MapReduce approach to relational influence propagation in complex networks

Tomasz Kajdanowicz · Wojciech Indyk ·
Przemysław Kazienko

Received: 12 March 2012 / Accepted: 6 August 2012 / Published online: 5 September 2012
© The Author(s) 2012. This article is published with open access at Springerlink.com

Abstract The relational label propagation problem for large data sets using MapReduce programming model was considered in the paper. The method we propose estimates class probability in relational domain in the networks. The method was examined on large real telecommunication data set. The results indicated that it could be used successfully to classify networks' nodes and, thanks to that, new offerings or tariffs might be proposed to customers who belong to other providers. Moreover, basic properties of relational label propagation were examined and reported.

Keywords MapReduce · Relational influence propagation · Classification in networks · Label propagation · Collective classification · Relational learning

1 Introduction

Relations between objects in various systems are commonly modelled by networks. For instance, those are hyperlinks connecting Web pages, paper citations, conversations via e-mail or social interaction in social portals. Network models are further a base for different types of processing and analyses. One of them is node classification (labelling of the nodes in the network). Node classification has a deep theoretical background; however, due to new phenomena

appearing in artificial environments like social networks on the Internet, the problem of node classification is being recently re-invented and re-implemented.

Nodes may be classified in networks either by inference based on known profiles of these nodes (regular concept of classification) or based on relational information derived from the network. This second approach utilizes information about connections between nodes (structure of the network) and can be very useful in assigning labels to the nodes being classified. For example, it is very likely that a given Web page x is related to sport (label *sport*), if x is linked by many other Web pages about sport.

One of the strong motivations to use the relational model is the ability for modelling a relationship between correlated observations. There is an intuitive desire to use information about one object, to reach conclusions (information) with other related objects. For example, a Web network should be able to propagate information about the topic of the document to other documents that contain links to it, and the propagation, following the successive degree of the neighbourhood, could indicate a network of topics of documents. In this paper, an algorithm is proposed using the above process in accordance with the principle of relational influence propagation [1–3].

Hence, a form of collective classification should be provided, with simultaneous decision making on every node's label rather than classifying each node separately. Such approach allows taking into account correlations between connected nodes, which deliver usually undervalued knowledge.

Moreover, a rising trend of data explosion in transactional systems requires more sophisticated methods to analyse enormous amounts of data. There is a huge need to process large data in parallel, especially in complex analyses like collective classification.

T. Kajdanowicz (✉) · W. Indyk · P. Kazienko
Faculty of Computer Science and Management,
Wrocław University of Technology, Wrocław, Poland
e-mail: tomasz.kajdanowicz@pwr.wroc.pl

W. Indyk
e-mail: wojciech.indyk@pwr.wroc.pl

P. Kazienko
e-mail: przemyslaw.kazienko@pwr.wroc.pl

MapReduce approach to collective classification which is able to perform processing on huge data is proposed and examined in the paper. The proposed method is able to handle massive data thanks to parallel and distributed processing of Relational Influence Propagation algorithm. Section 2 covers related work, while in Sect. 3 a proposal of MapReduce approach to label propagation in the network appears. Section 4 contains description of the experimental setup and the obtained results. The paper is concluded in Sect. 5.

2 Related work

2.1 Relational influence propagation

Collective classification problems may be solved using two main approaches: within-network and across-network inference. Within-network classification, for which training entities are connected directly to entities whose labels are to be classified, stays in contrast to across-network classification, where models learnt from one network are applied to another similar network [4]. Across-network classification can be understood as a transfer learning approach that accomplishes relational classification [5]. Overall, the networked data have several unique characteristics that simultaneously complicate and provide leverage to learning and classification.

One of the methods using within-network approach is relational influence propagation. The main idea is based on iterative propagation of the known labels in a network to the non-labelled nodes [3]. The method was originally derived from enhanced hypertext categorization [1]. Due to the profile of the method, its accuracy strictly depends on sampling of the training set. This is due to the fact that several false interference phases (propagation of initial information) can lead to a "snowball" effect [3]. Generally, the final result of such a class of algorithms depends on network sampling methods, where independence and identity of distribution assumption made by a standard classification and clustering models are inappropriate in the complex relational domain [6].

Among others, several statistical relational learning (SRL) techniques were introduced, including probabilistic relational models, relational Markov networks and probabilistic entity-relationship models [7, 8]. Two distinct types of classification in networks may be distinguished: based on collection of local conditional classifiers and based on the classification stated as one global objective function. The most known implementations of the first approach are iterative classification (ICA) and Gibbs sampling algorithm (GS), whereas example of the latter are loopy belief propagation (LBP) and mean field relaxation labelling (MF) [9]. In general, the second group of algorithms

represents the idea of label propagation. Label propagation starts with labelled nodes and each of these nodes propagate its known label to its neighbours (nodes without labels) and the process is repeated until convergence.

2.2 MapReduce programming model

Large data processing requires parallel computational model and parallel execution. To address these requirements, MapReduce programming model may be incorporated. MapReduce provides means for data processing derived from functional language [10] and is dedicated to solving complex and distributable problems [11–13]. It utilizes a large number of nodes, hereafter collectively referred to as a cluster. MapReduce breaks the processing into two consecutive phases: the *Map* and the *Reduce* phases.

The first phase commences with data splitting into separate chunks. According to input file configuration, each data chunk must meet the requirement of the $\langle key, value \rangle$ format. Data are then processed by a *Map* function, which takes this input pair. Assuming independence between each Map function, the processing may be conducted in a parallel manner. Each computational node in the cluster may perform multiple number of Map functions. The aim of the Map function is propagation of the processed input data to further phase using again the $\langle key, value \rangle$ format. Before the next step of processing is accomplished, all results from Map functions are sorted (called shuffle in MapReduce). This allows to split the data according to the value of *key* into separate data chunks, one chunk for each value of *key*. Then particular data chunk, provided in $\langle key, list(value) \rangle$ format, is processed in a separate reducer, called *Reduce* phase. Reducer implements final processing and propagates $\langle key, value \rangle$ pairs as results. The results are saved as an output of processing. Usually, each reducer outputs one pair.

Both, Map and Reduce phases, need to be specified and implemented by the user [14, 15]. The aforementioned process is presented in Fig. 1.

Thanks to initial data splitting, the MapReduce programming model is able to process large data sets, which is impossible in other models. The most common open-source implementation of MapReduce is Apache Hadoop library [16]. Apache Hadoop is a framework that allows distributed processing of large data sets. Such processing can be performed across clusters of computers offering local computation and storage. The architectural solutions of Hadoop deliver high availability and robustness, not only because of hardware properties but also due to failures handling in the application layer. Moreover, data replication facilitates retaining fault-tolerant and highly reliable computational environment. The single MapReduce phase in Hadoop is called *Job*. The *Job* consists of the map method, reduce method, data *inputFiles* and configuration.

Fig. 1 The MapReduce programming model

3 Collective classification by means of relational influence propagation using MapReduce

The most common way to utilize the information of labelled and unlabelled data is to construct a graph from data and perform a Markov random walk on it. The idea of Markov random walk has been used multiple times [17–19] and involves defining a probability distribution over the labels for each node in the graph. In case of labelled nodes, the distribution reflects the true labels. The aim then is to recover this distribution for the unlabelled nodes. Using such a label propagation approach allows performing classification based on relational data.

In this paper we assume, similarly as proposed in [3], that iterative classification algorithm works on relational data structure that consists of nodes and ties between them. In our approach, relation influence propagation is realized iteratively and is based on physical modelling of harmonic energy minimization presented in [20]. The main idea of the algorithm is to maintain the potential balance between nodes in a network. Assuming that the potential represents the label probability, labels of nodes in this model are distributed by weighted arcs in a graph structure. Our proposal has been additionally fed with improved conception of *dummy nodes* initially proposed in [18]. Due to the fact that propagation of information about known labels needs to preserve the balance in the whole network, it is allowed to change the probability of label even in initially chosen informative nodes. Due to the impact of incoming connections to these nodes, their labels may eventually be changed, which is not at all desirable. Therefore, there is a need keep the original labels in such nodes and this is achieved by *dummy node*.

Let $G(V, E, W)$ denote a graph with vertices V , arcs E and an $n \times n$ arcs weight matrix W . According to [18] in a weighted graph $G(V, E, W)$ with $n = |V|$ vertices, label propagation may be solved by linear Eqs. 1, 2.

$$\forall i, j \in V \sum_{(i,j) \in E} w_{ij} P_i = \sum_{(i,j) \in E} w_{ij} P_j \quad (1)$$

$$\forall i \in V \sum_{c \in \text{classes}(i)} P_i = 1 \quad (2)$$

where $w_{ij} \in W$ and P_i denotes the class likelihood for the i th node.

Let us assume that the set of nodes V is partitioned into labelled V_L and unlabelled V_U vertices, $V = V_L \cup V_U$. Let P_u denote the probability distribution over the labels associated with vertex $u \in V$. For each node $v \in V_L$, for which P_v is known, a dummy node v' is inserted such that $w_{v'v} = 1$ and $P_{v'} = P_v$. This operation is equivalent to ‘clamping’ discussed in [18]. Let V_D be the set of dummy nodes. Then the solution of Eqs. 1 and 2 can be performed according to Iterative Label Propagation, Algorithm 1.

Algorithm 1 The pseudo code of Iterative Label Propagation algorithm

```

1: repeat
2:   for all  $v \in (V \cup V_D)$  do
3:      $P_v = \frac{\sum_{(u,v) \in E} w_{uv} P_u}{\sum_{(u,v) \in E} w_{uv}}$ 
4:   end for
5: until convergence
    
```

The introduced *dummy nodes* are therefore artificially added nodes to a graph and are connected with labelled nodes. Each labelled node in the network has one *dummy node* added. A particular *dummy node* has an original label of its labelled parent. During the iterative calculation of the propagation algorithm, the label probability value might not change as the appended *dummy nodes* are connected with arc directed from *dummy node* to the original labelled node; see Fig. 2. This is a contrary proposal for preserving original values of labels to the one presented in [20], where arcs between a particular dummy node and the corresponding labelled one were directed from the real node to the dummy. As proposed in [20], no prevention for keeping the original label is preserved as only incoming arcs may influence the class probability. We propose to model the direction of influence: from a dummy to a corresponding

Fig. 2 Two situations of network balancing using Iterative Label Propagation: a *graph* with unable to balance structure and a *graph* with added *dummy nodes* (black nodes) that may be balanced

labelled node. Therefore, the dummy node keeps the true label of the original node according to Eq. 3.

$$P_i = \frac{w_{i'i'}F_{i'} + \sum_{(j,i) \in E} w_{ji}P_j}{w_{i'i} + \sum_{(j,i) \in E} w_{ji}}$$

$$w_{i'i} = 1; \sum_{(j,i) \in E} w_{ji} = 1 \quad (3)$$

$$P_i = \frac{P_{i'} + \sum_{(j,i) \in E} w_{ji}P_j}{2}$$

where P_i denotes the probability of classes for node i , and i' denotes *dummy node* corresponding to node i . When $P_{i'} = 1$ or $P_{i'} = 0$, which is a common initial situation in crisp labels classification problem, the probability P_i will remain $P_i \approx P_{i'}$. It means, the initially known label 1 or 0 will not be changed during the algorithm iterations.

In general, using the proposed *dummy nodes*, we are able to avoid the situation where the network is not balanced. As an example, please consider Fig. 2. In the first settlement with two nodes and two arcs between them, it is impossible to balance the network. Each iteration of the algorithm will cause a shift of probabilities (relevance of labels) between both nodes and no stabilization will be achieved. What is more in this situation, we change the originally known labels of nodes. When adding *dummy nodes* to the second situation in Fig. 2, it is possible to balance the network and it will be reached in a few iterations.

As it can be observed, at each iteration of Iterative Label Propagation, certain operations on each of the nodes are performed. These operations are calculated basing on local information only, namely the node's neighbourhoods. This fact can be utilized in a parallel version of the algorithm; see Algorithm 2.

Algorithm 2 The pseudo code of MapReduce approach to Iterative Label Propagation algorithm

```

1: map < node; adjacencyList >
2: for all n ∈ adjacencyList do
3:   propagate< n; node.label, n.weight >
4: end for
1: reduce < n, list(node.label, weight) >
2: propagate< n,  $\frac{\sum \text{node.label} \cdot \text{weight}}{\sum \text{weight}}$  >

```

The MapReduce version of Iterative Label Propagation algorithm consists of two phases. The Map phase gets all labelled and dummy nodes and propagates their labels to all nodes in adjacency list taking into account arc weights between nodes. The Reduce phase calculates new label for each node with at least one labelled neighbour. Reducers calculate new label for nodes based on a list of labelled neighbours and relation strength between nodes (weight). The final result, namely a new label for a particular node, is computed as the weighted sum of labels' probabilities from the neighbourhood.

4 Experiments and results

To present the profile of the proposed method, the telecommunication network was built with a 3 months' history of phone calls from a leading European telecommunication company. The original data set consisted of about 500,000,000 phone calls and more than 16 million unique users.

All communication facts (phone calls) were performed using one of 38 tariffs, of 4 types each. In order to establish a binary classification task, only two types of phone calls were extracted and utilized in the experiments.

Users were labelled with class conditional probability of tariffs, namely, the sum of outgoing phone call durations in a particular tariff was divided by the summarized duration of outgoing calls. Eventually, the final data set consisted of 12,787,114 users.

To test the method, an initial sampling of known labels in the network and unknown label to be populated from the former ones was required. Therefore the data set was partitioned into two subsets: training set and testing set. The training set consisted of 7,163,227 users who made calls in the first and second month. The testing set consisted of 1,522,759 users who made calls only in the third month. Both subsets were disjoint. Additionally, there were also 5,566,376 unknown users in the network who did not make any calls and they can be regarded as clients who belong to other external telecommunication providers.

To build a relational structure of the data required in the proposed algorithm, detailed billing information was used

and transformed to the network. The users' network was calculated using Eq. 4 to obtain a connection strength between particular users:

$$w_{ij} = \frac{2 \cdot d_{ij}}{d_i + d_j} \quad (4)$$

where d_{ij} denotes the summarized duration of calls between user i and j , d_i the summarized duration of i th outgoing calls and d_j the summarized duration of j th incoming calls. The consequently obtained network was composed of 67,184,654 weighted arcs between the aforementioned users.

The goal of the first experiment was to predict class conditional probability of tariff for unlabelled users. Using previously selected nodes for training, the MapReduce Iterative Label Propagation method was employed. After 35 iterations, 12,787,114 labels were reached. However only 399,075 labels from the testing set could be evaluated. This revealed that some of the nodes from the testing set were not reachable in label propagation—these nodes did not contain incoming connection thanks to which the label could be propagated. It means, 1,123,684 nodes in the testing set contained outgoing connections (therefore

have labels; see 4) but do not have any incoming connection. This situation is illustrated in Fig. 3. Starting from the training set, labels are propagated to connected nodes in the testing set and further to nodes belonging to an external provider. However, there exists a black node in the testing set that cannot be accessed in the iterative process as it has no incoming arcs. Thus, this node will remain unlabelled. As a consequence, some nodes from the external provider might not be similarly unlabelled. In the experiment, there were 341,564 nodes unable to be labelled (see black node in Fig. 3) in the external provider set.

The Iterative Label Propagation algorithm was implemented in MapReduce programming model. It consists of six Jobs, each accomplishing map and reduce phases. A detailed description of Jobs is presented in Table 1. The convergence criterion in the algorithm has been controlled by ϵ coefficient, which was a threshold for the change of class conditional probability for each node. The algorithm was iterating until these changes were greater than ϵ . In all experiments, ϵ was set to 0.001. The algorithm was implemented in Hadoop distributed system [16] and was run on cluster computer composed of six nodes: one master and five slaves machines. Each machine contained 8 CPU,

Fig. 3 Distinct types of nodes in Iterative Label Propagation: labelled and unlabelled, training, testing and external provider ones

Table 1 MapReduce jobs implemented in the algorithm

Job name	Job description
adjacencyList	The job takes arc list as an input and returns an adjacency list for all nodes
dummyAdjListAndLabels	The job creates a list of dummy nodes with labels according to algorithm [7] and updates an adjacency list by newly created arcs from dummy nodes
mergeAdjListAndLabel	The job merges a list of node labels with adjacency list resulting in collective classification input
collectiveClassification	The job processes collective classification data according to algorithm and results in a new label list
singleLabelsComparison	The job results in absolute difference of class conditional probability of labels from actual iteration and previous iteration
allLabelComparison	The job returns maximal difference of input list (absolute difference of class conditional probability)

Table 2 Average execution time in [s] for particular map–reduce jobs

Job name	Average time	SD
adjacencyList	593	77
dummyAdjListAndLabels	174	21
mergeAdjListAndLabel	399	55
collectiveClassification	412	81
singleLabelsComparison	133	22
allLabelComparison	78	7

20 GB RAM and 160 GB hard disk. The master configuration was set to two mappers and two reducers, and slaves machines were configured with three mappers and three reducers. The experiment was organized to examine the average computational time devoted to each of the map–reduce steps as well as to measure the accuracy.

The computation time was measured in 120 iterations and is presented in Table 2. On average, the time required

for computation of one iteration in a given experimental scenario was about 17 min. Moreover during first 35 iterations of the algorithm, the number of unstable (in terms of ϵ threshold) nodes was measured. Additionally, F measure was evidenced.

As observed in Fig. 4, after about 15 iterations of the algorithm, the number of unstable nodes stabilized around 1,000 maintaining the decreasing trend. Just after the sixth iteration, there was no improvement in F measure that in general was very low. It revealed that either the Iterative Label Propagation algorithm was not a good model for tariff modelling in the telecommunication data or the assumed sampling method used to derive the training set was inappropriate.

Therefore, there were other sampling approaches examined in order to obtain the training set. We examined a sampling method based on standard measure of the node's degree. Namely, for each node $v \in V$, a number of outgoing connections was calculated $deg^+(v)$, which is called the out-degree measure. According to this structural

Fig. 4 Number of nodes with not stabilized labels and F measure obtained after a particular number of algorithm iterations**Fig. 5** Number of nodes with not stabilized labels after a particular number of algorithm iterations in three distinct sampling methods $S3$, $S6$ and $S9$

Fig. 6 F measure for both binary classes obtained after a particular number of algorithm iterations in three distinct sampling methods $S3$, $S6$ and $S9$

Table 3 Number of nodes with not stabilized labels after a particular number of algorithm iterations in three distinct sampling methods $S3$, $S6$ and $S9$

Iteration	Sampling method		
	$S3$	$S6$	$S9$
1	8,685,986	8,685,986	8,685,986
2	7,782,653	7,988,777	8,005,878
3	7,145,317	7,629,938	7,803,992
4	6,779,987	7,413,902	7,659,189
5	5,133,932	6,502,205	7,037,190
6	4,563,753	6,103,953	6,765,670
7	2,117,914	3,681,615	4,970,453
8	1,798,344	3,212,133	4,557,691
9	910,654	1,543,663	2,216,317
10	755,623	1,371,551	1,940,987
11	323,697	596,492	892,399
12	255,691	535,246	789,330
13	116,913	215,200	367,545
14	89,463	192,408	354,833
15	14,652	53,193	188,274
16	12,632	48,321	175,010
17	10,231	39,376	157,415
18	9,527	37,523	149,431

property, the original data set was partitioned into the training and testing set, as in the formula presented in Eq. 5.

$$\forall v \in V_{\text{labelled}} \begin{cases} v \in \text{TrainingSet}, \deg^+(v) \geq d \\ v \in \text{TestSet}, \deg^+(v) < d \end{cases} \quad (5)$$

In such a situation, nodes with out-degree higher or equal to d were assigned to training set, and the rest to the test set.

To examine the proposed sampling method, three distinct scenarios of sampling were designed. As the method

is based on out-degree structural measure, three distinct d thresholds for this measure were established: 3, 6 and 9. Afterwards, they were used in sampling in $S3$, $S6$ and $S9$, respectively. The values of d were selected based on the histogram of the out-degree measure. The results obtained in the $S3$, $S6$ and $S9$ experiments are presented in Figs. 5 and 6.

According to the results presented in Fig. 5 and Table 3, the fastest converging was the scenario with $S3$ sampling. The F measure obtained in all three scenarios (results gathered in Table 4) was dramatically higher than previously reported (see Fig. 6 against Fig. 4). Therefore while using the second proposed approach to the sampling method, we were able to obtain quite satisfactory results of nodes classification. It can be observed that the $S3$ sampling method provides the best results for class 1 among other sampling methods. This is due to the fact that the training and testing data set is constructed according to Eq. 5. The network used in experimentation had a power law distribution, and sampling such a graph worked in the way that the smaller the value of d , the bigger was the training data set and smaller the test set. As a greater number of nodes with known labels in the graph makes the within-network classification easier, $S3$ sampling provided the best results.

5 Conclusions

The problem of collective classification using MapReduce programming model and relational influence propagation approach was considered in the paper. We examined the algorithm behaviour in a large network using a parallel environment that can perform complicated calculation using large data sets. The proposed method was examined on real data set in the telecommunication domain. The

Table 4 *F* measure obtained after a particular number of algorithm iterations in three distinct sampling methods *S3*, *S6* and *S9*

Iteration	<i>S3</i>		<i>S6</i>		<i>S9</i>	
	Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
1	0.9379	0.8198	0.9238	0.8697	0.9163	0.8959
2	0.9648	0.8976	0.9576	0.9298	0.9561	0.948
3	0.956	0.8692	0.9426	0.9025	0.9371	0.9234
4	0.9592	0.8787	0.9486	0.913	0.9456	0.9342
5	0.9575	0.8732	0.9451	0.9065	0.9405	0.9275
6	0.9582	0.8752	0.9466	0.909	0.9428	0.9305
7	0.9578	0.8739	0.9455	0.9071	0.9412	0.9284
8	0.958	0.8744	0.946	0.9079	0.942	0.9294

results indicated that it could be used to classify network's users in order to propose new offerings or tariffs to them. Experiments revealed that using various network sampling methods, we were able to improve classification accuracy, and probably the problem of network sampling is one of the most important in learning for relational domain.

Further experimentation will consider a comparison of the presented method with other approaches. Moreover, further studies on much larger data sets will be conducted. Additionally, the data set sampling problem will be examined according to nodes' attributes.

Acknowledgments This work was partially supported by The Polish National Center of Science the research project 2011–2012, 2011–2014 and Fellowship co-financed by The European Union within The European Social Fund. The authors are grateful to Wrocław Networking and Supercomputing Center for granting access to the computing infrastructure built in the project No. POIG.02.03.00-00-028/08 “PLATON - Science Services Platform”.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

- Chakrabarti S, Dom BE, Indyk P (1998) Enhanced hypertext categorization using hyperlinks. In: Proceedings of SIGMOD-98, ACM international conference on management of data, pp 307–318
- Slattery S, Mitchell T (2000) Discovering test set regularities in relational domains. In: Proceedings of the seventeenth international conference on machine learning, pp 895–902
- Neville J, Jensen D (2000) Iterative classification in relational data. In: Proceedings of AAAI-2000 workshop on learning statistical models from relational data, pp 13–20
- Jensen D, Neville J, Gallagher B (2004) Why collective inference improves relational classification. In: Proceedings of 10th ACM SIGKDD international conference on knowledge discovery and data mining, pp 593–598
- Kajdanowicz T, Plamowski S, Kazienko P, Indyk W (2012) Transfer learning approach to debt portfolio appraisal. Lecture Notes Comp Sci 7209:46–55
- Taskar B, Segal E, Koller D (2001) Probabilistic clustering in relational data. In: Proceedings of the seventeenth international joint conference on artificial intelligence, pp 870–876
- Desrosiers C, Karypis G (2009) Within-network classification using local structure similarity. Lecture Notes Comp Sci 5781: 260–275
- Knobbe A, deHaas M, Siebes A (2001) Propositionalisation and aggregates. In: Proceedings of Fifth European Conference on principles of, data mining and knowledge discovery, pp 277–288
- Kramer S, Lavrac N, Flach P (2001) Propositionalization approaches to relational data mining. In: Dezeroski S (ed) Relational data mining, Springer, Berlin, pp 262–286
- White T (2009) Hadoop: the definitive guide, O'Reilly, Sebastopol
- Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
- Dean J, Ghemawat S (2010) MapReduce: a flexible data processing tool. Commun ACM 53(1):72–77
- He Q, Du C, Wang Q, Zhuang F, Shi Z (2011) A parallel incremental extreme SVM classifier. Neurocomputing 74(16): 2532–2540
- Ekanayake J, Pallickara S, Fox G (2008) MapReduce for data intensive scientific analyses. In: Proceedings of 4th IEEE international conference on eScience, pp 277–284
- Dean J, Ghemawat S (2004) Mapreduce: simplified data processing on large clusters. In: Proceedings of 6th conference on symposium on operating systems design & implementation, pp 10–24
- Hadoop official web site. <http://hadoop.apache.org>. Accessed 02 June 2012
- Szummer M, Jaakkola T (2001) Clustering and efficient use of unlabeled examples. In: Proceedings of neural information processing systems
- Zhu X, Ghahramani Z, Lafferty J (2003) Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the international conference on machine learning, pp 912–919
- Azran A (2007) The rendezvous algorithm: multiclass semi-supervised learning with markov random walks. In: Proceedings of international conference on machine learning, pp 49–56
- Rao D, Yarowsky D (2009) Ranking and semi-supervised classification on large scale graphs using map-reduce. In: Proceedings of the 4th international joint conference on natural language processing of the Asian Federation of Natural Language Processing, pp 58–65