} + \beta\underline{\underline{W}}\right)\underline{\underline{X}}_n + \begin{bmatrix} \gamma\underline{\underline{L}} & \delta\underline{1} \end{bmatrix}\right\}_{i*}\right\|_1 \tag{3}$$

$$= \sum_j \left(\sum_k \{\sigma\underline{\underline{I}} + \beta\underline{\underline{W}}\}_{ik} \{\underline{\underline{X}}_n\}_{kj} \right) + \sum_j \left\{ \begin{bmatrix} \gamma\underline{\underline{L}} & \delta\underline{1} \end{bmatrix} \right\}_{ij} \tag{4}$$

$$= \sum_k \{\sigma\underline{\underline{I}} + \beta\underline{\underline{W}}\}_{ik} \sum_j \{\underline{\underline{X}}_n\}_{kj} + \gamma \sum_j \{\underline{\underline{L}}\}_{ij} + \delta \tag{5}$$

$$= \sum_k \{\sigma\underline{\underline{I}} + \beta\underline{\underline{W}}\}_{ik} + \gamma \left\|\{\underline{\underline{L}}\}_{i*}\right\|_1 + \delta \tag{6}$$

$$= \sigma + \beta \left\|\{\underline{\underline{W}}\}_{i*}\right\|_1 + \gamma \left\|\{\underline{\underline{L}}\}_{i*}\right\|_1 + \delta \tag{7}$$

Equation (3) simply provides the expansion of the L_1 row norm using the propagation (1). Equation (4) makes use of the non-negativity conditions that we are requiring, in order to remove the absolute values implied by the L_1 norm and expands the norm summation, as well as the summation implicit in the $\underline{\underline{W}}\underline{\underline{X}}_n$ matrix multiply. Equation (5) swaps the order of summation, allowing us to make use of the unit L_1 row norm for $\underline{\underline{X}}_n$ in (6). Simplifying the summations and noting the use of the row-norm definitions for $\underline{\underline{L}}$ and $\underline{\underline{W}}$ finally results in (7).

The useful property of (7) is that $\left\|\{\tilde{\underline{\underline{X}}}_{n+1}\}_{i*}\right\|_1$ depends only the initial combination weights and the row norms of $\underline{\underline{L}}$ and $\underline{\underline{W}}$. We can use this property to pre-normalize by first defining

$$\lambda_i = \sigma + \beta \left\|\{\underline{\underline{W}}\}_{i*}\right\|_1 + \gamma \left\|\{\underline{\underline{L}}\}_{i*}\right\|_1 + \delta \tag{8}$$

$$\hat{\sigma}_i = \sigma / \lambda_i \qquad \hat{\underline{\underline{\sigma}}} = diag(\hat{\sigma}_i) \tag{9}$$

$$\hat{\beta}_i = \beta \left\|\{\underline{\underline{W}}\}_{i*}\right\|_1 / \lambda_i \qquad \hat{\underline{\underline{\beta}}} = diag(\hat{\beta}_i) \tag{10}$$

$$\hat{\gamma}_i = \gamma \left\|\{\underline{\underline{L}}\}_{i*}\right\|_1 / \lambda_i \qquad \hat{\underline{\underline{\gamma}}} = diag(\hat{\gamma}_i) \tag{11}$$

$$\hat{\delta}_i = \delta / \lambda_i \qquad \hat{\underline{\delta}} = vec(\hat{\delta}_i) \tag{12}$$

and then using these new quantities in a pre-normalized Adsorption algorithm.

$$\underline{\underline{X}}_{n+1} = \hat{\underline{\underline{\sigma}}}\underline{\underline{X}}_n + \hat{\underline{\underline{\beta}}}\underline{\underline{W}}\underline{\underline{X}}_n + \begin{bmatrix} \hat{\underline{\gamma}}\underline{\underline{L}} & \hat{\underline{\delta}} \end{bmatrix} \tag{13}$$

 ISBN: 978-1-61208-265-3

Note that direct use of (13) is exactly the power-iteration approach to finding the solution (used in [1]) and will give the same solutions at every iteration as the combination of (1) and (2): the pre-normalization has the exact same effect, even though it is only done once, as the interleaved normalizations. Equation (13), therefore, also is guaranteed to converge to a stable solution, just as the original Adsorption algorithm is guaranteed. The advantage is that we do not need to normalize at each step and, as a result, we can compute an incomplete set of labels, while still deriving the benefits of the full label set to limit belief within the set of labels that are interested in. This slicing directly reduces the computational costs by the same percentage as the percentage of dropped labels. Furthermore, with the use of (13) as the system of equations for which we want a solution, we can use standard linear-algebra tools, like BiCGStab (for faster convergence) and Gaussian elimination (for shrinking our graph matrix). We discuss these algorithms and their large-graph implementations next.

IV. MAP-REDUCE FORMULATION OF STABILIZED BI-CONJUGATE GRADIENT DESCENT (BICGSTAB)

In [1], Baluja et al. implicitly use power iteration to solve their system of constraints. For symmetric systems of constraints, gradient-descent methods can find solutions in fewer iterations, for any given level of accuracy (as measured by the average residual error). However, due to the pre-normalization of Adsorption, we no longer have a symmetric matrix, and must move to bi-conjugate gradient approaches. Since the most direct generalization (biconjugate gradient descent) is not numerically stable, we focus on stabilized biconjugate gradient descent [7], which has been shown to converge more uniformly than power iteration, without the numerical issues (not-stablized) bi-conjugate gradient descent. We ran several simulations using power iteration and BiCGStab, based on random graph matrices with the same level of regularization as we expect to see through the abandonment variable in our true graphs. In these tests, when the graph matrix and the beginning label estimates were non-sparse, on average, BiCGStab converged to the correct solution 12 times faster than the power-iteration method (e.g., BiCGStab would converge in two iterations, requiring only 5 graph-matrix multiplies, while power iteration would require 60 iterations, needing 60 graph-matrix multiplies to converge to the same level of accuracy).

When the graph matrix and the beginning label estimates were sparse, there were similar differences in the rate of convergence, away from the "wavefront boundary". We use the term *wavefront* to emphasize that (for both power iteration and BiCGStab), updates are done in such a way that non-zero values propagate through the graph according to the neighborhood connections. When the labels are sparsely injected, non-zero values move in a "wave", outward from non-zero areas into areas that were zero (due to sparseness). Both power iteration and BiCGStab rely on the graph matrix to determine the label-estimate update, so both have their non-zero wavefronts progress in the same way.

Due to the size of the graph over which we will be operating, we implemented BiCGStab using three MapReduce [8] stages per iteration. Using the notation from the Wikipedia article on BiCGStab [9], we have a distinct set of vectors for each of the labels on which we want to estimate the final distribution. We arrive at the BiCGStab components $\underline{\underline{A}}$ and $\underline{b}$ (at least conceptually) by separating $\hat{\gamma}\underline{\underline{L}}$ into columns corresponding to $\underline{b}$, by separating $\underline{\underline{X}}_n$ into columns corresponding to $\underline{x}_n$ and by using

$$\underline{\underline{A}} = \underline{\underline{I}} - \hat{\alpha}\underline{\underline{W}} - \hat{\beta}\underline{\underline{W}} \quad (14)$$

We select an initial *shadow direction* $\hat{\underline{r}}_0$ for each column aligned with its first-pass residual vector, $\underline{r}_0$. Note that computing the first-pass residual vector takes one MapReduce to compute $\underline{r}_0 = b - \underline{\underline{A}}\underline{x}_0$. (For our applications, $\underline{b}$ itself is often a good initial estimate for $\underline{x}$.) It is this separate estimation of each column (where each column corresponds to a single label) that makes label slicing so simple and powerful in combination with BiCGStab.

Unlike [9], we mark all our auxiliary variables with the iteration on which they were computed, since this makes our Reduce processing more uniform and reliable: therefore, we use α_n, $\underline{s}_n$ and $\underline{t}_n$ here (instead of their un-versioned form from [9]). To allow the remaining framework to operate smoothly, starting from the initialization (the 0'th pass), we also use the settings for our auxiliary variables that are suggested in [9], namely: $\rho_0 = \alpha = \omega_0 = 1$ and $\underline{v}_0 = \underline{p}_0 = \underline{0}$.

For all iterations after this initialization, there are 3 MapReduce stages: (*A*) updating the search direction and its projection through $\underline{\underline{A}}$; (*B*) updating the shadow direction and its projection through $\underline{\underline{A}}$; and (*C*) combining the computed components to give a new state estimate and residual.

For all three MapReduce stages, the reduce processing is the same: from the set of inputs computed in the Map stage, as well as the inputs passed directly through to the Reducer from previous stages or iterations, keep and combine the results for each variable (auxiliary variables, residual, and state estimate) that is marked with the highest iteration number observed for that variable, and throw away earlier versions.

A. Updating the search direction and its projection

1) *Map (shared) context:*
   a. From initial selection: $\hat{\underline{r}}_0$
   b. From previous iteration:
      ρ_{n-1}, α_{n-1}, ω_{i-1}, $\underline{r}_{n-1}$, $\underline{v}_{n-1}$, $\underline{p}_{n-1}$
   c. From pre-map computation:
      $\rho_n = \langle \hat{\underline{r}}_0, \underline{r}_{n-1} \rangle$
      $\underline{p}_n = \underline{r}_{n-1} + \left(\frac{\rho_n}{\rho_{n-1}}\right)\left(\frac{\alpha_{n-1}}{\omega_{n-1}}\right)\left(\underline{p}_{n-1} - \omega_{n-1}\eta_{n-1}\right)$
2) *Map computation:*
   For each row in $\underline{\underline{A}}$, *compute* $\left\{\underline{\eta}_n\right\}_i = \left\{\underline{\underline{A}}\right\}_{i*} \underline{p}_n$

B. Updating the shadow direction and its projection

1) *Map (shared) context:*

a. From initial selection: $\hat{\underline{r}}_0$
b. From previous iteration: $\underline{r}_{n-1}$
c. From previous stage of current iteration: ρ_n, $\underline{\eta}_n$
d. From pre-map computation:
$\alpha_n = \rho_n \big/ \langle \hat{\underline{r}}_0, \underline{\eta}_n \rangle$
$\underline{s}_n = \underline{r}_{n-1} - \alpha_n \underline{\eta}_n$

2) *Map computation:*
For each row in $\underline{\underline{A}}$, *compute* $\{\underline{t}_n\}_i = \{\underline{\underline{A}}\}_{i^*} \underline{s}_n$

C. *Combining components for residual and state estimates*

1) *Map (shared) context:*
a. From previous iteration: $\underline{x}_{n-1}$
b. From previous stages of current iteration: α_n, $\underline{s}_n$, $\underline{t}_n$, $\underline{p}_n$

2) *Map computation: For each label, compute*
$\omega_n = \langle \underline{s}_n, \underline{t}_n \rangle \big/ \langle \underline{t}_n, \underline{t}_n \rangle$
$\underline{x}_n = \underline{x}_{n-1} + \alpha_n \underline{p}_n + \omega_n \underline{s}_n$
$\underline{r}_n = \underline{s}_n - \omega_n \underline{t}_n$

V. MAPREDUCE FORMULATION OF GAUSSIAN ELIMINATION

Label slicing allows us to compute our distributions on the subset of labels that are of most interest, while still benefiting from the constraints effectively imposed by the full label set. In a similar way, Gaussian elimination allows us to compute our distribution on a subset of nodes (domains), while still benefiting from the indirect interconnections that are formed through the nodes that we do not want to explicitly include in our calculation. The computational savings provided by Gaussian elimination is linear with the percentage reduction in the number of graph connections. In addition, Gaussian elimination can speed up convergence, by effectively increasing the wavefront-propagation speed through those parts of the graph that were originally connected via the eliminated nodes.

Gaussian elimination is much simpler to implement in the MapReduce framework than BiCGStab, requiring only a single stage and capable of handling elimination of multiple nodes per run. The Reduce processing in the MapReduce is a straight pass-through of the outputs from the map stage.

To make the description more concise, define

$$\underline{\underline{A}}_{keep} = \{\underline{\underline{A}}\}_{i^*} \quad \underline{\underline{L}}_{\gamma\,keep} = \{\hat{\underline{\underline{\gamma}}}\underline{\underline{L}}\}_{i^*} \quad i \in \left\{ \begin{array}{c} nodes \\ to\ be\ kept \end{array} \right\}$$

$$\underline{\underline{A}}_{remove} = \{\underline{\underline{A}}\}_{j^*} \quad \underline{\underline{L}}_{\gamma\,remove} = \{\hat{\underline{\underline{\gamma}}}\underline{\underline{L}}\}_{j^*} \quad j \in \left\{ \begin{array}{c} nodes\ to\ be \\ eliminated \end{array} \right\}$$

Using this notation, the map processing is

1) *Map (shared) context:*
From stored representation:
$\underline{\underline{A}}_{remove}$, $\underline{\underline{L}}_{\gamma\,remove}$

2) *Map computation: For each row, i, in* $\underline{\underline{A}}_{keep}$ *and* $\underline{\underline{L}}_{\gamma\,keep}$

a) Initialize

$$\tilde{\underline{\underline{A}}}_{keep} = \underline{\underline{A}}_{keep}, \quad \tilde{\underline{\underline{A}}}_{remove} = \underline{\underline{A}}_{remove}$$
$$\tilde{\underline{\underline{L}}}_{\gamma\,keep} = \underline{\underline{L}}_{\gamma\,keep}, \quad \tilde{\underline{\underline{L}}}_{\gamma\,remove} = \underline{\underline{L}}_{\gamma\,remove}$$

b) Compute the pivot strength, π_{ij}, for each $j \in \{nodes\ to\ be\ eliminated\}$:

$$\pi_{ij} = \{\tilde{\underline{\underline{A}}}_{keep}\}_{ij} \big/ \{\tilde{\underline{\underline{A}}}_{remove}\}_{jj}$$

and select the elimination node, $\tilde{j}$, with the smallest amplitude $|\pi_{ij}|$

c) Eliminate all non-zero entries in the $\tilde{j}$'th column in $\{\tilde{\underline{\underline{A}}}_{keep}\}_{i^*}$ and $\tilde{\underline{\underline{A}}}_{remove}$, with matched operations on $\{\tilde{\underline{\underline{L}}}_{\gamma\,keep}\}_{i^*}$ and $\tilde{\underline{\underline{L}}}_{\gamma\,remove}$:

$$\{\tilde{\underline{\underline{A}}}_{keep}\}_{ik} \leftarrow \{\tilde{\underline{\underline{A}}}_{keep}\}_{ik} - \pi_{i\tilde{j}} \{\tilde{\underline{\underline{A}}}_{remove}\}_{\tilde{j}k}$$
$$\{\tilde{\underline{\underline{L}}}_{\gamma\,keep}\}_{ik} \leftarrow \{\tilde{\underline{\underline{L}}}_{\gamma\,keep}\}_{ik} - \pi_{i\tilde{j}} \{\tilde{\underline{\underline{L}}}_{\gamma\,remove}\}_{\tilde{j}k}$$
$$\{\tilde{\underline{\underline{A}}}_{remove}\}_{nk} \leftarrow \{\tilde{\underline{\underline{A}}}_{remove}\}_{nk} - \tilde{\pi}_{n\tilde{j}} \{\tilde{\underline{\underline{A}}}_{remove}\}_{\tilde{j}k} \quad \forall n \neq \tilde{j}$$
$$\{\tilde{\underline{\underline{L}}}_{\gamma\,remove}\}_{nk} \leftarrow \{\tilde{\underline{\underline{L}}}_{\gamma\,remove}\}_{nk} - \tilde{\pi}_{n\tilde{j}} \{\tilde{\underline{\underline{L}}}_{\gamma\,remove}\}_{\tilde{j}k} \quad \forall n \neq \tilde{j}$$
$$\text{with } \tilde{\pi}_{nj} = \{\tilde{\underline{\underline{A}}}_{remove}\}_{nj} \big/ \{\tilde{\underline{\underline{A}}}_{remove}\}_{jj}$$

d) Remove row $\tilde{j}$ from $\tilde{\underline{\underline{A}}}_{remove}$, $\tilde{\underline{\underline{L}}}_{\gamma\,remove}$

e) Repeat (b), (c), and (d), until there are no more rows (nodes) to be removed.

f) Output $\{\tilde{\underline{\underline{A}}}_{keep}\}_{i^*}$ and $\{\tilde{\underline{\underline{L}}}_{\gamma\,keep}\}_{i^*}$

VI. LARGE-SCALE DOMAIN-LEVEL TOPIC LABELING

Baluja et al. [1] already showed the usefulness of the Adsorption approach in video recommendations. The pre-normalized Adsorption algorithm provides identical results at a fraction of the computational cost using the new formulation with label slicing, Gaussian elimination, and BiCGStab. The final computational cost is reduced by the product of the savings of all three approaches (label slicing, BiCGStab and Gaussian elimination).

For this paper, we explored using pre-normalized Adsorption for topic labeling on web domains, for search and advertising. Many page urls, and even whole domains, are poorly classified by standard topic-analysis approaches, due to having little in the way of machine-understandable content to classify. A standard example of this problem are domains that primarily host images or video – while the page url can be examined for clues to the topic, as well as the linked-to urls, the results are impoverished and noisy. If we can improve the topic labeling, we could more accurately index these pages for search and for content-matched advertisement.

 ISBN: 978-1-61208-265-3

Specifically, we created a graph with domains as nodes and a measure of shared searches for cross-domain pairs of urls as weighted connections between nodes. Our measure looked at, for each search term, the click rates for each url served in the results and set the strength of the url-url-term triple to the lower of the click rates between the paired urls. The connection weight between pairs of urls is the sum over all triples that terminate at those two urls. To aggregate from url-pair connections, up to domain-pair connections, we sum across those url-pair connections where the first of the pair of urls is from the first domain and the second is from the second domain. Similarly, our injection labeling is based on combining topic analysis of the urls within the domain, dropping those topics that were based on keywords that showed too much within-domain variance in their strength. We aggregate the link and topic-label strength up to the domain level to improve coverage and reliability of our graph connections. Even with this aggregation of urls to domain-level nodes and filtering of keyword labels to within-domain-stable sets, our initial data provides a graph of about 13 million domains (nodes), with about 4 billion node-to-node connections based on analysis of more than 253 million search terms. Our topic analysis provides more than 4,500 general topics, using traditional text-based classification.

From this set of 4,500 topics, we focused on 71 commercial topics (see Fig. 1 for examples). The computational savings (over the original Adsorption approach) for the label slicing alone was a factor of 63 times. We do not include this savings in the remainder of this discussion, since it is available to both power iteration and BiCGstab, as long as we are using the pre-normalized Adsorption formulation. That said, it is the most significant source of computational savings, compared to the original work [1].

We ran this set of 71 labels through two iterations of BiCGStab (5 graph-matrix multiplies) and through 70 iterations of the power method, both starting from the same initial estimate. Fig. 2 shows the size of the per-node residual for BiCGStab on these labels (using an L_1 norm). As with our small-scale simulations, at the end of our second iteration, the not-insignificant residuals occurred at the 3% of the nodes that were at the "wavefront boundary" of one or more of the topic labels. This level of convergence, with just 5 matrix multiplies, is not seen in the power-iteration

- Clothing
  - Women's, Men's, Children's
  - Athletic, Casual, Formal, Outerwear, Sleepwear
  - Shoes, Boots
- Accessories
  - Jewelry, Watches, Purses
- Toys
  - Building Toys, Dolls, Stuffed Animals, Ride-on Toys
- Gifts
  - Flowers, Cards, Party Items, Holiday Items
- Discounts
  - Coupons, Loyalty Cards

Figure 1: Examples from selected 71 commercial topics.

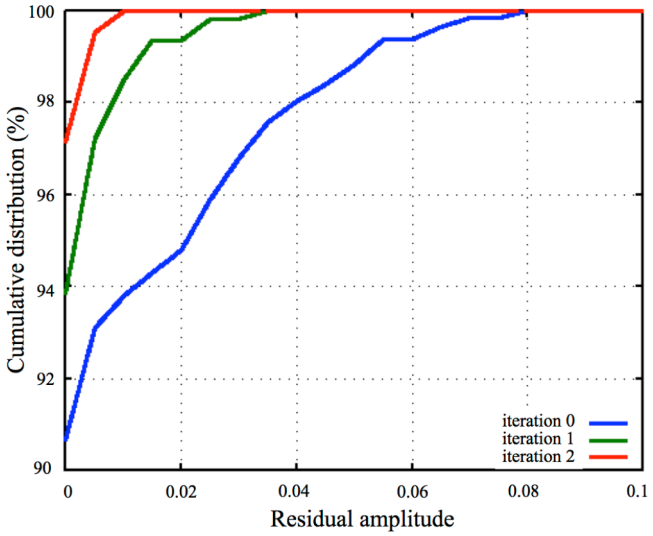

Figure 2: Cumulative residual distribution (by iteration).

solution until the 62th iteration (an additional savings of nearly 12.5 times).

Since the goal of our label propagation is to increase the richness and extent of the topic labeling on poorly labeled (or unlabeled) domains without over-extending into domains that are not related to our commercial subset, it is helpful to look at the statistics summarized in Fig. 3 through 5.

Fig. 3 gives a measure of the richness of our labels on commercial domains and how that richness increases as a function of iteration. The plot shows the percentages of domains by how many commercial-topic labels are seen on that domain. If a domain is commercial, the more commercial labels that are associated with the domain, the richer the topic description. As shown by the plots, our injection labels (those given by topic analysis) within each domain provides sparse topic labels, with the largest percentage of commercial domains having only one label.

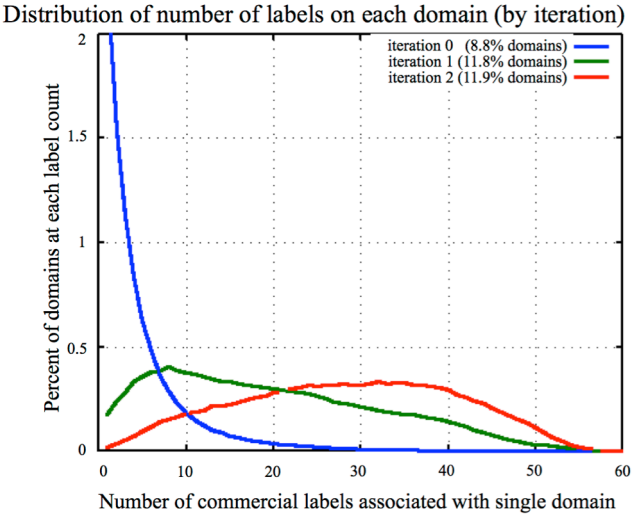

Figure 3: Node-level coherence of commercial labels.

Since our 71 commercial topics are actually a hierarchical set, this sparseness is unlikely to be correct for most domains. By the end of the second iteration, the mode of that distribution has moved to around 30 topic labels per commercial domain.

Also, the legend in Fig. 3 gives us the information needed to check that we are not just expanding the support of our commercial-topic labels indiscriminately across the full domain graph. The first iteration extends the support of the commercial labels by a third, from just under 9% of all domains to just under 12%, suggesting the addition of a subset of the unlabeled domains within the graph. After the first iteration, the support of the commercial-label set is effectively unchanged. This can be traced back to the effect of pre-normalizing on the full set of topic labels. Even though the non-commercial topics are not being explicitly computed in our iterations, they still have an effect, keeping the commercial labels from spreading onto distant (in the graph-connection sense) domains, as they otherwise would as the commercial wavefront progressed. This highlights both one of the main advantages of the original Adsorption as well as the most compelling advantage of the pre-normalized Adsorption. With the original Adsorption, each node has a limited capacity for supporting labels, thereby limiting propagation – but enforcing that limited capacity forced computation of all label distributions, not just the labels of interest. With pre-normalized Adsorption, there is still the per-node limited capacity for supporting labels, but we achieve that capacity limit by pre-normalizing, freeing us to compute only at that subset of labels that we are interested in, without having those labels spread unchecked.

Up to now, our analysis of our results has focused on the richness and extent of our commercial labels but not on the likely quality of the mix of labels that we are introducing onto commercial nodes. Since our topics are structured into a hierarchical framework, intuitively what we would like is to have each commercial site labeled mostly by closely related subsets of the available topics. We can use *dendrite distances* between the labels to capture this sense of closeness among the sets of labels associated with each domain node. As with standard dendrite measures, for each pair of labels on a domain, we count the number of hierarchical topic links that we have to go across in order to travel from one topic label to the other. We lengthen that distance by one for each generation that *both* labels have to travel back through, in order to penalize siblings more than grandparent-grandchild relations. As an example, if we need to calculate the distance between women's jewelry and men's clothing and we have the two tree branches "Jewelry -> Women's Accessories -> Apparel" and "Men's Clothing -> Apparel", our dendrite distance measure would be 4: two (for "Women's Jewelry" to "Apparel") plus one (for "Men's Clothing" to "Apparel") plus one (for the one generation removal from direct descendent connection).

As a way to evaluate our label distributions on domains with 2 to 6 labels, we computed all pairwise dendrite distances within each domain and averaged them (again, on a per-domain basis). Due to the use of the topic hierarchy in our dendrite-distance measure, smaller distances amongst the labels on a single domain correspond to more believable topic mixes. Fig. 4 shows our results, as function of iteration. When the initial topic labeling provides more than one label, it includes many dissimilar labels, with the mode of the dendrite average distance being up between 6 and 7. Our propagation reduces that average distance, filling in parent and children nodes, to give a mode that is just above one. While parents could always be filled in by knowing the hierarchical structure of our topic labels, the propagation graph is doing this without that knowledge – it is finding these associations purely through propagation of neighbor labels. (Furthermore, we could not use the tree-structure meta-information to fill in the correct children labels – if we blindly used the tree structure, we would get numerous nearby but irrelevant labels.) For this set of nodes, we are enriching the topic description without introducing unrelated labels. This measure of quality is a stringent one, since at no point do we use the dendrite structure to limit our propagation.

Fig. 5 shows a similar measure, for domains with more than 6 labels, again averaging the dendrite distances within each node. We did this separation between Fig. 4, for domains with 2-6 commercial labels, and Fig. 5, for domains with more than 6 commercial labels, since the dendrite distances across larger sets of labels, taken from the same hierarchy will have a larger minimum-average distance than will smaller sets of labels. For small sets, you can often find 2-6 labels, with all parent-child or sibling relationships with one another but, for large sets of labels, this is not possible and first and second cousin relationships become a major part of even the most compact set of labels. Same as with Fig. 4, Fig. 5 shows that the average dendrite distance decreases with each iteration, even on nodes with more than 6 labels. Since closely related sets of topic labels are more likely to be a full and accurate description of the domain topic, our topic labeling seems to be improved by our graph

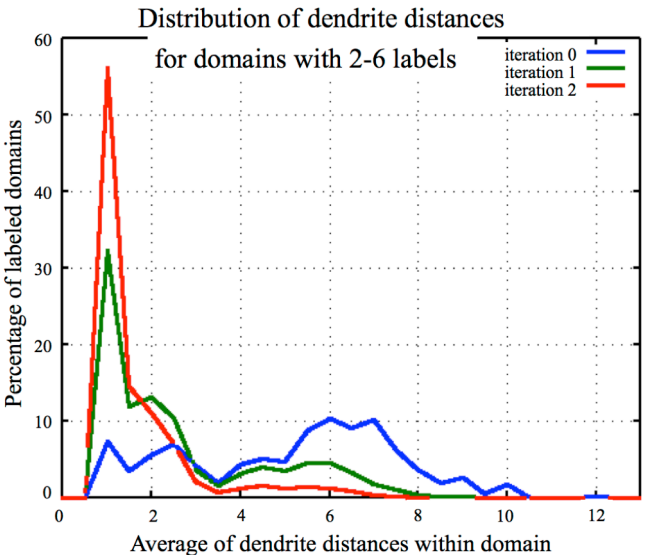

Figure 4: Dendrite topic-label distance on domains with 2-6 labels (by iteration).

 ISBN: 978-1-61208-265-3

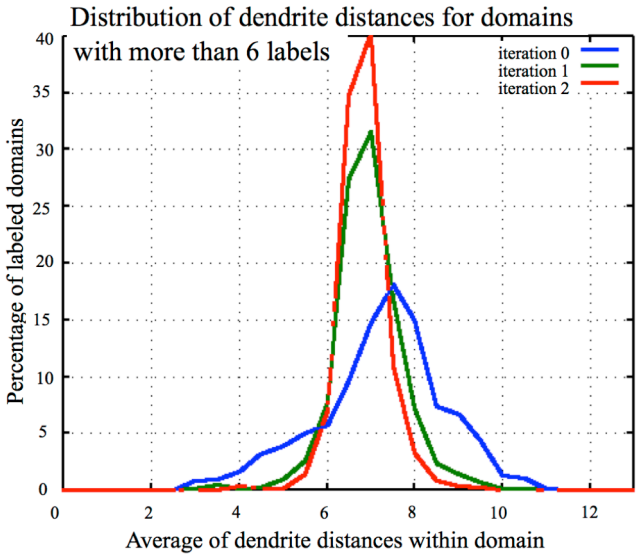

Figure 5: Dendrite topic-label distance on domains with more than 6 labels (by iteration).

propagation work.

All of the measurements conducted on the propagation of web labels on this large set of domains indicate an improvement in search indexing and content-matched advertising. In the future, we will expand these experiments in two directions. First, we will run live trials, with full user-facing experiments, to determine the quality improvement in the user experience. Second, we will increase our graph size and specificity by including individual urls, for those sites that have enough textual information to support that level of analysis.

VII. CONCLUSIONS

This paper improves the computational efficiency of Adsorption, a graph-based labeling approach that has already been shown to be highly effective. We do so by replacing propagation-interleaved normalization with pre-normalization, without changing the results provided by Adsorption. Specifically, if the power-method approach to finding a solution is used, as it was with Adsorption, the answers at every iteration will be exactly the same using either the original or the pre-normalized Adsorption. The advantage of the pre-normalized Adsorption is computational efficiency in determining the label distribution. With the pre-normalized version, we can use label slicing, to compute only those labels that are of direct interest, without losing the beneficial belief-limiting characteristics of the full label set. Label slicing reduces the computational cost linearly with the percentage of dropped labels. Similarly, we can use Gaussian elimination, to compute the labels only on those nodes that are of direct interest, without losing the effects of the connections that occur indirectly through currently not-of-interest nodes. Finally, we can speed up convergence to the steady-state solution by a factor of 12 (in numbers of graph matrix multiples), by using stabilized biconjugate gradient descent, instead of power iteration. We also applied pre-normalized Adsorption to a new, large-scale application area, topic labeling on web domains, with promising results.

REFERENCES

[1] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly, "Video suggestion and discovery for YouTube: taking random walks through the view graph," Proc. International Conference on World Wide Web, ACM, April 2008, pp. 895-904.

[2] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," CMU tech report, CMU-CALD-02-107, 2002.

[3] P.P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira, "Weakly-supervised acquisition of labeled class instances using graph random walks," Proc. Conf. on Empirical Methods in Natural Language Processing, Assoc. Computational Linguistics, October 2008, pp. 582-590.

[4] Y. Jing and S. Baluja, "Visual Rank: applying Page Rank to large-scale image search," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 30, November 2008, pp. 1877-1890.

[5] J. Liu, W. Lai, X S. Hua, Y. Huang, and S. Li, "Video search re-ranking via multi-graph propagation," Proc. International Conference on Multimedia, ACM, September 2007, pp. 208-217.

[6] M. Speriosu, N. Sudan, S. Upadhyay, and J. Baldridge, "Twitter polarity classification with label propagation over lexical links and the follower graph," Proc. Workshop on Unsupervised Learning in NLP, Assoc. Computational Linguistics, July 2011, pp. 53-63.

[7] H. A. Van der Vorst, "Bi-CGSTAB: A Fast and Smoothly Converging Variant of BiCG for the Solution of Nonsymmetric Linear Systems," SIAM Journal on Scientive and Statistical Computing, vol. 13, March 1992, pp. 631-644.

[8] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Symposium on Operating System Design and Implementation, December 2004, pp. 137-150.

[9] Wikipedia, "Biconjugate Gradient Stablized Method," http://en.wikipedia.org/wiki/Biconjugate_gradient_stabilized_method [retrieved February, 2013].

 ISBN: 978-1-61208-265-3