Exact and Approximate Inference for Annotating Graphs with Structural SVMs*

Thoralf Klein[1], Ulf Brefeld[2], and Tobias Scheffer[1]
{tklein,scheffer}@mpi-inf.mpg.de, brefeld@cs.tu-berlin.de

[1] Max Planck Institute for Computer Science, Saarbrücken, Germany
[2] Machine Learning Group, Technische Universität Berlin, Germany

Abstract. Training processes of structured prediction models such as structural SVMs involve frequent computations of the *maximum-a-posteriori* (MAP) prediction given a parameterized model. For specific output structures such as sequences or trees, MAP estimates can be computed efficiently by dynamic programming algorithms such as the Viterbi algorithm and the CKY parser. However, when the output structures can be arbitrary graphs, exact calculation of the MAP estimate is an NP-complete problem. In this paper, we compare exact inference and approximate inference for labeling graphs. We study the exact junction tree and the approximate loopy belief propagation and sampling algorithms in terms of performance and ressource requirements.

1 Introduction

Many problem settings which require the prediction of multiple dependent variables arise naturally. For instance, sequential input and output variables occur in protein secondary structure prediction and tree-structured output is produced in natural language parsing. Examples for general graphical output structures include classification of linked documents or webpages [11, 8, 10], and simultaneous prediction of multiple dependent class labels [3].

Many classical learning algorithms have been lifted to deal with structured variables. Generally, the learning task is phrased as finding a function f such that

$$\hat{y} = \operatorname*{argmax}_y f(x, y) \tag{1}$$

is the desired output for a given input x. Conditional random fields [6] and structural support vector machines [13, 12] are parameter estimation techniques that are wrapped around the collective inference machinery. However, inferring the actual prediction acts as a bottleneck in the training process. Dynamic programming approaches such as the Viterbi algorithm and the CKY algorithm efficiently solve the inference problem for sequential and tree-structured output variables.

* *Proc. of the European Conference on Machine Learning (ECML),* ©*Springer, 2008*

When the structures involved can be arbitrary graphs, exact inference is not tractable: For instance, there is no efficient analytic solution for discrete variables and dynamic programming for exact inference scales exponentially in the size of the largest clique of the graph in terms of memory and computation time requirements. Hence, practitioners usually resort to approximate inference techniques. For instance, loopy belief propagation and Gibbs sampling are appealing and intuitive approaches that lead to efficient algorithms which alleviate excessive computational requirements. Although these approximate variants are not guaranteed to converge – let alone to find good optima – they seem to be surprisingly well-suited for many practical applications [7, 10, 3].

In this paper, we present a substantial comparison of exact and approximate inference techniques for their use with structural support vector machines. We address their implications on the SVM algorithm in terms of convergence, resource requirements, and performance. Our experiments reveal that exact inference is indispensable for the reliable learning of accurate prediction models. According to our findings, a remedy to the computational costs of the junction tree algorithm is to decompose large graphs into smaller subgraphs.

Our paper is structured as follows. We introduce the learning task formally in Section 2 and review structural support vector machines in Section 3. Section 4 addresses exact and approximate inference algorithms and Section 5 reports on our empirical results. We discuss our findings in Section 6 and Section 7 concludes.

2 Learning with Graphs

Let $G = (V, E)$ be a graph such that the set of nodes decomposes into a set of observed nodes X and a set of latent nodes Y the labeling of which remains to be conjectured. The set of edges $E \subseteq (X \times Y) \cup (Y \times Y)$ introduces a dependency structure between nodes $V = X \cup Y$. That is, two variables are connected if they directly depend on each other. A *structured input* x is a manifestation of a graph together with a labeling of its observed nodes. A *structured output* is a labeling $y \in \Sigma^{|Y|}$, where Σ denotes the label alphabet.

The random variables $X \cup Y$ form a Markov random field with dependency structure E. The conditional probability of Y given a partial realization (observation) x can be written as

$$\hat{p}(y|x) = \exp\{\langle \lambda, \Phi(x, y) \rangle - \log g(\lambda|x)\}, \tag{2}$$

where $g(\lambda|x) = \sum_{\bar{y}} \exp\{\langle \lambda, \Phi(x, \bar{y}) \} < \infty$ is called the partition function, Φ is the sufficient statistics, and λ denotes the natural parameter. The sufficient statistics factorizes according to the dependency structure into terms of the maximal cliques $C \in \mathcal{C}$ of the graph,

$$\Phi(x, y) = \sum_{C \in \mathcal{C}} \phi_C(x_C, y_C).$$

Functions ϕ_C may be interpreted as joint feature vectors, extracted from clique C and therefore encode the dependency structure to allow the model to learn about dependencies between input and output variables.

When dealing with arbitrary graphs, cliques may grow arbitrarily large and employing feature functions for all possible cliques can become intractable. As a remedy, one usually resorts to factorizing over all cliques that contain a pair of nodes, rather than over all maximally large cliques. In this case, G is sometimes called a Markov network and every edge in G is associated with a feature function ϕ. That is, we have edges between label and observation pairs,

$$\phi_1(x_i, y_i) = (\delta_{k_1,y_i}, \ldots, \delta_{k_{|\Sigma|},y_i})^\mathsf{T} \otimes \psi(x_i),$$

and between neighboring labels,

$$\phi_2(y_i, y_j) = \begin{pmatrix} \delta_{k_1,y_i} \\ \vdots \\ \delta_{k_{|\Sigma|},y_i} \end{pmatrix} \otimes \begin{pmatrix} \delta_{k_1,y_j} \\ \vdots \\ \delta_{k_{|\Sigma|},y_j} \end{pmatrix},$$

where ψ is a feature vector solely drawn from the observation, $k \in \Sigma$ enumerates all possible labels, $\delta_{i,j}$ is the Kronecker product and $\otimes$ denotes the tensor product. Equation 2 says that given parameters λ, the most likely labeling $\hat{y}$ can be computed using the generalized linear model

$$\hat{y} = \operatorname{argmax}_{\bar{y}} \hat{p}(\bar{y}|x) = \operatorname{argmax}_{\bar{y}} f(x, \bar{y}),$$

with $f(x, y) = \langle \lambda, \Phi(x, y) \rangle$. We thus seek to find a parameterization λ, such that the model f generalizes well on new and unseen graphs.

The quality of f is measured by a task-dependent loss function $\Delta : (y, y') \mapsto r \in \mathbb{R}_0^+$. We require that Δ is decomposable in terms of the nodes of graph, for instance, Δ may be a Hamming-like loss given by

$$\Delta(y, y') = \sum_{j=1}^{|y|} [[y_j \neq y'_j]]. \tag{3}$$

The ultimate goal is to find f such as to minimize the true loss $\sum_y \int \Delta(y, \operatorname{argmax}_{\bar{y}} f(x, \bar{y})) p(x, y) dx$. Being ignorant of the true distribution $p(x, y)$ of instances and labelings of the unobserved variables, one resorts to declaring minimization of the regularized empirical loss on an iid sample $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{\ell}$ to be the operational goal

$$\hat{R}[f] = \sum_{i=1}^{\ell} \Delta(y^{(i)}, \operatorname*{argmax}_{\bar{y}} f(x^{(i)}, \bar{y})) + \frac{1}{\eta} \|f\|^2.$$

3 Structural Support Vector Machines

Structural SVMs [12, 13] adapt the parameters of ranking functions that are defined over joint attributes of input and output:

$$f(x, y) = \langle \lambda, \Phi(x, y) \rangle. \tag{4}$$

Given training data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{\ell}$, SVMs aim at finding λ that minimizes $|\lambda|^2 + \sum_i \xi_i$ subject to the constraint that, for all $(x^{(i)}, y^{(i)})$, the desired output $y^{(i)}$ exceeds any other output $\bar{y}$ in its decision function values by at least $1 - \xi_i$:

$$\forall_{i=1}^{\ell} \quad f(x^{(i)}, y^{(i)}) - \max_{\bar{y} \neq y^{(i)}} f(x^{(i)}, \bar{y}) \geq 1 - \xi_i.$$

Optimization Problem 1 is solved iteratively by column generation: If at least one output violates the margin constraint for a given x, then the output $\bar{y}$ that violates it most strongly is inferred and added to a working set of explicitly represented constraints. Column generation is interleaved with optimization steps in which the current hypothesis λ is refined according to the current working set. Typically only a small fraction of conceivable constraints are represented in the working set and structural SVMs provide sparse solutions.

Optimization Problem 1 *Given data $\mathcal{D}$, loss function Δ, and $\eta > 0$, the structural SVM optimization problem is defined as:*

$$\min_{\lambda, \xi \geq \mathbf{0}} \frac{|\lambda|^2}{2} + \eta \langle \xi, \mathbb{1} \rangle$$

subject to the constraints

$$\langle \lambda, \Phi(x^{(i)}, y^{(i)}) \rangle \geq \max_{\bar{y} \neq y^{(i)}} [\Delta(y^{(i)}, \bar{y}) + \langle \lambda, \Phi(x^{(i)}, \bar{y}) \rangle] - \xi_i$$

for all $i = 1, \ldots, \ell$.

Provided that exact inference of large margin violators runs in polynomial time, the solution to Optimization Problem 1 converges to the optimum in polynomial time [13].

4 Inference Strategies

Intuitively, Equation 1 can be solved by explicitly computing the score for all possible assignments of the output variables and choosing the output $\hat{y}$ that realizes the highest score. However, there are exponentially many different assignments in the size of the graph and explicit enumeration is prohibitive. In this section, we briefly review the exact junction tree algorithm, the approximate loopy belief propagation and an inference strategy based on Gibbs sampling.

4.1 Junction Tree Algorithm

Belief propagation [1] sends messages across edges of the graph that are used to update actual beliefs about labelings. Belief propagation terminates and produces the exact joint probability of all unobserved variables when the graph is free of cycles. Therefore, one transforms the graph into a junction tree $J = (\mathcal{C}, E_J)$; nodes in the junction tree correspond to cliques in the underlying graph.

Message propagation consists of two phases. In the *distribute evidence* phase, every node $A \in \mathcal{C}$ in the junction tree that receives a message from its parent, sends a messages m_{AB} to its children.

$$m_{AB}(y_{A \cap B}) = \underset{y_{A \setminus B}}{\operatorname{argmax}} \langle \lambda, \phi(x_A, y_A) \rangle.$$

In the *collect evidence* phase, every node in the junction tree that received messages from all children, sends the messages m_{BA} back to its parent.

After the two phases the junction tree is in equilibrium and further iterations will not affect the actual beliefs. The Viterbi (for sequences) and inside-outside algorithms (for trees) are special cases of message propagation. The most likely labeling can be computed by dynamic programming [9] according to

$$\hat{y} = \underset{y}{\operatorname{argmax}} \sum_{C \in \mathcal{C}} \langle \lambda, \phi_C(x_C, y_C) \rangle - \sum_{AB \in E_J} m_{BA}(y_{A \cap B})$$

The overall complexity is $\mathcal{O}(\exp |V|)$ for building the junction tree and $\mathcal{O}(|\Sigma|^{max|C|})$ for the message passing.

4.2 Loopy Belief Propagation

Similar to Section 4.1, loopy belief propagation propagates messages across the graph. However, instead of using the computationally expensive junction tree, messages are sent in the original graph. Messages encode beliefs about the labeling of direct neighbors in the graph and are passed simultaneously between all nodes. The message from node i to j is computed according to

$$m_{ij}(y_j) = \max_{y_i} \left\{ \langle \lambda, \phi_1(x_i, y_i) \rangle + \langle \lambda, \phi_2(y_i, y_j) \rangle + \sum_{j : y_{ij} \in E} m_{ji}(y_i) \right\}.$$

Although iteratively updating the marginals does not necessarily result in convergence – let alone convergence to the global optimum – loopy belief propagation has been found to work well in practice [7]. After termination, the highest scoring labeling can be computed by

$$\hat{y}_i = \underset{y_i}{\operatorname{argmax}} \left[\langle \lambda, \phi_1(x_i, y_i) \rangle + \sum_{j : y_{ij} \in E} m_{ji}(y_i) \right].$$

The computational cost for each iteration is $\mathcal{O}(|E||Y|^2)$.

4.3 Gibbs Sampling

Repeatedly iterating over the latent variables and drawing a value for each variable y_i according to the conditional probability $\hat{p}(y_i | x, \{y_j : j \neq i\})$ creates a Markov chain of observations that converges to the joint probability $\hat{p}(y|x)$.

This Gibbs sampling rule [4] can be used to approach the assignment of values to y that maximizes the decision function $f(x, y)$. The Gibbs decoder starts with an initial random guess at y. In each iteration, a latent variable Y_i is drawn uniformly and labeled according to $\hat{p}(y_i|x, \{y_j : j \neq i\})$. The new labeling is kept if it is more likely than the previous one. The conditional probability can be derived from the score which the SVM assigns to the pair of input x and output y; we will now describe this process. Parameters λ induce the joint probability $p(y)$ given by

$$\hat{p}(y) \propto \exp\left\{\sum_{i \in V} \langle \lambda, \phi_1(x_i, y_i)\rangle + \sum_{ij \in E} \langle \lambda, \phi_2(y_i, y_j)\rangle\right\},$$

which follows from the definition of the conditional probability. Inserting $p(y)$ in Equation 5 gives us Equation 6. Please note, that nominator and denominator have many factors in common, that is, all factors that do not contain the variable y_i. It is easy to see that cancelling these factors gives Equation 7. This final equation also shows that the full conditionals of y_i only depend on parameters λ and the given feature functions. Furthermore, we only need to consider the neighborhood of the i-th node, i.e. it does not depend on the size of the graph. Let $\bar{y}^\sigma$ defined as $\bar{y}_i^\sigma = y_i$ for $j \neq i$ and $\bar{y}_j^\sigma = \sigma$ in case $i = j$, we have,

$$\hat{p}(y_i|x, \{y_j : j \neq i\}) = \frac{p(Y_i = y_i, x, \{y_j : j \neq i\})}{\sum_{\sigma \in \Sigma} p(Y_i = \sigma, x, \{y_j : j \neq i\})} \tag{5}$$

$$= \frac{\exp\left\{\sum_{k \in V} \langle \lambda, \phi_1(x_k, y_k)\rangle + \sum_{kl \in E} \langle \lambda, \phi_2(y_k, y_l)\rangle\right\}}{\sum_{\sigma \in \Sigma} \exp\left\{\sum_{k \in V} \langle \lambda, \phi_1(x_k, \bar{y}_k^\sigma)\rangle + \sum_{kl \in E} \langle \lambda, \phi_2(\bar{y}_k^\sigma, \bar{y}_l^\sigma)\rangle\right\}} \tag{6}$$

$$= \frac{\exp\left\{\langle \lambda, \phi_1(x_i, y_i) + \sum_{k \in \mathcal{N}(i)} \langle \lambda, \phi_2(y_k, y_i)\rangle\right\}}{\sum_{\sigma \in \Sigma} \exp\left\{\langle \lambda, \phi_1(x_i, \sigma) + \sum_{k \in \mathcal{N}(i)} \langle \lambda, \phi_2(y_k, \sigma)\rangle\right\}} \tag{7}$$

A heuristic test for convergence can be implemented by testing whether the highest-scoring values of the latent variables in several parallel sampling chains remain constant over many iterations. Updating all $|Y|$ variables once is in $\mathcal{O}(|E||Y|^2)$.

5 Empirical Evaluation

In this section, we evaluate SVMs with the exact and approximate inference strategies described in Section 4 in terms of performance, convergence, and execution time. We experiment on the WebKB, Cora, and the Reuters21578 data sets.

In order to evaluate the impact of the size of the graphs on execution time for WebKB and Cora, we generate training instances as follows. Given a maximal graph size m and a data set, we draw a node randomly without replacement from the data. We iteratively add nodes from the neighborhood of the actual graph until the maximal number of nodes is reached or there are no more nodes that can be added. By doing so, we generate graphs of sizes between one and m. We discard singleton graphs and draw training, parameter tuning, and holdout sets randomly from the remaining instances. We employ $m = 2, 4, \ldots, 20$. To guarantee a fair comparison, we make sure that for every data set, all training, tuning, and holdout sets contain on average the same number of documents.

As additional baselines, we include structural multi-class SVMs (mc) [2, 13] and a structural SVM that is deprived of all link information (naïve). The main difference between multi-class and naïve SVM is that the former provides slack variables for all documents while the latter relates documents within a graph to the same slack variable [5]. The optimal SVM parameter C is determined for all methods within the interval $[10^{-4}, 10^4]$. When the optimal parameter is at the border of the interval we extend the search appropriately. We report on averages of 100 repetitions with randomly drawn training, tuning, and holdout sets; errorbars indicate standard error.

Loopy belief propagation converges in all our experiments to a stationary distribution. For Gibbs sampling, we employ three chains of length 10000 and use a variance-based criterion to detect convergence.

5.1 WebKB

The WebKB data consist of 8282 web pages from five universities. Every page is labeled with one out of 7 different labels, including *course*, *department*, *student*, and others. We remove tokens with less than four occurrences and employ a bag-of-words representation.

Figure 1 details the error rates for the WebKB data set, where training, tuning, and holdout sets consist on average of 4500 documents, respectively. The two baseline methods, multiclass and naïve, perform worst and almost constantly in the size of the graphs. The exact junction tree together with loopy belief propagation lead to the most accurate prediction models. However, the former could not be computed for graph sizes larger than 8. For loopy belief propagation, we further observe a negative correlation of graph size and error, that is, the larger the graph, the more accurate is the prediction model with loopy belief propagation. Gibbs sampling deteriorates with increasing graph sizes since the number of iterations becomes too small. We will address this issue again in the discussion.

5.2 Cora

The Cora data set consists of 25891 computer science articles of 11 areas including *artificial intelligence*, *machine learning*, *information retrieval*, *databases*, and

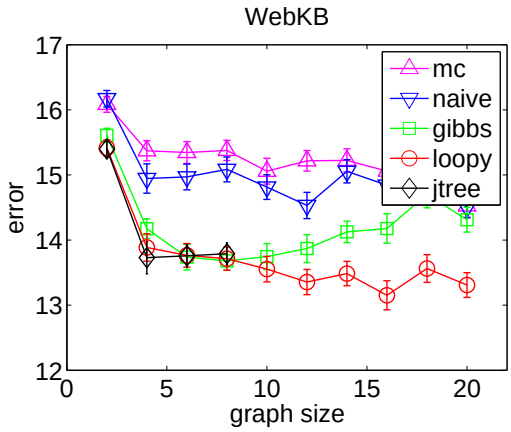

Fig. 1. Results for the WebKB data set.

others. We remove stop words and tokens that occurred less than four times and apply a bag-of-words representation of the input data.

Figure 2 details error rates for different graph sizes. The average number of documents in the training, tuning, and holdout sets is 17300, respectively. SVMs utilizing exact junction tree and approximate loopy belief progagation inference perform nearly identical and lead to the most accurate prediction models. However, for the large Cora data set, junction tree inference is only computable up to graph sizes of six. As expected, the multi-class and naïve baselines that do not utilize link information are unaffected by varying sizes of the underlying graphs and perform worst. The Gibbs sampling performs significantly worse than the junction tree algorithm or loopy belief propagation.

5.3 Reuters21578

The Reuters data set consist of 21578 documents from the Reuters news archive. Documents are classified according to their topic into 120 classes, including *grain*, *oil*, and *trade*. Since the topics interdepend and co-occur frequently in a single document, many documents are assigned with multiple labels. We employ a bag-of-words representation for the documents. We represent the multi-class multiple-label problem by a fully connected graph, consisting of a single observation node (the document) and a binary latent variable for every possible label. A value of one indicates the presence of the corresponding topic in the observed document and a value of zero its absence.

In order to quantify the performance of the methods in terms of the number of labels, we organize the data as follows. We begin with the largest two classes and increment the number of labels until we arrive at the twelve largest classes. Since the underlying graphs are completely connected, the number of considered classes is identical to the graph-size. For each graph size we utilize all 21578

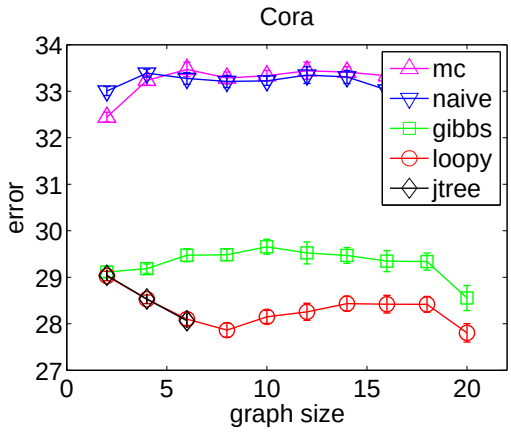

Fig. 2. Results for the Cora data set.

documents that are randomly divided into training, tuning, and holdout sets. All sets are equally sized. We report on averages over 100 repetitions.

Figure 3 shows the results for inference based on naïve, Gibbs sampling, loopy belief propagation, and the exact junction tree algorithm. First of all, every tested method leads to more accurate prediction models when the size of the graph is increased. Moreover, except for loopy belief propagation, all methods perform equally well. As for the junction tree algorithm this is not surprising. The task is also well suited for Gibbs sampling that copes with the small sized graphs and leads to an almost identical performance.

Surprisingly, the naïve approach that does not contain any edge information performs similarly for all numbers of labels. The reason lies in the reduced number of classes in our experimental setup. Focusing on only a subset of all classes implies that the bulk of the documents correspond to discarded labels which carry the dominant information: There are hardly significant co-occurrences of large classes within documents; highly correlated labels involve small classes in the majority of cases. As a consequence, transition probabilities are discarded from the training process and cannot be exploited by the link-based models. A similar observation has been made by Finley and Joachims [3]. In line with their findings is also the performance of loopy belief propagation that is significantly worse compared to the other inference strategies. Loopy belief propagation struggles with the fully connected graph and frequently ends up in local maxima. This is also reflected by the large standard errors.

5.4 Parameter Optimization

The results for the parameter search are shown in Figure 4 that depicts holdout errors across the search interval. For all data sets, the optimal values for the trade-off parameters lie in the right half of the search spaces. For WebKB and

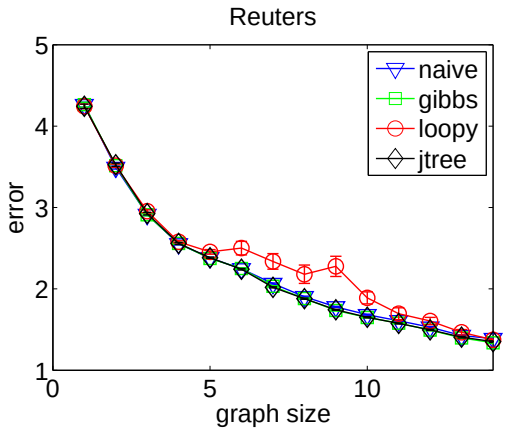

Fig. 3. Results for the Reuters data set.

Reuters, all methods provide a large region in which an almost optimal value can be found, that is, $C^* \gg 1$. The optimal parameters for the Cora data set are found in a narrow interval around $C^* \approx 10$.

5.5 Execution Time

Intuitively, the exact junction tree exhibits the worst execution time for all data sets. The multiclass and naïve baselines that do not exploit link information perform comparably fast across the data sets; both are unaffected by graph sizes. Loopy belief propagation and Gibbs sampling differ significantly in their execution time, although they are both in $\mathcal{O}(|E||Y|^2)$. While loopy belief propagation scales well for increasing graph sizes, Gibbs sampling turns out to be computationally demanding.

For small graphs, the exact junction tree algorithm can be computed with only little additional resources compared to loopy belief propagation. However, the junction tree scales exponentially in the size of the graph and exhibits the worst execution time for moderately sized graphs; exact inference is not feasible for larger structures.

6 Discussion

For two out of three studied tasks, approximate inference with loopy belief propagation is competitive to the exact junction tree and leads to comparable results. We observe a positive correlation between the size of the graphs and the achieved accuracies. For these tasks, the impact of the approximate inference on the SVM algorithm is negligible which can also be verified by monitoring the primal-dual gap. However, loopy belief propagation performs poorly for the Reuters data set.

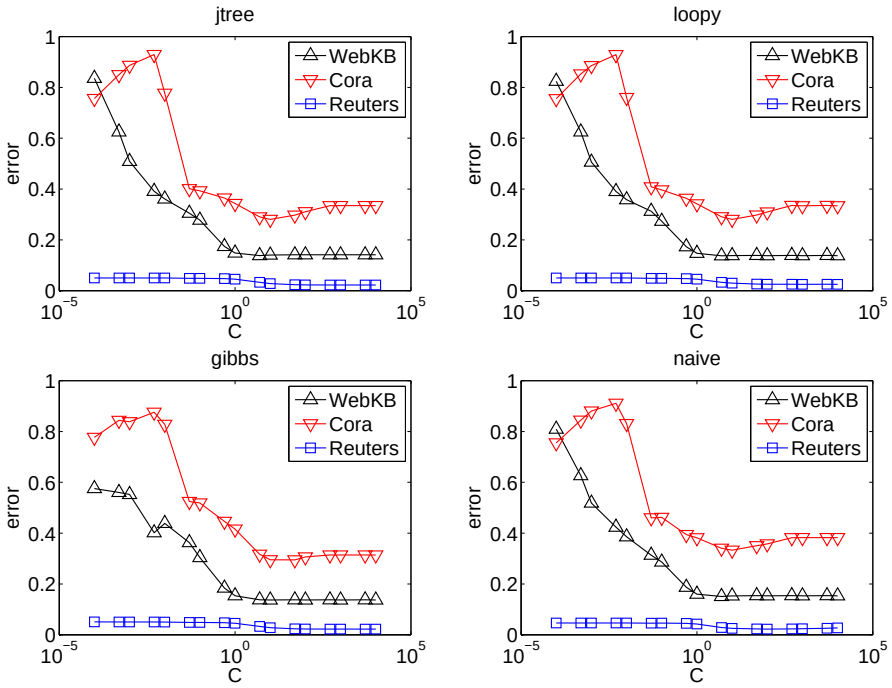

Fig. 4. Results of the parameter search.

Support vector machines with approximate Gibbs sampling are well suited for small graphs but the performance deteriorates quickly when the size of the graphs is increased. Of course, the number of iterations could also be increased to cope with increasing graph sizes but a look at the execution times of Gibbs sampling indicates that it is already demanding in terms of computational time. Increasing the number of iterations would clearly multiply the required ressources.

Inference with the junction tree algorithm consistently leads to the most accurate prediction models but due to its computational complexity, junction tree algorithms can only be applied to small graphs. However, the accuracies achieved by the junction tree algorithm for small graphs is hardly beaten by other methods. Thus, to circumvent the computational dead end, our findings indicate that it is often beneficial to split large graphs into smaller components for which the junction tree algorithm can be computed. The support vector optimization algorithm allows the inclusion of several training instances and thus balances the discarded information.

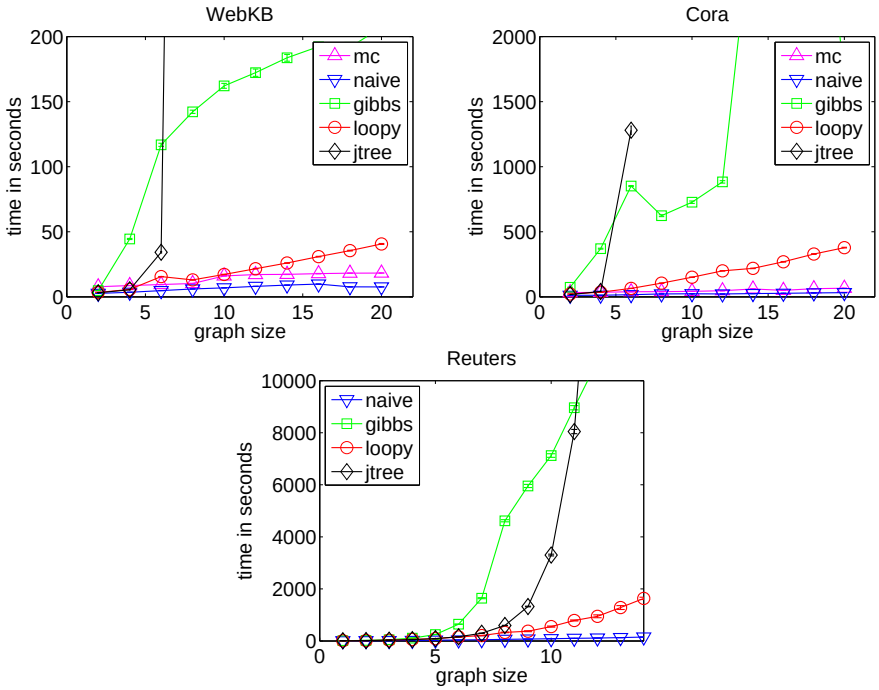

Fig. 5. Execution time for Reuters (left), Cora (center), and WebKB (right).

7 Conclusion

In this paper, we compared exact and approximate inference strategies for labeling graphs with support vector machines. We studied the exact junction tree algorithm and approximate inference with loopy belief propagation and Gibbs sampling. Our findings showed that the exact junction tree inference leads to the most reliable and to the most accurate prediction models in our discourse area. By contrast, the tested approximate inference strategies performed throughout inconsistently and led to poor predictions on some data sets.

To remedy the computational barrier of the junction tree, our results showed that decomposing large graphs into smaller subgraphs is beneficial in several ways: Including the subgraphs as additional training examples in the learning process not only rendered exact inference feasible but also preserved reliable results.

Acknowledgements This work was supported in part by the German Science Foundation DFG and by the FP7-ICT Programme of the European Community, under the PASCAL2 Network of Excellence, ICT-216886. We are grateful to

Steffen Bickel, Laura Dietz, and Klaus-Robert Müller for valuable discussions and comments.

References

1. Robert G. Cowell, Philip, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems (Information Science and Statistics)*. Springer, May 2003.
2. K. Crammer and Y. Singer. On the algorithmic implementation of multi-class kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.
3. T. Finley and T. Joachims. Parameter learning for loopy markov random fields with structural support vector machines. In *ICML Workshop on Constraint Optimization and Structured Output Spaces*, 2007.
4. S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.
5. T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning*, 2005.
6. J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: representation and clique selection. In *Proceedings of the International Conference on Machine Learning*, 2004.
7. K. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, 1999.
8. J. Neville and D. Jensen. Collective classification with relational dependency networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2003*, 2003.
9. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, Ca., 1988.
10. Prithviraj Sen and Lise Getoor. Empirical comparison of approximate inference algorithms for networked data. In *ICML Workshop on Statistical Relational Learning (SRL)*, 2006.
11. B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2002.
12. B. Taskar, C. Guestrin, and D. Koller. Max–margin Markov networks. In *Advances in Neural Information Processing Systems*, 2004.
13. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.