Weakly supervised graph-based methods for classification

Dragomir R. Radev
University of Michigan
Ann Arbor, MI 48109-1092

Abstract

We compare two weakly supervised graph-based classification algorithms: spectral partitioning and tripartite updating. We provide results from empirical tests on the problem of number classification. Our results indicate (a) that both methods require minimal labeled data, (b) that both methods scale well with the number of unlabeled examples, and (c) that tripartite updating outperforms spectral partitioning.

1 Introduction

Information extraction (IE) systems analyze unrestricted text in order to extract information about pre-specified types of events, entities or relationships. Traditionally, IE systems [9, 4, 7] have focused on the extraction and classification of entities into major categories like people, places, organizations, numbers, and dates. Many applications such as Question Answering (QA) [17, 13] require much finer grained entity categories (up to 100 and more phrase types overall) to identify candidate phrases for answers to factual questions. In such applications, high-level classification is only partially helpful.

In this paper, we consider the problem of number classification which has been somehow overlooked in the literature on IE. Number classification is particularly important in question answering systems. There are more than a dozen types of numbers which can be used to answer different question types (e.g., "What is the temperature in Phoenix?", "What year did Columbus reach America?", "What is the value of the Dow Jones index?", etc). We will compare two types of graph-based algorithms for weakly supervised classification, and empirically evaluate their performance on number classification. Our machine learning approaches are based on a minimal amount of supervision with only a small number $O(1)$ of la-

beled examples. Such algorithms are known in the literature as weakly supervised algorithms.

2 Related work

2.1 Weakly-supervised learning

One of the earliest papers on bootstrapping for NLP problems, [21] presents an unsupervised learning algorithm for word sense disambiguation that, when trained on unannotated English text, rivals the performance of supervised techniques that require time-consuming hand annotations. The algorithm is based on two powerful constraints – that words tend to have one sense per discourse and one sense per collocation – exploited in an iterative bootstrapping procedure. Tested accuracy exceeds 96%.

Blum and Mitchell [5] introduced co-training for the problem of classifying Web pages based on two views including different types of features (one based on the words in the pages themselves and another on the words on the hyperlinks of the pages pointing to them). Their main contribution is to show how two views can iteratively train each other to label a set of data.

Collins and Singer [7] discuss the use of unlabeled examples for the problem of named entity classification. They develop a technique that uses only 7 manually labeled "seed" examples to classify entities into three classes plus "other". Their approach works because given a particular instance to classify, many features correlate with any particular class and one can thus iteratively augment the set of features associated with a given class. Two algorithms are presented. The first method uses a decision list algorithm similar to that of [21], with modifications motivated by [5]. The second algorithm extends ideas from boosting algorithms, designed for supervised learning tasks, to the framework suggested by [5].

Recent theoretical results can be found in [1], who refines the analysis of co-training, defines and evaluates a new co-training algorithm that gives a theoretical justification for the Yarowsky algorithm [21], and shows that co-training and the Yarowsky algorithm are based on different independence assumptions.

Nigam et al. [15] show that the accuracy of text classifiers can be improved by adding large collections of unlabeled documents. The authors use an algorithm based on a combination of Expectation Maximization (EM) and Naive Bayes. The initial classifier is trained on labeled data and then used to label the unlabeled set. The next classifier uses the large pool as part of its training process. After some iterations, the algorithm converges. Some of the improvements on this basic algorithm, proposed by Nigam et al. include adding a weighting factor to determine the contribution of the unlabeled data and the use of multiple mixture components

for each class. The paper reports a reduction in classification error on three real-world tasks by up to 30%.

Collins and Singer [7] also investigate EM as a weakly supervised learning algorithm for the application of named entity classification. The performance was shown to be not as good as the Co-boosting algorithm proposed by the authors.

2.2 Graph-based classification and clustering

Graph-based methods for clustering and classification have existed for decades. A classic example is graph partitioning (see e.g., [11]). That algorithm is used to identify groups of nodes in a graph that are more strongly connected internally than to each other. Kernighan and Lin's method is based on breadth first traversal of a graph G and splits it into two components G_1 and G_2 such that $G = G_1 \cup G_2$ and $|G_1| = |G_2| = |G|/2$ such that the cost of the partitioning, which is equal to the number of edges that cross the partitioning ($C = |E(G_1, G_2)|$), is minimal.

Other techniques for graph partitioning are based on the spectrum (set of all eigenvectors) of the graph. Spectral partitioning (a.k.a. bisection) uses the Laplacian of a graph G. The Laplacian is symmetric and the values of all rows and columns add up to zero. The second smallest eigenvalue of the Laplacian, $\lambda_2(L(G))$, is known as the algebraic connectivity of the graph. The vector corresponding to it is called the Fiedler vector. If a graph G consists of two subgraphs G_1 and G_2 such that there are relatively few edges *between* G_1 and G_2 compared to the number of edges *within* each of them, then the Fiedler vector is effectively a two-class classifier. If f_i and f_j are two components of the Fiedler vector, then

$f_i * f_j \geq 0 \iff$ elements i and j of G correspond to the same subgraph (G_1 or G_2).

while

$f_i * f_j \leq 0 \iff$ elements i and j of G correspond to different subgraphs.

Graph-based partitioning methods can in general be applied to more than 2-way classification, although their accuracy is not very high if the underlying data differ significantly from the assumed distribution. One such method, recursive spectral bisection, is described in detail in [16].

A bipartite graph consists of two components of differing functionality. Edges exist only across components and not within a single component. Bipartite graphs are very popular in information retrieval and social network analysis. For example, one of the subgraphs can represent a set of documents and another, a set of terms in them or one of the components may be people and the other – the clubs in which they belong. Bipartite graphs are a very useful representation for classification problems. Several incarnations of methods based on bipartite graphs exist. Kleinberg's HITS algorithm [12] models *hubs* (Web pages that contain a lot of pointers

to important pages) and *authorities* (the important pages that are pointed to by the hubs). For example, a list of bookmarks on sports is a hub while the home pages of sports organizations and teams such as FIFA or Manchester United are authorities. The HITS algorithm uses an iterative method to compute the hub and authority scores of each page. In his model, H is the vector of the hub scores and A is the vector of the authority scores. The iterative process updates H and A in turn as follows: $A = G^T H$ and $H = GA$. This process converges to the stationary values of H and A.

In [3], Beeferman and Berger describe a method for analyzing user transactions on an Internet search engine to discover clusters of queries and URLs. Their model uses information about each user query as well as the document that the user clicked on from the list presented by the search engine. In this case, one of the components of the bipartite graph corresponds to the queries and the other one to the URLs. Beeferman and Berger apply an agglomerative clustering method to identify groups of related queries and groups of related pages. This method doesn't use any information about the textual content of the queries or pages and instead, makes all of its decisions based on the link information alone.

In [22], the authors propose a method for bipartite graph clustering that is based on the singular value decomposition (SVD) of the associated edge weight matrix of the bipartite graph. They apply their technique successfully on document clustering.

[23] describe a classification method based on the Gaussian random field model. The represent labeled and unlabeled data as vertices in a weighted graph with edge weights representing the similarity between data instances. They apply belief propagation methods to identify the labeled node that is closest based on the graph topology to a given unlabeled instance. Results on digit classification are very promising.

[14] present a simple spectral clustering algorithm implemented in a few lines of matlab. They analyze the algorithm based using matrix perturbation theory and determine the conditions under which it can be expected to do well in theory.

In Natural Language Processing, [6] describe an application of spectral clustering [14] to the problem of unsupervised clustering of German verbs. They present results comparing the output of the spectral algorithm to a gold standard.

Vert and Kanehisa [20] present an algorithm to extract features from high-dimensional gene expression profiles, based on a graph that indicates which genes are known to participate together in reactions in metabolic pathways.

To classify a large number of unlabeled examples, [19] start from a small number of labeled examples and implement a Markov random walk over the unlabeled examples. Results are shown on synthetic examples and text classification problems.

2.3 Numbers

Number classification has been traditionally overlooked in the NLP literature. A recent paper [18] discusses numbers in the context of non-standard "words" (NSWs) which include time expressions, dates, currency amounts, abbreviations, and acronyms. They indicate that such words are actually quite common in textual documents and furthermore (coming from a speech perspective) there are no standard approaches to generate pronunciations for them automatically. Numerous Question Answering papers (e.g., [17, 13]) show that identifying and correctly labeling numerical expressions can significantly help question answering. [2] also point out the pervasiveness of numerical expressions on Web pages and indicate their importance in document retrieval. More specifically, Agrawal et al. focus on the problem of retrieving documents with particular attribute-value pairs (e.g., power=660mW, speed=18ns, etc.) without having the attributes for each number annotated. They claim that the distributions of numbers for each possible attribute overlap very little and that one can infer the correct attribute with reasonable accuracy purely by looking at the number itself.

3 Number classification

Our goal is to classify numbers (integers) automatically extracted from a text corpus (the APW section of the AQUAINT corpus distributed by the LDC). The APW section is 731 MB large and contains 113M word tokens, however we have only used a small portion of it (as described below) for our experiments.

A cursory analysis of our corpus indicates that there are more than a dozen significant classes of numbers.

Numbers are a special type of entity with many interesting properties, e.g., if the numbers under investigation are not entirely random but somehow socially or naturally related, the distribution of the first digit is not uniform. More accurately, digit D appears as the first digit with the frequency proportional to log10(1 + 1/D). In other words, one may expect 1 to be the first digit of a random number in about 30% of cases, 2 will come up in about 18% of cases, 3 in 12%, 4 in 9%, 5 in 8%, etc. This is known as Benford's Law. While we don't use this law in this paper, we have empirically validated its applicability to the APW corpus. For example, there are 53,107 instances of "1", 45,090 instances of "2", 34,395 instances of "3", etc.

For our experiments, we consider the following four types of numerical entities: quantity, time, money, and miscellaneous, effectively merging many of the classes above into a single class "Miscellaneous".

- Quantity: numbers used for counting physical objects or units, such as "1,012

Class	Example
Quantity	25 people
Money	167 3/8
Time	8 a.m., Sept. 17
Score	5 to 2, 4-under-par 68
Age	Kozlov, 24
Address	201 Pennsylvania Ave
Duration	12 years
Percent	25 percent
Temperature	Highs of 28 to 32
Distance	4 feet
Telephone	(212) 555-1902
StockIndex	10000
Measure	10 mph
Miscellaneous	Women's 100, No. 9, Game 4

Table 1: Sample numbers extracted from the APW corpus.

people", "160 miles", etc.

- Time: any number that represents the notion of time, such as year, date, etc.
- Money: all numbers that represent monetary value, such as "$100", "2 million dollars", etc.
- Miscellaneous: all other numbers that don't fall into the categories above, such as rate, percentage, address, phone number, etc.

4 Classification algorithms based on bipartite graphs

Both algorithms that we use for number classification are based on bipartite graphs.

We are considering a set of objects S which is split into two classes, labeled L and unlabeled U.

We are considering joint binary features $f_i(F_j, S_k)$ which is 1 if feature F_j holds on object S_k. For example, f_1 may be a feature that corresponds to the word to the left of an object. In that case, given the word sequence "today 5 people", the following two features can be defined: $f_1(\text{“}today\text{”}, \text{“}5\text{”}) = 1$ while $f_1(\text{“}yesterday\text{”}, \text{“}5\text{”})$ is 0.

The representation is illustrated in Figure 1.

The following notation is used:

- F: the set of features

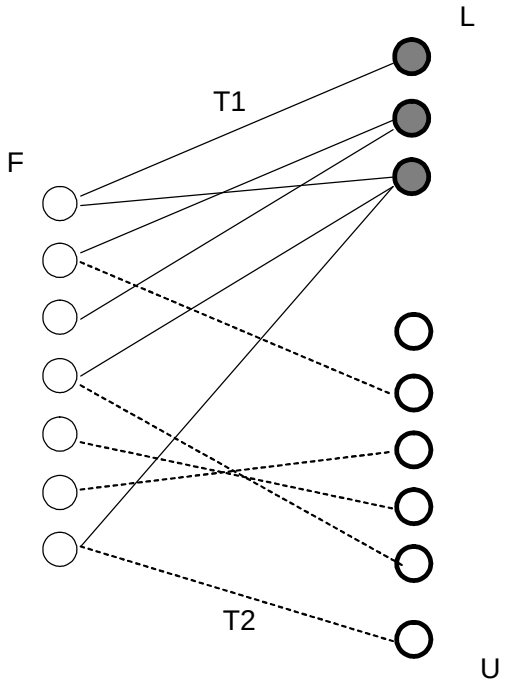

Figure 1: Bipartite representation

- L: the set of labeled (positive) examples
- U: the set of unlabeled examples
- T_1: the connectivity matrix (represented by the bold lines) between F and L
- T_2: the connectivity matrix (represented by the dashed lines) between F and U.

The initial values are set as follows: $|F|$ is equal to the number of distinct features associated with either L or U; each component F_i is set to 0.5; the initial values of L_i are all 1; and the initial values of U_i are all 0.5. The idea is that a 1 represents a certain connection between a feature or a data instance with the positive class and a 0 indicates a certain connection with the negative class.

In this paper, each feature is based on the following history for each object w_i:

$$h_i = \{w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, r(w_i)\}$$

where $r(w_i)$ is the *range* of a number ($w_i \leq 3$, $3 < w_i \leq 10$, $10 < w_i \leq 30$, $30 < w_i \leq 100$), etc. This feature is based on an idea by Jerry Hobbs [10] who claims that humans tend to group numbers into "half-orders of magnitude".

In the rest of this chapter we will describe in turn two graph-based classification algorithms, Tripartite Updating and Spectral Partitioning.

4.1 Tripartite Updating

This is the novel algorithm that we introduce in this paper. It is essentially a bipartite method, however we call it tripartite for reasons that should be evident in the rest of this subsection. Tripartite updating is related to the principal eigenvector of a stochastic Markov process. This algorithm is a variant of the HITS algorithm (it uses a bipartite underlying structure and its stationary solution is computed iteratively), though it differs from it in three important ways: (a) the "right-hand" component of the graph is split into two groups: labeled and unlabeled data instances – therefore the name "tripartite", (b) there is an initial assignment of values for the labeled examples, and (c) the scores of the labeled examples are not allowed to change with time.

The actual algorithm is described here.

$$F^{(t)} = T_1^T L + F^{(t-1)}$$

$$U^{(t)} = T_2 F^{(t)} + U^{(t-1)}$$

$$F^{(t+1)} = T_2^T U^{(t)} + F^{(t)}$$

This process iteratively updates F and U. Note that L is not updated during this process.

The final U vector is then normalized according to the following formula:

$$U = U * 0.5 * |U|$$

4.2 Spectral Partitioning

We use an implementation of Spectral Partitioning (see 2.2) implemented as part of the *meshpart* package developed by Gilbert et al.[8].

4.3 Four-way classifier

We built individual classifiers for each of the four binary cases (money:yes/no, time:yes/no, quantity:yes/no, miscellaneous:yes/no). In the four-way classification scheme, we assign an unlabeled data instance into the class for the classifier that gives it the highest score. An example is shown in Figure 2. The history-based features are marked as follows: *bb* is the word *before-before* w_i (that is, it is w_{i-2}), *b* is w_{i-1}, *a* is w_{i+1} and *aa* is w_{i+2}. The final two features are the Hobbs range of the number and the number itself (not shown in the Feature representation column in the Figure). Note that sometimes the context is incomplete, i.e., some features (e.g., both words on the left of w_i) are not defined. This is due to the way that we extracted context. Instead of picking out entire sentences from the corpus, we limited ourselves to (approximately) 80-byte substrings from the documents as formatted in the original SGML files (that is, each sentence may span several lines but we would look at one line at a time).

The column labeled "Correct class" indicates the gold standard for this word and the column "Assigned class" shows the output of the four-way classifier (based on the largest of the values in the four per-class columns).

5 Experimental results

We used two disjoint sets of numbers for the experiments: labeled data (L) and unlabeled data (U). We limited ourselves to 5,000 unlabeled examples (extracted randomly, with replacement), of which 300 were manually annotated (only for evaluation purposes). We also marked up 200 labeled examples (split into four classes). Again, the labeled examples and the unlabeled examples are drawn independently and don't overlap. The frequencies of these classes are shown in Table 2. The classes are very unevenly distributed with the majority class (miscellaneous)

Nb	Feature representation	Correct class	Money	Quantity	Time	Misc	Assigned
2	bb_300,000 b_to a_ aa_million. 0_3	Money	0.4776	0.4543	**0.6387**	0.5231	Time
10	bb_of b_every a_ aa_acres 4_10	Quantity	0.4773	0.4731	**0.6390**	0.4873	Time
18	bb_14 b_to a_ aa_cents 10_30	Money	0.3671	**0.6241**	0.4461	0.5059	Quantity
8	bb_game b_at a_ aa_p.m. 4_10	Time	0.4774	0.4356	**0.6393**	0.4873	Time
202	bb_of b_the a_ aa_properties 101_300	Quantity	0.3629	**0.5292**	0.4376	0.3976	Quantity
17	bb_the b_Sept. a_ aa_death 10_30	Time	0.3666	0.6053	**0.6426**	0.5237	Time
27	bb_5 b_24 a_ aa_30 10_30	Misc	0.3662	**0.5864**	0.4446	0.5057	Quantity
12	bb_runs b_and a_ aa_hits 10_30	Misc	0.3665	**0.5865**	0.4451	0.5058	Quantity
218	bb_to b_get a_ aa_signatures 101_300	Quantity	0.3624	**0.4916**	0.4369	0.3975	Quantity
56	bb_at b_least a_ aa_ 31_100	Quantity	**0.9375**	0.6055	0.4504	0.5420	Money
199	bb_rounds b_in a_8, aa_on 101_300	Misc	0.0021	**0.1317**	0.0039	0.0003	Quantity
22	bb_squads b_executed a_ aa_people 10_30	Quantity	0.3662	**0.5864**	0.4445	0.5057	Quantity

Figure 2: The four-way tripartite classifier illustrated.

labeled as Miscellaneous. We will report classification results under two conditions (similar to [7]): (a) including and (b) ignoring the miscellaneous (noise) category.

Class	Frequency	Percentage
Money	13	6.5%
Quantity	72	36.0%
Time	16	8.0%
Miscellaneous	99	49.5%

Table 2: Frequencies of the four classes among the 200 labeled examples.

5.1 Evaluation Measures

We will report a number of measures of performance: overall accuracy (how many numbers were classified into each of the four classes), as well the per-class Precision and Recall scores for the four classes.

We will report these measures under two conditions. In the first case, we will include the items labeled as "miscellaneous" in the gold standard while we will ignore them in the second case.

5.2 Results

Tables 3 and 4 show our final results.

The 4-way classification results are very interesting. Tripartite Updating outperforms Spectral Partitioning. None of the two methods seemed to change its performance when given 200 instead of 50 training (labeled) examples. This is

		50 training examples		200 training examples	
		Tripartite	Spectral	Tripartite	Spectral
1000 unlabeled examples	Accuracy	0.28	0.29	0.28	0.08
	Money P/R	0.06/0.33	0/0	0.07/0.33	0.05/0.67
	Quantity P/R	0.47/0.61	0.38/0.39	0.46/0.61	0.75/0.08
	Time P/R	0.19/0.67	0.04/0.33	0.17/0.67	0.06/0.83
	Misc P/R	0/0	0.45/0.29	0/0	0.25/0.02
5000 unlabeled examples	Accuracy	0.28	0.24	0.28	0.17
	Money P/R	0.06/0.33	0.06/0.67	0.07/0.33	0/0
	Quantity P/R	0.47/0.61	0/0	0.46/0.61	0/0
	Time P/R	0.19/0.67	0/0	0.17/0.67	0.03/0.33
	Misc P/R	0/0	0.58/0.75	0/0	0.51/0.71

Table 3: Comparative evaluation of Tripartite Updating and Spectral Partitioning on 100 data points from a set of 1000 or 5000 unlabeled data points. The numbers in this figure are for 4-way classification.

encouraging as it indicates the power of even a minimal number of training examples. Finally, going from 1000 to 5000 unlabeled examples didn't seem to change performance either. This is an encouraging indication that these two methods are scaleable to large amounts of unlabeled data.

The 3-way classification results were obtained by ignoring the Miscellaneous category. The classifier decision was based only on the three binary classifiers for the other three classes while all instances labeled as Miscellaneous in the gold standard are ignored from the computation. The idea here is that in the future one could write classifiers for the classes currently lumped as Miscellaneous and avoid the awkward class frequency distribution that makes Miscellaneous as likely as the other three classes combined. The overall performance on 3-way classification (0.58 accuracy) is significantly higher than 4-way classification (0.28).

6 Conclusion and future work

We presented and compared two weakly supervised graph algorithms for number classification. The results are quite encouraging for future exploration. We found out that both algorithms don't require large amounts of training examples and that they appear to scale well to different ratios between the number of labeled training examples and the number of unlabeled examples. Future experiments are needed to verify these properties. We are also particularly interested in applying similar techniques to other problems such as word sense disambiguation, named entity classification, and document classification. We will also investigate the algorithmic

		50 training examples		200 training examples	
		Tripartite	Spectral	Tripartite	Spectral
1000 unlabeled examples	Accuracy	0.58	0.46	0.48	0.17
	Money P/R	0.17/0.33	0/0	0.17/0.33	0.10/0.67
	Quantity P/R	0.85/0.61	0.72/0.58	0.81/0.61	0.80/0.11
	Time P/R	0.40/0.67	0.13 0.50	0.44/0.67	0.12/0.83
5000 unlabeled examples	Accuracy	0.58	0.13	0.58	0.13
	Money P/R	0.17/0.33	0/0.13	0.17/0.33	0/0
	Quantity P/R	0.85/0.61	0.50/0.03	0.81/0.61	0/0
	Time P/R	0.40/0.67	0/0	0.44/0.67	0.13/1

Table 4: Comparative evaluation of Tripartite Updating and Spectral Partitioning (3-way classification).

properties of these methods by comparing them to known algorithms such as Naive Bayes and Co-training. We are currently working on an enhancement of tripartite updating with active learning.

References

[1] Steven Abney. Bootstrapping. In *Proceedings of ACL-02, 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

[2] R. Agrawal and R. Srikant. Searching with numbers, 2002.

[3] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Knowledge Discovery and Data Mining*, pages 407–416, 2000.

[4] Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

[5] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann Publishers, 1998.

[6] Chris Brew and Sabine Schulte im Walde. Spectral clustering for german verbs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 117–124, Philadelphia, July 2002. Association for Computational Linguistics.

[7] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *EMNLP/VLC-99*, 1999.

[8] John R. Gilbert, Gary L. Miller, and Shang-Hua Teng. Geometric mesh partitioning: Implementation and experiments. *SIAM Journal of Scientific Computing*, 19:2091–2110, 1998.

[9] R. Grishman and B. Sundheim. Message Understanding Conference-6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, June 1996.

[10] Jerry Hobbs. Half orders of magnitude. In *KR-2000 Workshop on Semantic Approximation, Granularity, and Vagueness*, Breckenridge, Colorado, April 2000.

[11] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

[12] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal Of The Acm*, 46(5):604–632, 1999.

[13] Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. The structure and performance of an open-domain question answering system. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong, October 2000.

[14] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[15] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[16] A. Pothen, D. H. Simon, and K. P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, 11:430–452, 1990.

[17] John Prager, Eric Brown, Anni Coden, and Dragomir Radev. Question-answering by predictive annotation. In *Proceedings, 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, July 2000.

[18] R. Sproat, A. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. Normalization of non-standard words. *Computer Speech and Language*, 15(3):287–333, 2001.

[19] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[20] Jean-Philippe Vert and Minoru Kanehisa. Graph-driven feature extraction from microarray data using diffusion kernels and kernel cca. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1425–1432. MIT Press, Cambridge, MA, 2003.

[21] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[22] Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 25–32. ACM Press, 2001.

[23] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. 20th International Conf. on Machine Learning*, 2003.