Latent Dynamic Model with Category Transition Constraint for Opinion Classification

Takeshi S. Kobayakawa
NHK / 1-10-11 Kinuta, Setagaya-ku, Tokyo 157-8510, JAPAN
kobayakawa.t-ko@nhk.or.jp

Abstract

Latent models for opinion classification are studied. Training a probabilistic model with a number of latent variables is found unstable in some cases; thus this paper presents how to construct a stable model for opinion classification by constraining classification transitions. The baseline model is a CRF classification model with plural latent variables, dynamically constructed from the dependency parsed tree. The aim of the baseline model is to have each latent variable convey a partial sentiment of the input sentence which is not explicitly given in the training data, and the complete sentiment of the sentence is computed by summing up such partial sentiment where those latent variables hold. Since such a conventional model has many degeneracies in principle, a model with a category transition constraint is proposed, which is expressed by a novel penalty term in the objective function for training the model. The constraint is such that the sentiment of a partial sentence more likely propagates to the same sentiment of the complete sentence, rather than to another sentiment. The effectiveness and the robustness of the proposed model are confirmed by the experiments on binary as well as multi-class opinion classification task.

1 Introduction

Opinion classification is a task to classify sentences into given categories, according to sentiment, evaluation, or some opinion-related points of view. A practical implementation of opinion classification would be very useful for managing customer relationships at contact centers, etc. The classification problem may be binary or sometimes multi-class.

One of the simplest modeling process is to use explicit bag features, such as word surfaces, polarity information from the sentiment dictionary, etc. Thanks to the good behavior of the Maximum Entropy or the Conditional Random Field (CRF) model, the maximum likelihood training is straight-forward, because the local optimum is always the global optimum.

A challenge is to introduce into the model latent variables, which are not explicitly observable. The implicit modeling here is supposed to express ambiguities of natural language; the partial sentiment of the sentence is not determined until the end of the sentence. This paper presents in detail a probabilistic model with latent variables. The baseline model is a CRF model which is constructed dynamically according to the dependency parsed tree and which contains latent variables on the nodes that correspond to the chunked expressions (Nakagawa et al., 2010). The latent variables in the model are expected to convey a partial sentiment of the sentence, such as the sentiment of the dependency-parsed-subtree itself, which is not explicitly observable.

Although this idea is attractive, it actually suffers from numerical instability. Our aim here is to find a way to deal with this problem. We tried using a global optimizer and investigated the behavior of the model, to ensure that this lack of stability comes from the degeneracy of the model.

Our contribution to remedy this problem is as follows: We propose a model with a penalty on category transitions and compare several optimizers to train the model. We also confirm the stability of the model

Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 1350–1359, Dublin, Ireland, August 23-29 2014.

by applying it to the multi-class classification problem. We also investigate the origin of the degeneracy and how regularizer works for this type of classification model, to see the robustness of the model.

2 Related Work

Studies on opinion classification have more than a decade of history, including the pioneer work (Turney, 2002; Pang et al., 2002), followed by (Takamura et al., 2005; Brun, 2012). Our baseline model treats the sentiment of a partial sentence (Nakagawa et al., 2010); CRF with latent variables are constructed dynamically by the dependency structure tree of the input sentence.

CRF model was first used in sequence labeling tasks (Lafferty, 2001). The model does not suffer from the label-bias problem as does the Maximum Entropy model, and its parameter estimation is well behaved with the help of a convex loss function. However, the convexity of the loss function of the CRF model does not hold anymore when there are unobserved data or latent variables (Sutton and McCallum, 2007).

Latent variables were first used with CRF for the purpose of noun coreference (McCallum and Wellner, 2005), and object recognition (Quattoni et al., 2005). Latent variables have been used to construct meaning representations in a process called grounded language acquisition (Liang et al., 2009). Another approach with hidden variables has been used an recursive auto-encoder to reduce the reliance on sentiment dictionaries (Socher et al., 2011).

Machine learning, used for training such models, is largely based on the concept of numerical optimization. A general discussion of convex optimization can be found in (Boyd and Vandenberghe, 2004); it can be proven that the `log-sum-exp` type of a convex function is still a convex function. This is the situation with the CRF model without latent variables. Since convexity holds, the best solution to the problem would be a numerical local optimization. A general discussion of local optimization can be found in the textbook (Nocedal and Wright, 2006), who is one of the authors of the Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimizer program. Many NLP application programs use probabilistic models, including this optimizer or its derived work. As the long name of the BFGS algorithm shows, the state-of-the-art local optimizer has a long history. Another textbook (Press et al., 2007) covers a global optimization algorithm, Simulated Annealing, originating from (Kirkpatrick et al., 1983). The main idea comes from the statistical physics of equilibrium; when a material is warm, its energy is distributed in excited states, whereas when it is cooled, it is very probable that the system will end in the ground state, which corresponds to the global minimum point of the energy.

Previous studies on CRF with latent variables were trained either by setting the initial parameters randomly (Nakagawa et al., 2010), or by online training (McCallum and Wellner, 2005). As far as the author knows, this is the first study to impose a penalty between latent variables in CRF model, and to compare several optimization algorithms.

3 The Model

The model studied is a CRF model, which has set of conditional probabilities whose log is a linear combination of model parameters associated with features given by (Lafferty, 2001):

$$\log p_\Lambda(\mathbf{y}|\mathbf{x}) \propto \sum_{v\in V,k} \mu_k g_k(v, \mathbf{y}|_v, \mathbf{x}) + \sum_{e\in E,k} \lambda_k f_k(e, \mathbf{y}|_e, \mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is an input vector and $\mathbf{y}$ is an output label sequence, and $\mathbf{y}|_v$ and $\mathbf{y}|_e$ are the vertex v and the edge e related to the component of $\mathbf{y}$, respectively. The model parameter vector Λ is estimated from the training data, whose components are the sum of the two sets: a vertex feature set $(\mu_1, \mu_2, \dots)$ and an edge feature set $(\lambda_1, \lambda_2, \dots)$. The complete set of features are supposed to be enumerated and fixed, so that each feature can be indicated by an index k in a rather relaxed way. g_k and f_k are so-called feature functions, to indicate whether the feature in the argument appears in the input, or not.

Nakagawa et al. (2010) proposed a CRF model with latent variables that uses a dependency parsed structure, where the latent variables are expected to convey the sentiment classifying the part underneath the parsed tree. We choose this model as a baseline and continue the same kind of treatment of the latent

variables. Section 3.1 and 3.2 briefly review this baseline model, then the proposed model follows in Section 3.3.

3.1 Sentence Classification Model with Latent Variables

To classify sentence $\mathbf{x}$ into a given set of a class, say C, the classification problem is formulated as:

$$\arg\max_{s_0 \in C} p_\Lambda(s_0|\mathbf{x}), \tag{2}$$

where s_0 is a class label of the complete sentence and $\mathbf{x}$ is a given sentence such that

$$p_\Lambda(s_0|\mathbf{x}) = \sum_{\mathbf{s}} p_\Lambda(s_0, \mathbf{s}|\mathbf{x}), \tag{3}$$

where $\mathbf{s}$ is the latent variables to be summed up, and Λ is the set of all parameters in the model. Each element of $\mathbf{s}$ takes one of these class labels, corresponding to the partial sentiment of the sentence, and is to be summed up to construct the complete sentiment of the sentence. A partial sentiment of a sentence is sometimes ambiguous, so it is treated as such an unobserved variable. The model parameters Λ are estimated from the training data; the sentiment label is only available for a whole sentence, not for a part of the sentence.

3.2 Dependency Structure as a Graphical Model

The given sentence $\mathbf{x}$ is parsed into a dependency structure of phrase chunks:

$$\mathbf{x} \xrightarrow{\text{Dependency Parsing}} G(\mathbf{x}) = \{V(\mathbf{x}), E(\mathbf{x})\}, \tag{4}$$

where $V(\mathbf{x})$ are the set of chunks and $E(\mathbf{x})$ is the set of dependency arcs. The dependency structure is regarded as a graphical model, an example of which is shown in Figure 1. The words are chunked up into phrases, and the dependencies between those chunks are determined by dependency parsing. Each chunk corresponds to a variable that is supposed to convey a sentiment.

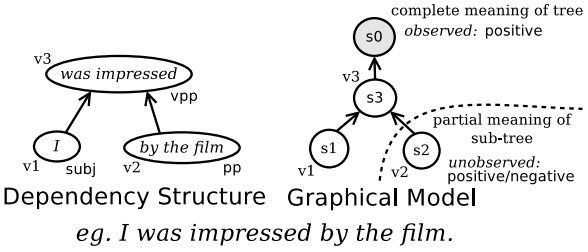

Figure 1: Correspondence between dependency structure and graphical representation of the model

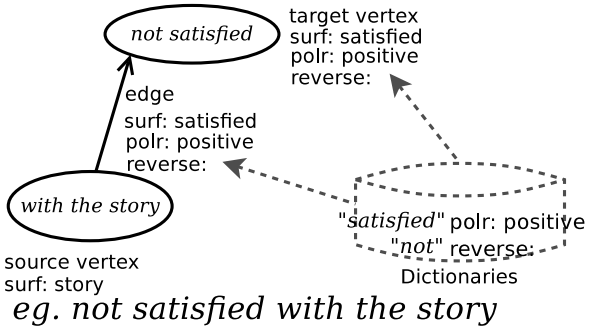

Figure 2: Features attached to vertices and edges

Vertex features	Edge features
word surface unigrams	word surface unigrams of the parent vertex
succeeding word surface bigrams	word surface unigrams of the child vertex
sentiment information from a dictionary	sentiment information from a dictionary
negation expression from a dictionary	negation expression from a dictionary
meaning label of functional expression	meaning label of functional expression of the parent vertex
	meaning label of functional expression of the child vertex

Table 1: Summary of features adopted in the model

Every feature belongs to one of two types: vertex features or edge features. Vertex features are those locally related to a vertex, and edge features are those that affect both sides of vertices of the dependency. The specific features adopted in the model are summarized in Table 1.

A symbol μ as a notation for model parameters related to vertex features. It has two indices since they are related to a vertex feature and a classification category. As for the vertex feature f_v related part, the log probability that the vertex v has the category s is:

$$\log p_{f_v}(s) = \mu_{f_v,s}. \tag{5}$$

A symbol λ as a notation for model parameters related to edge features. It has three indices since they are related to an edge feature and classification categories of both sides of vertices of the edge.

As for the edge feature f_e related part, the log conditional probability that the target vertex takes the category s_2 is:

$$\log p_{f_e}(s_2|s_1) = \lambda_{f_e,s_2,s_1}, \tag{6}$$

given the category of the source vertex is s_1.

Multiple features can be attached either on a vertex or on a edge. So, the whole vertex or edge probability is constructed as follows:

$$\log p_v(s) = \sum_{f_v \in F^{(\text{vertex})}(v)} \log p_{f_v}(s) = \sum_{f_v \in F^{(\text{vertex})}(v)} \mu_{f_v,s}, \tag{7}$$

$$\log p_e(s_2|s_1) = \sum_{f_e \in F^{(\text{edge})}(e)} \log p_{f_e}(s_2|s_1) = \sum_{f_e \in F^{(\text{edge})}(e)} \lambda_{f_e,s_2,s_1}, \tag{8}$$

where $F^{(\text{vertex})}(v)$ is a set of features attached to a vertex v, and where $F^{(\text{edge})}(e)$ is those attached to an edge e.

Finally, the probability of a given sentence $\mathbf{x}$ is constructed as a log-linear model:

$$\log p(s_0, \mathbf{s}|\mathbf{x}) = \sum_{v \in V(\mathbf{x})} \log p_v(s^{(\text{v})}) + \sum_{e \in E(\mathbf{x})} \log p_e(s^{(\text{target}(e))}|s^{(\text{source}(e))}), \tag{9}$$

where the set of vertices and edges are dynamically constructed from the dependency parsed tree of the sentence, *i.e.* eq. (4), and the notation source(e) and target(e) are the source and target vertex of the edge e, respectively (Figure 2.)

Care is necessary when assigning values to the latent variables in eq. (9). Because each latent variable which is assigned on a vertex and the connecting edges, share the same values, the latent variables must be summed up in such way; The summations can be done efficiently by using dynamic programming (*a.k.a.* the factor graph in graphical model terminology.) The tables of probabilities are constructed for each vertex, and the tree is constructed in a bottom up manner.

The sets[1] of all the vertex and edge features appearing in the training data $\mathcal{D}$ are denoted as $\mathcal{V}^{(\mathcal{D})}$ and $\mathcal{E}^{(\mathcal{D})}$ respectively:

$$\mathcal{V}^{(\mathcal{D})} = \sum_{\mathbf{x} \in \mathcal{D}} \sum_{v \in V(\mathbf{x})} F^{(\text{vertex})}(v), \qquad \mathcal{E}^{(\mathcal{D})} = \sum_{\mathbf{x} \in \mathcal{D}} \sum_{e \in E(\mathbf{x})} F^{(\text{edge})}(e), \tag{10}$$

so that the complete set of parameters is:

$$\Lambda = \{\mu_{v,c} | v \in \mathcal{V}^{(\mathcal{D})}, c \in C\} + \{\lambda_{e,c_1,c_2} | e \in \mathcal{E}^{(\mathcal{D})}, \quad c_1, c_2 \in C\},$$

where the number of parameters is $\mathcal{V}^{(\mathcal{D})} \times C + \mathcal{E}^{(\mathcal{D})} \times C \times C$.

[1] Note that the following equations up to eq. (11) are in the terminology of set theory; the addition is done with the elimination of duplicated elements, and a product means a direct product, and a n-th power is an abbreviation of n direct products of the set.

The log likelihood of the training data is given by

$$\mathbb{L}(\Lambda;\mathcal{D}) = \sum_{\mathbf{x}\in\mathcal{D}} \log\left(\sum_{\mathbf{s}\in C^{|V(\mathbf{x})|}} p_\Lambda(s_0,\mathbf{s}|\mathbf{x})\right), \tag{11}$$

where $|V(\mathbf{x})|$ is the number of vertices in a given sentence $\mathbf{x}$, and where s_0 is the correct classification label for $\mathbf{x}$, and where $\mathbf{s}$ is the set of latent variables whose number is as many as $|V(\mathbf{x})|$.

3.3 Category Transition Penalty

We found that the classification accuracy of the trained model remained low, because little number of parameters for edge features moved away from initial non-contributing values during the training. The following form of regularizer[2] is used for the training:

$$\mathbb{R}(\Lambda) = C_{\text{regularizer}} \left|\Lambda - \frac{1}{n}\cdot\mathbb{1}\right|^2. \tag{12}$$

The constant $C_{\text{regularizer}}$ is the strength of the regularizer, and no matter how strong, the classification accuracy did not improve in our preliminary experiments. This phenomenon seems to be because of the degeneracies the model has in principle.

Degeneracy is a notion to explain the same probabilities of different configurations. If the two different configurations are preferably distinguished, an asymmetric treatment of them is required.

In order to avoid such extra degeneracies, we introduce a novel constraint between latent variables expressed by the following penalty term:

$$\mathbb{P}(\Lambda) = C_{\text{penalty}} \sum_{f_e\in\mathcal{E}^{(\mathcal{D})}} \left(\sum_{s_1=s_2} (\log p_{f_e}(s_2|s_1) - \log C_{\text{same}})^2 + \sum_{s_1\neq s_2} (\log p_{f_e}(s_2|s_1) - \log C_{\text{different}})^2\right), \tag{13}$$

to satisfy

$$C_{\text{same}} + (n-1)C_{\text{different}} = 1, \tag{14}$$

where C_{penalty} is the weight of this penalty, and where C_{same} and $C_{\text{different}}$ are constant probabilities for the following two cases; that is the categories connected to the other side of the edge should be the same or different, respectively. This term is incorporated into the objective function for training the model, to form a soft constraint that diminishes the change in the classification category.

3.4 Model Training

The maximum likelihood training of a probabilistic model is a constrained optimization problem. A probabilistic interpretation is possible if and only if 1) all probabilities are non-negative, and 2) the sum of the probabilities are one (or renormalizable to one).

Using log probabilities almost automatically satisfies the first condition: Real number in log space corresponds to positive number in anti-log space, so that the only consideration needed is zero probability (which corresponds to negative infinity in the log space.) In the experiments, overflow is checked that none occurred in the final results. The model parameter to express zero probability could be finite but reasonably small, instead of zero.

As for the second condition, instead of the strict constraint, we adopted a quadratic penalty in log space:

$$\mathbb{C}(\Lambda) = C_{\text{prob}} \left(\sum_{f_v\in\mathcal{V}^{(\mathcal{D})}} \left(\log\sum_{s\in C} p_{f_v}(s)\right)^2 + \sum_{f_e\in\mathcal{E}^{(\mathcal{D})}} \sum_{s_1\in C} \left(\log\sum_{s_2\in C} p_{f_e}(s_2|s_1)\right)^2\right), \tag{15}$$

[2]The offset $\frac{1}{n}\cdot\mathbb{1}$ in the regularizer is so as to avoid singularity around zero probabilities in real space, which causes negative infinity in log space calculation.

where the strictly normalized probabilities lead to zero penalty; otherwise, a quadratic penalty is given according to the amount away from strictly normalized probabilities. The weight of the penalty C_{prob} is chosen to be heavy enough for the sum of the model probability to be adequately normalized.

Finally, we adopted the probability calculation in the log space, with the quadratic penalty for normalization during the training. The objective function for training the model is:

$$\mathbb{O}(\Lambda; \mathcal{D}) = \mathbb{L}(\Lambda; \mathcal{D}) - \mathbb{R}(\Lambda) - \mathbb{P}(\Lambda) - \mathbb{C}(\Lambda). \tag{16}$$

4 Experiments

Experiments are conducted to evaluate the effect of the proposed penalty term expressed by eq. (13).

4.1 Test Sets

Two kinds of test set in Japanese were used: Opinions in the Kyoto University and NTT Blog (KNB) Corpus[3]and Comments on an TV cultural program. Both sets are balanced in the numbers sentence categories. The characteristics of each test set are shown in Table 2.

The first test set, the KNB Corpus, is a collection of opinion sentences about Kyoto sightseeing spots, cellular phones, gourmet food, and sports. The sentences are categorized in terms of many aspects, and we used the sentences labeled with Evaluation+ or Evaluation–. "Evaluation" is a category of subjective but non-emotional opinions.

The second test set is used for non-binary classification. To make this set, viewers were asked to comment (in Japanese natural language) on a certain TV program. The comments are classified into categories, *i.e.* evaluations, impressions, requirements and questions. The following four categories are used: positive and negative evaluations, and impressions of what the viewers learned from the program, and what they thought after watching the program.

Name of Test Set	# of Sentences	Categories
Opinions in KNB Corpus	328	2(Evaluation +/–)
Comments on TV cultural program	432	4(positive/negative evaluations, what viewers learned/think)

Table 2: Characteristics of Test Set

4.2 NLP Resources

The input sentence was processed by a morphological analyzer to split it into words, since Japanese is an agglutinative language. The words were then chunked and the dependencies between those chunks were determined. Functional multi-word expressions were also detected by the analyzer we developed. We used a dictionary of sentiment expressions, and one of negation expressions, both of which were distributed with the KNB Corpus.

We did not prepare any special sentiment dictionary for the second test set because preparing such a dictionary is too costly. Furthermore, robustness can be estimated without a domain-adapted dictionary. The parameter values for training the models were tuned for the first test set, and the tuned values were used without any extra tuning for the second test set. In this situation, the first test set can be regarded as the development test set, and the second as the evaluation test set.

4.3 Latent Dynamic Model with Category Transition Constraint

Experiments on classifying opinions using the KNB Corpus are shown in Table 3. The rightmost column is a trivial baseline, where the classification category is decided by the majority occurrence of sentiment

[3]The corpus is publicly available from `http://nlp.ist.i.kyoto-u.ac.jp/kuntt/\#ga739fe2`, and the details of corpus are explained at `http://alaginrc.nict.go.jp/opinion/index_e.html`. We excluded short sentences, made up of a few words, that were exclamations rather than natural complete sentences. They do not form tree structures, which are not aimed to this study. Accuracy of experiments conducted below are different from that by (Nakagawa et al., 2010) in that only the subset of the test set that satisfy the condition is used.

words in the sentence. This method needs no training, so only one figure is indicated. The other columns are figures for trained CRF; closed tests are the case where all the training data is used for the evaluation as well, shown in the upper row, while 10-folds open test are the case is 1:9 split of data for evaluation and training and the evaluation is done cyclically 10 times, shown in the lower row. The leftmost column is CRF without latent variables. The 2nd left is a model with latent variables but without penalty, which is the model in (Nakagawa et al., 2010). The 3rd left is a model with latent variables that has a category transition penalty, which is the proposed model. The proposed model performs the best accuracy among all the models.

Experiments on classifying opinions using the Comments on the TV cultural Program are shown in Table 4. Majority voting is not possible because a suitable dictionary that has the same class polarity as this classification problem does not exist.

	Trained CRF			(no training)
	without latent variables	with latent variables		Majority
		non-penalized	penalized	Voting
closed test	95.12	95.73	95.73	
10-fold open test	61.59	63.72	**65.55**	64.79

Table 3: Effect of Latent Variables and Penalized Model (Opinions in KNB Corpus)

	Trained CRF			(no training)
	without latent variables	with latent variables		Majority
		non-penalized	penalized	Voting
closed test	99.54	99.77	99.31	
10-fold open test	60.42	60.42	**64.81**	N/A

Table 4: Effect of Penalized Latent Dynamic Model (Comments on TV cultural program)

4.4 Comparison of Optimizers

Three optimization algorithms for model training were compared, two of which are local optimizers (BFGS and Steepest Descent), and one of which is a global optimizer (Simulated Annealing).

BFGS was used as batch training where all of the training data were used during the training iteration. Two types of initial parameter configurations were tried for BFGS; initial parameters have the same fixed values, or were chosen randomly. Steepest Descent (SD) was used as online training where some portion (*i.e.* chunk) of the training data were used during an iteration. Two types of chunk selection scheme were tried for Steepest Descent; chunks were fixed during the training, or chunks were randomly shuffled after every complete loop of the whole training data.

Simulated Annealing was adopted as a global optimizer. We implemented feature level granularity for acceptance or rejection: Every parameter corresponding to a feature was randomly moved, and decided probabilistically whether or not to accept according to the Boltzmann distribution under scheduled cooling down. Although all of these methods utilize random variables, they were used in different ways. The accuracy ranges of ten trials are shown in Table 5 and 6.

The results show that the proposed model trained by a global optimizer outperforms models trained by the other local optimizers (Steepest Descent and BFGS); The penalty in the proposed model works well because the degeneracies in the penalized model seem to decrease, and the computation is noteworthy stable.

5 Discussion

The degeneracy in the model is illustrated in Figure 3 and 4.

Firstly, convergence for penalized model is quicker, as shown in Figure 3; The horizontal axis is the number of iterations, and the left vertical axis is the acceptance ratio, where the lower is well converged.

	Batch Training (BFGS) parameter initialization		Online Training (SD) chunked data selection		Simulated Annealing
	random	*fixed*	*shuffled*	*fixed*	
closed test	89.33-82.32	87.80	76.83-72.56	76.83	95.73-95.73
10-fold open test	62.80-58.54	59.15	66.46-65.55	65.55	65.55-64.63

Table 5: Comparison of Optimizers (Opinions in KNB Corpus)

	Batch Training (BFGS) parameter initialization		Online Training (SD) chunked data selection		Simulated Annealing
	random	*fixed*	*shuffled*	*fixed*	
closed test	85.65-35.65	88.19	47.69-45.37	45.60	99.31-99.31
10-fold open test	58.56-34.72	59.95	38.89-36.81	37.50	**64.81-64.58**

Table 6: Comparison of Optimizers (Comments on TV cultural program)

The acceptance ratio remains high for the non-penalized model, while the penalized model quickly descends. The dash line indicates the log likelihood of the training data, for penalized and non-penalized models, which are almost identical.

Secondly, in order to split degeneracy, only a small penalty is adequate, as illustrated in Figure 4; how less the penalty is, as long as it exists, the improvement remains. The horizontal axis is the strength of the proposed penalty, C_{penalty} in eq.(13). The vertical axis is the classification accuracy. The dash line is a line fit for the accuracy by non-zero penalty. In general, such an extra constraint term for the original model may change the model itself, so, the less it is the better. That is the reason for decreasing accuracy when large penalty is used. The significant jump in the accuracy between zero and non-zero penalty strongly suggests the existence of degeneracies in the original model: Infinitesimally small penalty can lead to break those degeneracies.

Local optima usually do not matter when regularizers are used in training. However, according to our experiments in this type of models, the conventional regularizers are not able to avoid such local optima no matter how strong they are. The reason the introduced penalty works well for this model is considered that the term works for excessive latent variables, which are not controlled by the ordinary regularizers. The ordinary regularizers only works for excessive number of explicitly observed parameters (*i.e.* features). If only a few latent variables are used, such a penalty is not necessary, just as a regularizer is not necessary for a small number of features. When the model is constructed dynamically and the number of latent variables grows, there appear a number of latent variables having excessive freedom, which need to be controlled.

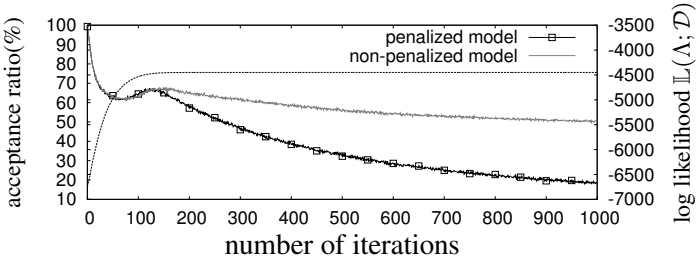

Figure 3: Transition of the Acceptance Ratio

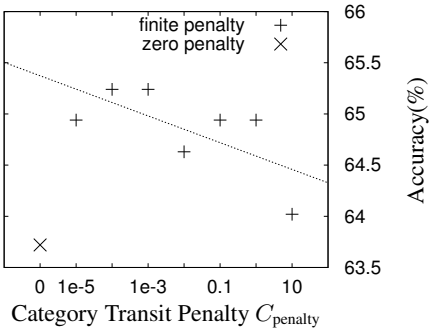

Figure 4: Effect of Weak Penalty

6 Conclusion

A latent dynamic model with a category transition constraint is proposed for opinion classification task. The constraint is such that the sentiment of a partial sentence tends to propagate toward the complete sentiment of the whole sentence, which is realized, in the objective function, by our novel term that penalizes, diminishing the change in the classification category.

According to our experiments, the penalized latent dynamic model outperforms the conventional model, not only in binary but also in multi-class opinion classification.

The comparison of optimizers strongly suggests that the degeneracies in the conventional model deteriorate the performance, and the proposed model solves such a defect. The numerical stability of training the proposed model is also confirmed.

Acknowledgments

The author would like to thank to Jun'ichi Tsujii, Akiko Aizawa, Yusuke Miyao, Takuya Matsuzaki, Hideki Tanaka and Tadashi Kumano. The author would also like to express the greatest gratitude for the discussions with Akio Kobayashi, Hiroyuki Segi, Tsuneo Hirano, Clara K. Hirano, and Mariko Hirano. Mariko Hirano also helped some of the experiments.

References

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Caroline Brun. 2012. Learning opinionated patterns for contextual opinion detection. In *Proceedings of COLING 2012: Posters*, pages 165–174, Mumbai, India, December. The COLING 2012 Organizing Committee.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220(4598):671–680.

John Lafferty. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, pages 282–289. Morgan Kaufmann.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Suntec, Singapore, August. Association for Computational Linguistics.

Andrew McCallum and Ben Wellner. 2005. Conditional models of identity uncertainty with application to noun coreference. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 905–912. MIT Press, Cambridge, MA.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794, Los Angeles, California, June. Association for Computational Linguistics.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization*. Springer Verlag, 2nd edition.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86. Association for Computational Linguistics, July.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical Recipes*. Cambridge University Press, 3rd edition.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2005. Conditional random fields for object recognition. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1097–1104. MIT Press, Cambridge, MA.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Charles Sutton and Andrew McCallum, 2007. *Introduction to Statistical Relational Learning*, chapter 4 An Introduction to Conditional Random Fields. The MIT Press. also available on e-Print archive: arXiv:1011.4088v1.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 133–140, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.