Transductive Structured Classification through Constrained Min-Cuts

Kuzman Ganchev **Fernando Pereira**
Computer and Information Science
University of Pennsylvania
Philadelphia PA
{kuzman,pereira}@cis.upenn.edu

Abstract

We extend the Blum and Chawla (2001) graph min-cut algorithm to structured problems. This extension can alternatively be viewed as a joint inference method over a set of training and test instances where parts of the instances interact through a pre-specified associative network. The method has has an efficient approximation through a linear-programming relaxation. On small training data sets, the method achieves up to 34.8% relative error reduction.

1 Introduction

We describe a method for transductive classification in structured problems. Our method extends the Blum and Chawla (2001) algorithm for transductive classification. In that algorithm, each training and test instance is represented by a vertex in a graph. The algorithm finds the min-cut that separates the positively and negatively labeled instances. We give a linear program that implements an approximation of this algorithm and extend it in several ways. First, our formulation can be used in cases where there are more than two labels. Second, we can use the output of a classifier to provide a prior preference of each instance for a particular label. This lets us trade off the strengths of the min-cut algorithm against those of a standard classifier. Finally, we extend the algorithm further to deal with structured output spaces, by encoding parts of instances as well as constraints that ensure a consistent labeling of an entire instance.

The rest of this paper is organized as follows. Section 2 explains what we mean by transductive classification and by structured problems. Section 3 reviews the Blum and Chawla (2001) algorithm, how we formulate it as a linear program and our proposed extensions. Section 4 relates our proposal to previous work. Section 5 describes our experimental results on real and synthetic data and Section 6 concludes the paper.

2 Concepts and Notation

In this work we combine two separate approaches to learning: transductive methods, in which classification of test instances arises from optimizing a single objective involving both training and test instances; and structured classification, in which instances involve several interdependent classification problems. The description of structured problems also introduces useful terminology for the rest of the paper.

2.1 Transductive Classification

In supervised classification, training instances are used to induce a classifier that is then applied to individual test instances that need to be classified. In transductive classification, a single optimization problem is set up involving all training and test instances; the solution of the optimization problem yields labels for the test instances. In this way, the test instances provide evidence about the distribution of the data, which may be useful when the labeled data is limited and the distribution of unlabeled data

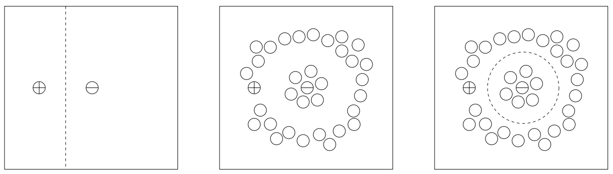

Figure 1: An example where unlabeled data helps to reveal the underlying distribution of the data points, borrowed from Sindhwani et al. (2005). The circles represent data points (unlabeled are empty, positive have a "+" and negative have a "-"). The dashed lines represent decision boundaries for a classifier. The first figure shows the labeled data and the max-margin decision boundary (we use a linear boundary to conform with Occam's razor principle). The second figure shows the unlabeled data points revealing the distribution from which the training examples were selected. This distribution suggests that a linear boundary might not be appropriate for this data. The final figure shows a more appropriate decision boundary given the distribution of the unlabeled data.

is informative about the location of the decision boundary. Figure 1 illustrates this.

2.2 Structured Classification

The usual view of structured classification is as follows. An *instance* consists of a set of classification problems in which the labels of the different problems are correlated according to a certain graphical structure. The collection of classification labels in the instance forms a single *structured label.* A typical structured problem is part of speech (POS) tagging. The parts of speech of consecutive words are strongly correlated, while the POS of words that are far away do not influence each other much. In the natural language processing tasks that motivate this work, we usually formalize this observation with a Markov assumption, implemented by breaking up the instance into *parts* consisting of pairs of consecutive words. We assign a score for each possible label of each part and then use a dynamic programming algorithm to find the highest scoring label of the entire instance.

In the rest of this paper, it will be sometimes more convenient to think of all the (labeled and unlabeled) instances of interest as forming a single joint classification problem on a large graph. In this joint problem, the atomic classification problems are linked according to the graphical structure imposed by their partition into structured classification instances. As we will see, other links between atomic problems arise in our setting that may cross between different structured instances.

2.3 Terminology

For structured problems, *instance* refers to an entire problem (for example, an entire sentence for POS tagging). A *token* refers to the smallest unit that receives a label. In POS tagging, a *token* is a word. A *part* is one or more *tokens* and is a division used by a learning algorithm. For all our experiments, a *part* is a pair of consecutive tokens, but extension to other types of parts is trivial. If two parts share a token then a consistent label for those parts has to have the same label on the shared token. For example in the sentence "I love learning ." we have parts for "I love" and "love learning". These share the token "love" and two labels for the two parts has to agree on the label for the token in order to be consistent. In all our experiments, a part is a pair of consecutive tokens so two parts are independent unless one immediately follows the other.

3 Approach

We extend the min-cut formulation of Blum and Chawla (2001) to multiple labels and structured variables by adapting a linear-programming encoding of metric labeling problems. By relaxing the linear program, we obtain an efficient approximate inference algorithm. To understand our method, it is useful to review the min-cut transductive classification algorithm (Section 3.1) as well as the metric labeling problem and its linear programming relaxation (Section 3.2). Section 3.3 describes how to encode a multi-way min-cut problem as an instance of metric labeling as well as a trivial extension that lets us introduce a bias when computing the cut.

Section 3.4 extends this formalism to structured classification.

3.1 Min-Cuts for Transductive Classification

Blum and Chawla (2001) present an efficient algorithm for semi-supervised machine learning in the unstructured binary classification setting. At a high level, the algorithm is as follows:

- Construct a graph where each instance corresponds to a vertex;
- Add weighted edges between similar vertices with weight proportional to a measure of similarity;
- Find the min-cut that separates positively and negatively labeled training instances;
- Label all instances on the positive side of the cut as positive and all others as negative.

For our purposes we need to consider two extensions to this problem: multi-way classification and constrained min-cut.

For multi-way classification, instead of computing the binary min-cut as above, we need to find the multi-way min-cut. Unfortunately, doing this in general is NP-hard, but a polynomial time approximation exists (Dahlhaus et al., 1992). In Section 3.3 we describe how we approximate this problem.

We extend this approach to structured data by constructing a graph whose vertices correspond to different parts of the instance, and add weighted edges between similar parts. We then find the multi-way min-cut that separates vertices with different labels subject to some constraints: if two parts overlap then the labels have to be consistent. Our main contribution is an algorithm that approximately computes this constrained multi-way min-cut with a linear programming relaxation.

3.2 Metric Labeling

Kleinberg and Tardos (1999) introduce the metric labeling problem as a common inference problem in a variety of fields. The inputs to the problem are a weighted graph $G = (V, E)$, a set of labels $L = \{i | i \in 1 \ldots k\}$, a cost function $c(v, i)$ which represents the preference of each vertex for each possible label and a metric $d(i, j)$ between labels i and j. The goal is to assign a label to each vertex $l : V \to L$ so as to minimize the cost given by:

$$\begin{aligned} c(l) &= \textstyle\sum_{v \in V} c(v, l(v)) \\ &+ \textstyle\sum_{(u,v) \in E} d(l(u), l(v)) \cdot w(u, v)\ . \end{aligned} \qquad (1)$$

Kleinberg and Tardos (1999) give a linear programming approximation for this problem with an approximation factor of two and explain how this can be extended to an $O(\log k)$ approximation for arbitrary metrics by creating a hierarchy of labels. Chekuri et al. (2001) present an improved linear program that incorporates arbitrary metrics directly and provides an approximation at least as good as that of Kleinberg and Tardos (1999). The idea in the new linear program is to have a variable for each edge labeling as well as one for each vertex labeling.

Following Chekuri et al. (2001), we represent the event that vertex u has label i by the variable $x(u, i)$ having the value 1; if $x(u, i) = 0$ then vertex v must have some other label. Similarly, we use the variable and value $x(u, i, v, j) = 1$ to mean that the vertices u and v (which are connected by an edge) have label i and j respectively. The edge variables allow us to encode the costs associated with violated edges in the metric labeling problem. Edge variables should agree with vertex labels, and by symmetry we should have $x(u, i, v, j) = x(v, j, u, i)$. If the linear program gives an integer solution, this is clearly the optimal solution to the original metric labeling instance. Chekuri et al. (2001) describe a rounding procedure to compute an integer solution to the LP that is guaranteed to be an approximation of the optimal integer solution. For the problems we considered, this was very rarely necessary. Their linear program relaxation is shown in Figure 2. The cost function is the sum of the vertex costs and edge costs. The first constraint requires that each vertex have a total of one labeling unit distributed over its labels, that is, we cannot assign more or less than one label per vertex. The second constraint

$$\begin{aligned}
\min \quad & \sum_{u\in V}\sum_{i\in L} c(u,i)x(u,i) \\
& + \sum_{(u,v)\in E}\sum_{k,j\in L} w(u,v)d(i,j)x(u,i,v,j)
\end{aligned}$$

subject to

$$\begin{aligned}
\sum_{i\in L} x(u,i) = 1 \quad & \forall u \in V \\
x(u,i) - \sum_{j\in L} x(u,i,v,j) = 0 \quad & \forall u \in V, v \in N(u), i \in L \\
x(u,i,v,j) - x(v,j,u,i) = 0 \quad & \forall u,v \in V, i,j \in L \\
x(u,i,v,j), x(u,i) \in [0,1] \quad & \forall u,v \in V, i,j \in L
\end{aligned}$$

Figure 2: The Chekuri et al. (2001) linear program used to approximate metric labeling. See text for discussion.

requires that vertex- and edge-label variables are consistent: the label that vertex variables give a vertex should agree with the labels that edge variables give that vertex. The third constraint imposes the edge-variable symmetry condition, and the final constraint requires that all the variables be in the range $[0, 1]$.

3.3 Min Cut as an Instance of Metric Labeling

Given an instance of the (multi-way) min-cut problem, we can translate it to an instance of metric labeling as follows. The underlying graph and edge weights will be the same as min-cut problem. We add vertex costs ($c(u,i)\ \forall u \in V, i \in L$) and a label metric ($d(i,j)\ \forall i,j \in L$). For all unlabeled vertices set the vertex cost to zero for all labels. For labeled vertices set the cost of the correct label to zero and all other labels to infinity. Finally let $d(i,j)$ be one if $i \neq j$ and zero otherwise.

The optimal solution to this instance of metric labeling will be the same as the optimal solution of the initial min cut instance: the cost of any labeling is the number of edges that link vertices with different labels, which is exactly the number of cut edges. Also by the same argument, every possible labeling will correspond to some cut and approximations of the metric labeling formulation will be approximations of the original min-cut problem.

Since the metric labeling problem allows arbitrary affinities between a vertex in the graph and possible labels for that vertex, we can trivially extend the algorithm by introducing a bias at each vertex for labels more compatible with that vertex. We use the output of a classifier to bias the cost towards agreement with the classifier. Depending on the strength of the bias, we can trade off our confidence in the performance of the min-cut algorithm against the our confidence in a fully-supervised classifier.

3.4 Extension to Structured Classification

To extend this further to structured classification we modify the Chekuri et al. (2001) linear program (Figure 2). In the structured case, we construct a vertex for every part of an instance. Since we want to find a consistent labeling for an entire instance composed of overlapping parts, we need to add some more constraints to the linear program. We want to ensure that if two vertices correspond to two overlapping parts, then they are assigned consistent labels, that is, the token shared by two parts is given the same label by both. First we add a new zero-weight edge between every pair of vertices corresponding to overlapping parts. Since its weight is zero, this edge will not affect the cost. We then add a constraint to the linear-program that the edge variables for inconsistent labelings of the new edges have a value of zero.

More formally, let $(u,i,v,j) \in \Lambda$ denote that the part u having label i is consistent with the part v having label j; if u and v do not share any tokens, then any pair of labels for those parts are consistent. Now add zero-weight edges between overlapping parts. Then the only modification to the linear program is that

$$x(u,i) - \textstyle\sum_{j\in L} x(u,i,v,j) = 0$$
$$\forall u \in V, v \in N(u), i \in L$$

will become

$$x(u,i) - \textstyle\sum_{j:(u,i,v,j)\in\Lambda} x(u,i,v,j) = 0$$
$$\forall u \in V, v \in N(u), i \in L \ .$$

$$\begin{aligned}\min \quad & \sum_{u \in V} \sum_{i \in L} c(u,i)x(u,i) \\ & + \sum_{(u,v)\in E} \sum_{k,j \in L} w(u,v)d(i,j)x(u,i,v,j)\end{aligned}$$

subject to

$$\begin{aligned} \sum_{i \in L} x(u,i) = 1 \quad & \forall u \in V \\ x(u,i) - \sum_{j:(u,i,v,j)\in\Lambda} x(u,i,v,j) = 0 \quad & \forall u \in V, v \in N(u), i \in L \\ x(u,i,v,j) - x(v,j,u,i) = 0 \quad & \forall\ (u,i,v,j) \in \Lambda \\ x(u,i,v,j), x(u,i) \in [0,1] \quad & \forall u,v \in V, i,j \in L \end{aligned}$$

Figure 3: The modified linear program used to approximate metric labeling. See text for discussion.

What this modification does is to ensure that all the mass of the edge variables between vertices u and v lies in consistent labelings for their edge. The modified linear program is shown in Figure 3. We can show that this can be encoded as a larger instance of the metric labeling problem (with roughly $|V|+|E|$ more vertices and a label set that is four times as large), but modifying the linear program directly results in a more efficient implementation. The final LP has one variable for each labeling of each edge in the graph, so we have $O(|E||L|^2)$ variables. Note that $|L|$ is the number of labelings of a pair of tokens for us – even so, computation of a single dataset took on the order of minutes using the Xpress MP package.

4 Relation to Previous work

Our work is set of extensions to the work of-Blum and Chawla (2001), which we have already described. Our extensions allow us to handle multi-class and structured data, as well as to take hints from a classifier. We can also specify a similarity metric between labels so that a cut-edge can cost different amounts depending on what partitions it spans.

Taskar et al. (2004a) describe a class of Markov networks with associative clique potentials. That is, the clique potentials always prefer that all the nodes in the clique have the same label. The inference problem in these networks is to find the assignment of labels to all nodes in the graph that maximizes the sum of the clique potentials. Their paper describes a linear programming relaxation to find (or approximate) this inference problem which is very similar to the LP formulation of Chekuri et al. (2001) when all cliques are of size 2. They generalize this to larger cliques and prove that their LP gives an integral solution when the label alphabet has size 2 (even for large cliques). For the learning problem they exploit the dual of the LP formulation and use a maximum margin objective similar to the one used by Taskar et al. (2004b). If we ignore the learning problem and focus on inference, one could view our work as inference over a Markov network created by combining a set of linear chain conditional random fields with an associative Markov network (with arbitrary structure). A direction for future work would be to train the associative Markov network either independently from the chain-structured model or jointly with it. This would be very similar to the joint inference work described in the next paragraph, and could be seen as a particular instantiation of either a non-linear conditional random field (Lafferty et al., 2001) or relational Markov network (Taskar et al., 2002).

Sutton and McCallum (2004) consider the use of linear chain CRFs augmented with extra *skip edges* which encode a probabilistic belief that the labels of two entities might be correlated. They provide experimental results on named entity recognition for e-mail messages announcing seminars, and their system achieves a 13.7% relative reduction in error on the "Speaker" field. Their work differs from ours in that they add skip edges only between identical capitalized words and only within an instance, which for them is an e-mail message. In particular, they can never have an edge between labeled and unlabeled parts. Their approach is useful for identification of personal names but less helpful for other named entity tasks where the names may not be capitalized.

Lafferty et al. (2004) show a representer theorem allowing the use of Mercer kernels with

CRFs. They use a kernel CRF with a graph kernel (Smola and Kondor, 2003) to do semi-supervised learning. For them, the graph defines an implicit representation of the data, but inference is still performed only on the (chain) structure of the CRF. By contrast, we perform inference over the whole set of examples at the same time.

Altun et al. (2006) extend the use of graph-based regularization to structured variables. Their work is in the framework of maximum margin learning for structured variables where learning is framed as an optimization problem. They modify the objective function by adding a penalty whenever two parts that are expected to have a similar label assign a different score to the same label. They show improvements of up to 5.3% on two real tasks: pitch accent prediction and optical character recognition (OCR). Unfortunately, to solve their optimization problem they have to invert an $n \times n$ matrix, where n is the number of parts in the training and testing data times the number of possible labels for each part. Because of this they are forced to train on an unrealistically small amount of data (4-40 utterances for pitch accent prediction and 10 words for OCR).

5 Experiments

We performed experiments using our approach on three different datasets using a conditional random field as the base classifier. Unless otherwise noted this was regularized using a zero-mean Gaussian prior with a variance of 1.

The first dataset is the pitch-accent prediction dataset used in semi-supervised learning by Altun et al. (2006). There are 31 real and binary features (all are encoded as real values) and only two labels. Instances correspond to an utterance and each token corresponds to a word. Altun et al. (2006) perform experiments on 4 and 40 training instances using at most 200 unlabeled instances.

The second dataset is the reference part of the Cora information extraction dataset.[1] This consists of 500 computer science research paper citations. Each token in a citation is labeled as being part of the name of an author, part of the title, part of the date or one of several other labels that we combined into a single category ("other").

The third dataset is the chunking dataset from the CoNLL 2000 (Sang and Buchholz, 2000) shared task restricted to noun phrases. The task for this dataset is, given the words in a sentence as well as automatically assigned parts of speech for these words, label each word with `B-NP` if it is the first word in a base noun phrase, `I-NP` if it is part of a base noun phrase but not the first word and `O` if it is not part of a noun phrase.

For all experiments, we let each word be a token and consider parts consisting of two consecutive tokens.

5.1 Pitch Accent Prediction

For the pitch accent prediction dataset, we used the 5-nearest neighbors of each instance according to the Euclidean distance in the original feature space to construct the graph for min-cut. Table 1 shows the results of our experiments on this data, as well as the results reported by Altun et al. (2006). The numbers in the table are per-token accuracy and each entry is the mean of 10 random train-test data selections.

For this problem, our method improves performance over the base CRF classifier (except when the training data consists of only 4 utterances), but we do not see improvements as dramatic as those observed by Altun et al. (2006). Note that even the larger dataset here is quite small – 40 utterances where each token has been annotated with a binary value.

5.2 Cora-IE

For the Cora information extraction dataset, we used the first 100 principal components of the feature space to find 5 nearest neighbors of each part. This approximation is due to the cost of comuting nearest neighbors in high dimensions. In these experiments we trained on 40 instances

[1]The Cora IE dataset has been used in Seymore et al. (1999), Peng and McCallum (2004), McCallum et al. (2000) and Han et al. (2003), among others. We obtained the dataset from `http://www.cs.umass.edu/~mccallum/data/cora-ie.tar.gz`.

Method	4:80	40:80	40:200
CRF	71.2	72.5	73.1
MinCut	69.4	74.4	74.3
STR	70.7	75.7	77.5
SVM	69.9	72.0	73.1

Table 1: Results on the pitch accent prediction task. The methods we compare are as follows. CRF is supervised CRF training. MinCut is our method with a CRF as base classifier. STR and SVM are the semi-supervised results reported in Altun et al. (2006). The experiments are 4 labeled and 80 unlabeled, 40 labeled and 80 unlabeled and 40 labeled and 200 unlabeled respectively.

Variance	10	100	1000
CRF	84.5%	84.3%	83.9%
MinCut	88.8%	89.6%	89.9%

Table 2: Accuracy on the Cora-IE dataset as a percentage of tokens correctly classified at different settings for the CRF variance. Results for training on 40 instances and testing on 80. In all cases the scores are the mean of 10 random selections of 120 instances from the set of 500 available.

and used 80 as testing data. In all cases we randomly selected training and testing instances 10 times from the total set of 500. Table 2 shows the average accuracies for the 10 repetitions, with different values for the variance of the Gaussian prior used to regularize the CRF. If we choose the optimal value for each method, our approach gives a 34.8% relative reduction in error over the CRF, and improves over it in each of the 10 random data selections, and all settings of the Guassian prior variance.

5.3 CoNLL NP-Chunking

Our results are worst for the CoNLL NP-Chunking dataset. As above, we used 10 random selections of training and test sets, and used the 100 principal components of the feature space to find 5 nearest neighbors of each part. Table 3 shows the results of our experiments. The numbers in the table are per-token accuracy as before. When the amount of training data is very small (20 instances) we improve slightly over the base CRF classifier, but with an increased amount of training data, the small improvement is replaced with a small loss.

Method	20:40	40:80
CRF	87.6	90.6
MinCut(CRF)	88.2	89.6

Table 3: Results on the NP-chunking task. The table compares a CRF with our method using a CRF as a base classifier. The experiments use 20 labeled and 40 unlabeled and 40 labeled and 80 unlabeled instances.

6 Discussion

We have presented a new transductive algorithm for structured classification, which achieves error reductions on some real-world problems. Unfortunately, those gains are not always realized, and sometimes our approach leads to an increase in error. The main reason that our approach does not always work seems to be that our measure of similarity between different parts is very coarse. In general, finding all the pairs of parts have the same label is as difficult as finding the correct labeling of all instances, but it might be possible to use unlabeled data to learn the similarity measure.

References

Yasemin Altun, David McAllester, and Mikhail Belkin. 2006. Maximum margin semi-supervised learning for structured variables. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 33–40. MIT Press, Cambridge, MA.

Avrim Blum and Shuchi Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conf. on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA.

Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. 2001. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Symposium on Discrete Algorithms*, pages 109–118.

E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. 1992. The complexity of multiway cuts (extended abstract). In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 241–251, New York, NY, USA. ACM Press.

H. Han, C. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E. Fox. 2003. Automatic document metadata extraction using support vector machines. In *Joint Conference on Digital Libraries.*

Jon Kleinberg and Eva Tardos. 1999. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 14, Washington, DC, USA. IEEE Computer Society.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 10th International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

John Lafferty, Xiaojin Zhu, and Yan Liu. 2004. Kernel conditional random fields: representation and clique selection. In *Proceedings of the twenty-first international conference on Machine learning*, page 64, New York, NY, USA. ACM Press.

A. McCallum, K. Nigam, J. Rennie, and K. Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163.

Fuchun Peng and Andrew McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *Main Proceedings of HLT-NAACL*, pages 329–336, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Erik Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*. Association for Computational Linguistics.

K. Seymore, A. McCallum, and R. Rosenfeld. 1999. Learning hidden markov model structure for information extraction. In *AAAI'99 Workshop on Machine Learning for Information Extraction.*

Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 824–831.

Alexander Smola and Risi Kondor. 2003. Kernels and regularization on graphs. In M. Warmuth and B. Scholkopf, editors, *Proceedings of the Sixteenth Annual Conference on Learning Theory and Kernels Workshop.*

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, July. Presented at ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields.

Ben Taskar, Abbeel Pieter, and Daphne Koller. 2002. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 485–492, San Francisco, CA. Morgan Kaufmann Publishers.

B. Taskar, V. Chatalbashev, and D. Koller. 2004a. Learning associative markov networks. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML).*

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004b. Max-margin markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16.* MIT Press, Cambridge, MA.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484.