Ranking and Semi-supervised Classification on Large Scale Graphs Using Map-Reduce

Delip Rao
Dept. of Computer Science
Johns Hopkins University
delip@cs.jhu.edu

David Yarowsky
Dept. of Computer Science
Johns Hopkins University
yarowsky@cs.jhu.edu

Abstract

Label Propagation, a standard algorithm for semi-supervised classification, suffers from scalability issues involving memory and computation when used with large-scale graphs from real-world datasets. In this paper we approach Label Propagation as solution to a system of linear equations which can be implemented as a scalable parallel algorithm using the map-reduce framework. In addition to semi-supervised classification, this approach to Label Propagation allows us to adapt the algorithm to make it usable for ranking on graphs and derive the theoretical connection between Label Propagation and PageRank. We provide empirical evidence to that effect using two natural language tasks – lexical relatedness and polarity induction. The version of the Label Propagation algorithm presented here scales linearly in the size of the data with a constant main memory requirement, in contrast to the quadratic cost of both in traditional approaches.

1 Introduction

Natural language data often lend themselves to a graph-based representation. Words can be linked by explicit relations as in WordNet (Fellbaum, 1989), and documents can be linked to one another via hyperlinks. Even in the absence of such a straightforward representation it is possible to derive meaningful graphs such as the nearest neighbor graphs, as done in certain manifold learning methods, e.g. Roweis and Saul (2000); Belkin and Niyogi (2001). Typically, these graphs share the following properties:

- They are edge-weighted.
- The edge weight encodes some notion of relatedness between the vertices.
- The relation represented by edges is at least weakly transitive. Examples of such relations include, "is similar to", "is more general than", and so on. It is important that the relations selected are transitive for the graph-based learning methods using random walks.

Such graphs present several possibilities for solving natural language problems involving ranking, classification, and clustering. Graphs have been successfully employed in machine learning in a variety of supervised, unsupervised, and semi-supervised tasks. Graph based algorithms perform better than their counterparts as they capture the latent structure of the problem. Further, their elegant mathematical framework allows simpler analysis to gain a deeper understanding of the problem. Despite these advantages, implementations of most graph-based learning algorithms do not scale well on large datasets from real world problems in natural language processing. With large amounts of unlabeled data available, the graphs can easily grow to millions of nodes and most existing non-parallel methods either fail to work due to resource constraints or find the computation intractable.

In this paper we describe a scalable implementation of Label Propagation, a popular random walk based semi-supervised classification method. We show that our framework can also be used for ranking on graphs. Our parallel formulation shows a theoretical connection between Label Propagation and PageRank. We also confirm this empirically using the lexical relatedness task. The

proposed Parallel Label Propagation scales up linearly in the data and the number of processing elements available. Also, the main memory required by the method does not grow with the size of the graph.

The outline of this paper is as follows: Section 2 introduces the manifold assumption and explains why graph-based learning algorithms perform better than their counterparts. Section 3 motivates the random walk based approach for learning on graphs. Section 4 introduces the Label Propagation method by Zhu et al. (2003). In Section 5 we describe a method to scale up Label Propagation using Map-Reduce. Section 6 shows how Label Propagation could be used for ranking on graphs and derives the relation between Label Propagation and PageRank. Parallel Label Propagation is evaluated on ranking and semi-supervised classification problems in natural language processing in Section 8. We study scalability of this algorithm in Section 9 and describe related work in the area of parallel algorithms and machine learning in Section 10.

2 Manifold Assumption

The training data $\mathcal{D}$ can be considered as a collection of tuples $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$ where $\mathcal{Y}$ are the labels and $\mathcal{X}$ are the features, and the learned model $\mathcal{M}$ is a surrogate for an underlying physical process which generates the data $\mathcal{D}$. The data $\mathcal{D}$ can be considered as a sampling from a smooth surface or a manifold which represents the physical process. This is known as the manifold assumption (Belkin et al., 2005). Observe that even in the simple case of Euclidean data ($\mathcal{X} = \{x : x \in \mathbb{R}^d\}$) as shown in Figure 1, points that lie close in the Euclidean space might actually be far off on the manifold. A graph, as shown in Figure 1c, approximates the structure of the manifold which was lost in vectorized algorithms operating in the Euclidean space. This explains the better performance of graph algorithms for learning as seen in the literature.

3 Distance measures on graphs

Most learning tasks on graphs require some notion of distance or similarity to be defined between the vertices of a graph. The most obvious measure of distance in a graph is the shortest path between the vertices, which is defined as the minimum number of intervening edges between two vertices. This is also known as the geodesic distance. To convert this distance measure to a similarity measure, we take the reciprocal of the shortest-path length. We refer to this as the geodesic similarity.

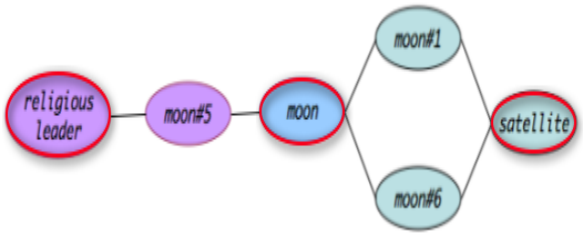


Figure 2: Shortest path distances on graphs ignore the connectivity structure of the graph.

While shortest-path distances are useful in many applications, it fails to capture the following observation. Consider the subgraph of WordNet shown in Figure 2. The term `moon` is connected to the terms `religious_leader` and `satellite`.[1] Observe that both `religious_leader` and `satellite` are at the same shortest path distance from `moon`. However, the connectivity structure of the graph would suggest `satellite` to be more similar than `religious_leader` as there are multiple senses, and hence multiple paths, connecting `satellite` and `moon`.

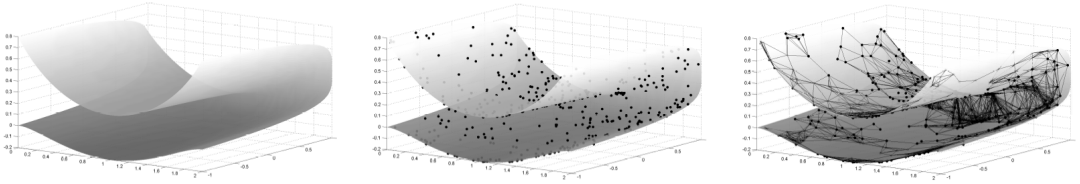
Thus it is desirable to have a measure that captures not only path lengths but also the connectivity structure of the graph. This notion is elegantly captured using random walks on graphs.

4 Label Propagation: Random Walk on Manifold Graphs

An efficient way to combine labeled and unlabeled data involves construction of a graph from the data and performing a Markov random walk on the graph. This has been utilized in Szummer and Jaakkola (2001), Zhu et. al. (2003), and Azran (2007). The general idea of Label Propagation involves defining a probability distribution F over the labels for each node in the graph. For labeled nodes, this distribution reflects the true labels and the aim is to recover this distribution for the unlabeled nodes in the graph.

Consider a graph $G(V, E, W)$ with vertices V, edges E, and an $n \times n$ edge weight matrix $W =$

[1] The `religious_leader` sense of `moon` is due to Sun Myung Moon, a US religious leader.



(a) (b) (c)

Figure 1: Manifold Assumption [Belkin *et al.*, 2005]: Data lies on a manifold (a) and points along the manifold are locally similar (b).

$[w_{ij}]$, where $n = |V|$. The Label Propagation algorithm minimizes a quadratic energy function

$$\mathcal{E} = \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(F_i - F_j)^2 \qquad (1)$$

The general recipe for using random walks for classification involves constructing the graph Laplacian and using the pseudo-inverse of the Laplacian as a kernel (Xiao and Gutman, 2003). Given a weighted undirected graph, $G(V, E, W)$, the Laplacian is defined as follows:

$$L_{ij} = \begin{cases} d_i & \text{if } i = j \\ -w_{ij} & \text{if } i \text{ is adjacent to } j \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where $d_i = \sum_j w_{ij}$.

It has been shown that the pseudo-inverse of the Laplacian L is a kernel (Xiao and Gutman, 2003), i.e., it satisfies the Mercer conditions. However, there is a practical limitation to this approach. For very large graphs, even if the graph Laplacians are sparse, their pseudo-inverses are dense matrices requiring $O(n^2)$ space. This can be prohibitive in most computing environments.

5 Parallel Label Propagation

In developing a parallel algorithm for Label Propagation we instead take an alternate approach and completely avoid the use of inverse Laplacians for the reasons stated above. Our approach follows from the observation made from Zhu et al.'s (2003) Label Propagation algorithm:

Observation: In a weighted graph $G(V, E, W)$ with $n = |V|$ vertices, minimization of Equation (1) is equivalent to solving the following system of linear equations.

$$\sum_{(i,j) \in E} w_{ij}F_i = \sum_{(i,j) \in E} w_{ij}F_j \qquad (3)$$
$$\sum_{c \in \text{classes}(i)} F_i(c) = 1 \qquad \forall i, j \in V.$$

We use this observation to derive an iterative Label Propagation algorithm that we will later parallelize. Consider a weighted undirected graph $G(V, E, W)$ with the vertex set partitioned into V_L and V_U (i.e., $V = V_L \cup V_U$) such that all vertices in V_L are labeled and all vertices in V_U are unlabeled. Typically only a small set of vertices are labeled, i.e., $|V_U| \gg |V_L|$. Let F_u denote the probability distribution over the labels associated with vertex $u \in V$. For $v \in V_L$, F_v is known, and we also add a "dummy vertex" v' to the graph G such that $w_{vv'} = 1$ and $F_{v'} = F_v$. This is equivalent to the "clamping" done in (Zhu et al., 2003). Let V_D be the set of dummy vertices.

Algorithm 1: Iterative Label Propogation

repeat
  forall $v \in (V \cup V_D)$ **do**
    $F_v = \sum_{(v,u) \in E} w_{uv}F_v$
    Row normalize F_v.
  end
until *convergence or maxIterations*

Observe that every iteration of Algorithm 1 performs certain operations on each vertex of the graph. Further, these operations only rely on local information (from neighboring vertices of the graph). This leads to the parallel algorithm (Algorithm 2) implemented using the map-reduce model. Map-Reduce (Dean and Ghemawat, 2004) is a paradigm for implementing distributed algorithms with two user supplied functions "map" and "reduce". The map function processes the input key/value pairs with the key being a unique iden-

tifier for a node in the graph and the value corresponds to the data associated with the node. The mappers run on different machines operating on different parts of the data and the reduce function aggregates results from various mappers.

Algorithm 2: Parallel Label Propagation

```
map(key, value):
begin
    d = 0
    neighbors = getNeighbors(value);
    foreach n ∈ neighbors do
        w = n.weight();
        d += w * n.getDistribution();
    end
    normalize(d);
    value.setDistribution(d);
    Emit(key, value);
end

reduce(key, values): Identity Reducer
```

Algorithm 2 represents one iteration of Algorithm 1. This is run repeatedly until convergence or for a specified number of iterations. The algorithm is considered to have converged if the label distributions associated with each node do not change significantly, i.e., $\left|\left|\mathbf{F}_{(i+1)} - \mathbf{F}_{(i)}\right|\right|_2 < \epsilon$ for a fixed $\epsilon > 0$.

6 Label Propagation for Ranking

Graph ranking is applicable in a variety of problems in natural language processing and information retrieval. Given a graph, we would like to rank the vertices of a graph with respect to a node, called the pivot node or query node. Label Propagation and its variants (Szummer and Jaakkola, 2001; Zhu et al., 2003; Azran, 2007) have been traditionally used for semi-supervised classification. Our view of Label Propagation (via Algorithm 1) suggests a way to perform ranking on graphs.

Ranking on graphs can be performed in the Parallel Label Propagation framework by associating a single point distribution with all vertices. The pivot node has a mass fixed to the value 1 at all iterations. In addition, the normalization step in Algorithm 2 is omitted. At the end of the algorithm, the mass associated with each node determines its rank.

6.1 Connection to PageRank

It is interesting to note that Algorithm 1 brings out a connection between Label Propagation and PageRank (Page et al., 1998). PageRank is a random walk model that allows the random walk to "jump" to its initial state with a nonzero probability (α). Given the probability transition matrix $\mathbf{P} = [P_{rs}]$, where P_{rs} is the probability of jumping from node r to node s, the weight update for any vertex (say v) is derived as follows

$$v_{t+1} = \alpha v_t \mathbf{P} + (1 - \alpha) v_0 \tag{4}$$

Notice that when $\alpha = 0.5$, PageRank is reduced to Algorithm 1, by a constant factor, with the additional $(1 - \alpha)v_0$ term corresponding to the contribution from the "dummy vertices" V_D in Algorithm 1.

We can in fact show that Algorithm 1 reduces to PageRank as follows:

$$\begin{aligned} v_{t+1} &= \alpha v_t \mathbf{P} + (1 - \alpha) v_0 \\ &\propto v_t \mathbf{P} + \frac{(1 - \alpha)}{\alpha} v_0 \\ &= v_t \mathbf{P} + \beta v_0 \end{aligned} \tag{5}$$

where $\beta = \frac{(1-\alpha)}{\alpha}$. Thus by setting the edge weights to the dummy vertices to β, i.e., $\forall (z, z') \in E$ and $z' \in V_D, w_{zz'} = \beta$, Algorithm 1, and hence Algorithm 2, reduces to PageRank. Observe that when $\beta = 1$ we get the original Algorithm 1. We'll refer to this as the "β-correction".

7 Graph Representation

Since Parallel Label Propagation algorithm uses only local information, we use the adjacency list representation (which is same as the sparse adjacency matrix representation) for the graph. This representation is important for the algorithm to have a constant main memory requirement as no further lookups need to be done while computing the label distribution at a node. The interface definition for the graph is listed in Appendix A. Often graph data is available in an edge format, as <source, destination, weight> triples. We use another map-reduce step (Algorithm 3) to convert that data to the form shown in Appendix A.

8 Evaluation

We evaluate the Parallel Label Propagation algorithm for both ranking and semi-supervised classification. In ranking our goal is to rank the vertices of a graph with respect to a given node called the pivot/query node. In semi-supervised classification, we are given a graph with some vertices

Algorithm 3: Graph Construction

```
map(key, value):
begin
    edgeEntry = value;
    Node n(edgeEntry);
    Emit(n.id, n);
end

reduce(key, values):
begin
    Emit(key, serialize(values));
end
```

labeled and would like to predict labels for the remaining vertices.

8.1 Ranking

To evaluate ranking, we consider the problem of deriving lexical relatedness between terms. This has been a topic of interest with applications in word sense disambiguation (Patwardhan et al., 2005), paraphrasing (Kauchak and Barzilay, 2006), question answering (Prager et al., 2001), and machine translation (Blatz et al., 2004), to name a few. Following the tradition in previous literature we evaluate on the Miller and Charles (1991) dataset. We compare our rankings with the human judegments using the Spearman rank correlation coefficient. The graph for this task is derived from WordNet, an electronic lexical database. We compare Algorithm 2 with results from using geodesic similarity as a baseline.

As observed in Table 1, the parallel implementation in Algorithm 2 performs better than ranking using geodesic similarity derived from shortest path lengths. This reinforces the motivation of using random walks as described in Section 3.

Method	Spearman Correlation
Geodesic (baseline)	0.28
Parallel Label Propagation	**0.36**

Table 1: Lexical-relatedness results: Comparison with geodesic similarity.

We now empirically verify the equivalence of the β-corrected Parallel Label Propagation and PageRank established in Equation 4. To do this, we use $\alpha = 0.1$ in the PageRank algorithm and set $\beta = \frac{(1-\alpha)}{\alpha} = 9$ in the β-corrected Parallel Label Propagation algorithm. The results are seen in Table 2.

Method	Spearman Correlation
PageRank ($\alpha = 0.1$)	0.39
Parallel Label Propagation ($\beta = 9$)	**0.39**

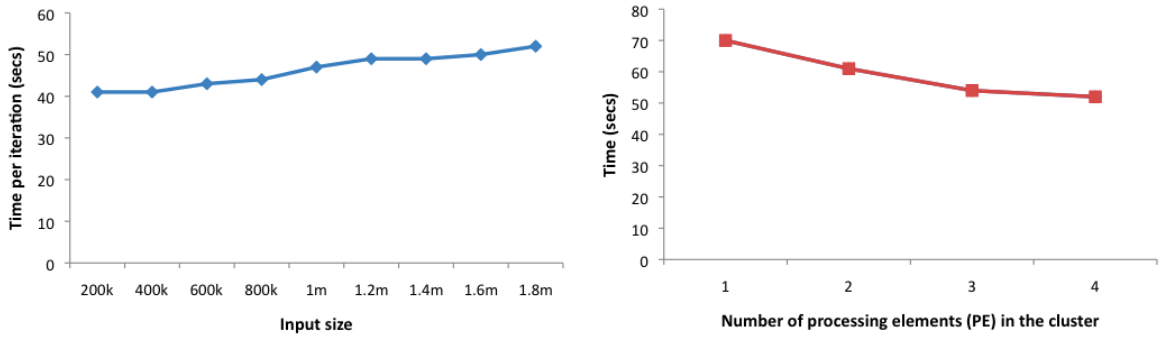
Table 2: Lexical-relatedness results: Comparision of PageRank and β-corrected Parallel Label Propagation

8.2 Semi-supervised Classification

Label Propagation was originally developed as a semi-supervised classification method. Hence Algorithm 2 can be applied without modification. After execution of Algorithm 2, every node v in the graph will have a distribution over the labels F_v. The predicted label is set to $\arg\max_{c \in \text{classes}(v)} F_v(c)$.

To evaluate semi-supervised classification we consider the problem of learning sentiment polarity lexicons. We consider the polarity of a word to be either positive or negative. For example, words such as *good, beautiful*, and *wonderful* are considered as positive sentiment words; whereas words such as *bad, ugly*, and *sad* are considered negative sentiment words. Learning such lexicons has applications in sentiment detection and opinion mining. We treat sentiment polarity detection as a semi-supervised Label Propagation problem in a graph. In the graph, each node represents a word whose polarity is to be determined. Each weighted edge encodes a relation that exists between two words. Each node (word) can have two labels: positive or negative. It is important to note that Label Propagation, and hence Algorithms 1&2, support multi-class classification but for the purpose of this task we have two labels. The graph for the task is derived from WordNet. We use the General Inquirer (GI)[2] data for evaluation. General Inquirer is lexicon of English words hand-labeled with categorical information along several dimensions. One such dimension is called valence, with 1915 words labeled "Positiv" (sic) and 2291 words labeled "Negativ" for words with positive and negative sentiments respectively. We used a random 20% of the data as our seed labels and the rest as our unlabeled data. We compare our results

[2] http://www.wjh.harvard.edu/∼inquirer/



(a) (b)

Figure 3: Scalability results: (a) Scaleup (b) Speedup

(F-scores) with another scalable previous work by Kim and Hovy (Kim and Hovy, 2006) in Table 2 for the same seed set. Their approach starts with a few seeds of positive and negative terms and bootstraps the list by considering all synonyms of positive word as positive and antonyms of positive words as negative. This procedure is repeated *mutatis mutandis* for negative words in the seed list until there are no more words to add.

Method	Nouns	Verbs	Adjectives
Kim & Hovy	34.80	53.36	47.28
Parallel Label Propagation	**58.53**	**83.40**	**72.95**

Table 3: Polarity induction results (F-scores)

The performance gains seen in Table 3 should be attributed to the Label Propagation in general as the previous work (Kim and Hovy, 2006) did not utilize a graph based method.

9 Scalability experiments

We present some experiments to study the scalability of the algorithm presented. All our experiments were performed on an experimental cluster of four machines to test the concept. The machines were Intel Xeon 2.4 GHz with 1Gb main memory. All performance measures were averaged over 20 runs.

Figure 3a shows scaleup of the algorithm which measures how well the algorithm handles increasing data sizes. For this experiment, we used all nodes in the cluster. As observed, the increase in time is at most linear in the size of the data. Figure 3b shows speedup of the algorithm. Speedup shows how well the algorithm performs with increase in resources for a fixed input size. In this case, we progressively increase the number of nodes in the cluster. Again, the speedup achieved is linear in the number of processing elements (CPUs). An appealing factor of Algorithm 2 is that the memory used by each mapper process is fixed regardless of the size of the graph. This makes the algorithm feasible for use with large-scale graphs.

10 Related Work

Historically, there is an abundance of work in parallel and distributed algorithms for graphs. See Grama et al. (2003) for survey chapters on the topic. In addition, the emergence of open-source implementations of Google's map-reduce (Dean and Ghemawat, 2004) such as Hadoop[3] has made parallel implementations more accessible.

Recent literature shows tremendous interest in application of distributed computing to scale up machine learning algorithms. Chu et al. (2006) describe a family of learning algorithms that fit the Statistical Query Model (Kearns, 1993). These algorithms can be written in a special summation form that is amenable to parallel speed-up. Examples of such algorithms include Naive Bayes, Logistic Regression, backpropagation in Neural Networks, Expectation Maximization (EM), Principal Component Analysis, and Support Vector Machines to name a few. The summation form can be easily decomposed so that the mapper can compute the partial sums that are then aggregated by a reducer. Wolfe et al. (2008) describe an approach to estimate parameters via the EM algorithm in a setup aimed to minimize communication latency.

The k-means clustering algorithm has been an archetype of the map-reduce framework with several implementations available on the web. In

[3]http://hadoop.apache.org/core

addition, the Netflix Million Dollar Challenge[4] generated sufficient interest in large scale clustering algorithms. (McCallum et al., 2000), describe algorithmic improvements to the k-means algorithm, called canopy clustering, to enable efficient parallel clustering of data.

While there is earlier work on scalable map-reduce implementations of PageRank (E.g., Gleich and Zhukov (2005)) there is no existing literature on parallel algorithms for graph-based semi-supervised learning or the relationship between PageRank and Label Propagation.

11 Conclusion

In this paper, we have described a parallel algorithm for graph ranking and semi-supervised classification. We derived this by first observing that the Label Propagation algorithm can be expressed as a solution to a set of linear equations. This is easily expressed as an iterative algorithm that can be cast into the map-reduce framework. This algorithm uses fixed main memory regardless of the size of the graph. Further, our scalability study reveals that the algorithm scales linearly in the size of the data and the number of processing elements in the cluster. We also showed how Label Propagation can be used for ranking on graphs and the conditions under which it reduces to PageRank. We evaluated our implementation on two learning tasks – ranking and semi-supervised classification – using examples from natural language processing including lexical-relatedness and sentiment polarity lexicon induction with a substantial gain in performance.

A Appendix A: Interface definition for Undirected Graphs

In order to guarantee the constant main memory requirement of Algorithm 2, the graph representation should encode for each node, the complete information about it's neighbors. We represent our undirected graphs in the Google's Protocol Buffer format.[5] Protocol Buffers allow a compact, portable on-disk representation that is easily extensible. This definition can be compiled into efficient Java/C++ classes.

The interface definition for undirected graphs is listed below:

[4] http://www.netflixprize.com

[5] Implementation available at http://code.google.com/p/protobuf/

```
package graph;

message NodeNeighbor {
   required string id = 1;
   required double edgeWeight = 2;
   repeated double labelDistribution = 3;
}

message UndirectedGraphNode {
   required string id = 1;
   repeated NodeNeighbor neighbors = 2;
   repeated double labelDistribution = 3;
}

message UndirectedGraph {
   repeated UndirectedGraphNode nodes = 1;
}
```

References

Arik Azran. 2007. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Micheal. Belkin, Partha Niyogi, and Vikas Sindhwani. 2005. On manifold regularization. In *Proceedings of AISTATS*.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proceeding of COLING*.

Cheng T. Chu, Sang K. Kim, Yi A. Lin, Yuanyuan Yu, Gary R. Bradski, Andrew Y. Ng, and Kunle Olukotun. 2006. Map-reduce for machine learning on multicore. In *Proceedings of Neural Information Processing Systems*.

Jeffrey Dean and Sanjay Ghemawat. 2004. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the symposium on Operating systems design and implementation (OSDI)*.

Christaine Fellbaum, editor. 1989. *WordNet: An Electronic Lexical Database*. The MIT Press.

D. Gleich and L. Zhukov. 2005. Scalable computing for power law graphs: Experience with parallel pagerank. In *Proceedings of SuperComputing*.

Ananth Grama, George Karypis, Vipin Kumar, and Anshul Gupta. 2003. *Introduction to Parallel Computing (2nd Edition)*. Addison-Wesley, January.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of HLT-NAACL*.

Michael Kearns. 1993. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*.

Soo-Min Kim and Eduard H. Hovy. 2006. Identifying and analyzing judgment opinions. In *Proceedings of HLT-NAACL*.

Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining (KDD)*, pages 169–178.

G. Miller and W. Charles. 1991. Contextual correlates of semantic similarity. In *Language and Cognitive Process*.

Larry Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The pagerank citation ranking: Bringing order to the web. *Technical report, Stanford University, Stanford, CA*.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2005. Senserelate::targetword - A generalized framework for word sense disambiguation. In *Proceedings of ACL*.

John M. Prager, Jennifer Chu-Carroll, and Krzysztof Czuba. 2001. Use of wordnet hypernyms for answering what-is questions. In *Proceedings of the Text REtrieval Conference*.

M. Szummer and T. Jaakkola. 2001. Clustering and efficient use of unlabeled examples. In *Proceedings of Neural Information Processing Systems (NIPS)*.

Jason Wolfe, Aria Haghighi, and Dan Klein. 2008. Fully distributed EM for very large datasets. In *Proceedings of the International Conference in Machine Learning*.

W. Xiao and I. Gutman. 2003. Resistance distance and laplacian spectrum. *Theoretical Chemistry Association*, 110:284–289.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning (ICML)*.