Blog Comments Classification using Tree Structured Conditional Random Fields

by

Wei Jin

B.Cs., The University of British Columbia, 2010

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

November 2012

Abstract

The Internet provides a variety of ways for people to easily share, socialize, and interact with each other. One of the most popular platforms is the online blog. This causes a vast amount of new text data in the form of blog comments and opinions about news, events and products being generated everyday. However, not all comments have equal quality. Informative or high quality comments have greater impact on the readers opinions about the original post content, such as the benefits of the product discussed in the post, or the interpretation of a political event. Therefore, developing an efficient and effective mechanism to detect the most informative comments is highly desirable. For this purpose, sites like Slashdot, where users volunteer to rate comments based on their informativeness, can be a great resource to build such automated system using supervised machine learning techniques.

Our research concerns building an automatic comment classification system leveraging these freely available valuable resources. Specifically, we discuss how comments in blogs can be detected using Conditional Random Fields (CRFs). Blog conversations typically have a tree-like structure in which an initial post is followed by comments, and each comment can be followed by other comments. In this work, we present our approach usingTree-structured Conditional Random Fields (TCRFs) to capture the dependencies in a tree-like conversational structure. This is in contrast with previous work [5] in which results produced by linear-chain CRF models had to be aggregated heuristically. As an additional contribution, we present a new blog corpus consisting of conversations of different genres from 6 different blog websites. We use this corpus to train and test our classifiers based on TCRFs.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my parents, Enlai Jin and Jingyuan Shao, for supporting me throughout my education, helping me all the way through my masters, and giving me the freedom of making my own choices. I would also like to thank my wife Xinyi Zhu for standing by me through my endeavour through graduate school. She is always there for me no matter what.

I would like to thank Giuseppe Carenini and Raymond Ng for being supportive supervisors throughout my research career at UBC. They have given me the freedom to follow my ideas while providing guidance to keep me on track. Our entire research group has been instrumental throughout my work.

This thesis work is definitely a group effort and I could not have done it alone.

Chapter 1

Introduction

The Internet provides a variety of ways for people to easily share, socialize, and interact with each other. One of the most popular platforms is the online blog. This causes a vast amount of new text data in the form of blog comments and opinions about news, events and products being generated everyday. People using online forum are often overwhelmed by the huge amount of information they are dealing with. They need an efficient way to help them reduce the information overload. Methods to accurately classifying the quality of a blog comment can be used to make various part of viewing online blog comments easier. For example, we can highlight those high quality comments, so people can just read those to get a pretty good grasp about the topic and the content of this conversation. Therefore, we want to have a tool to accurately classify the comments in a given blog conversation. In this thesis we present an approach based on Tree structured Conditional Random Fields.

1.1 Motivations of blog comments classification

Often people face huge amount of comments when they are browsing an online blog form. However, not all comments are equally informative. The most informative or high quality comments should have greater impact on the readers opinions about the original post content, such as the quality of the product discussed in the post, or the interpretation of a political event. Comments classification into high-quality vs. low-quality comments can help people view the online blog more efficiently. For example, high quality comments of a given blog conversation can be selected. In this way a user can just read the selected part of the conversation so that the reading

time will be greatly reduced.

Another useful way to apply comments classification would be to use the selected high quality comments to generate a summary of the whole discussion. People can use this summary to make decisions without spending too much time on viewing the whole discussions. For example, a potential smartphone buyer can get a summary of the reviews submitted by past customers for a given model of phone, so he could get a general idea about the price, quality of product and many other informations about this smartphone.

Therefore, developing an efficient and effective mechanism to detect the most informative comments is highly desirable.

1.2 Tree Structured Conditional Random Fields Blog Comments Classification

In this thesis we present an approach to classify the quality of blog comments using Conditional Random Fields (CRFs). CRFs are undirected graphical models that can exploit a large numbers of arbitrary local and global features in predicting labels for a given observation. For many NLP tasks, the underlying graphical model is usually implemented as a linear chain [5], where observations come in a linear sequence and dependencies exist between two consecutive labels. However, since the structure of blog comments is often a tree, we propose to apply Tree Structured Conditional Random Fields (TCRFs) to take advantage of the dependencies across hierarchically laid-out information. The main challenges we faced in applying TCRFs to our work are: (1) Data collection (2) Model implementation (3) Feature Selection (4) Result Analysis.

1.3 Outline of the thesis

In this work, we describe the process of creating a new corpus of 6 different blog sites for blog classification. We then develop a more advanced CRFs approach to take advantage of the underlying tree structure of the blog

conversation. At last, we test TCRFs approach with different feature sets on the corpus we have created to see the improvement with the respect to classification accuracy.

The thesis comprises six chapters. The first chapter, this one, has introduced the nature of the work and has provided the motivation. The second chapter provides background in the field of comments classification, previous application of CRFs and introduces the work our approach is improving upon. In Chapter 3 we describe the new corpus and the effort and method of building it. In Chapter 4 we introduce our TCRFs approach. Chapter 5 mainly focus on the experiments and the results of our TCRFs model. Finally, in the last chapter we conclude by recapping the achievements of our work. The bibliography and appendix are at the end of the thesis.

1.4 Contributions

As a preview we outline the main contributions of this thesis here:

- Annotated blog comments corpora are rarely available to the public. This makes research on blog comments classification especially difficult. In addition, in the field of classifying blog comments, there was no public accessible labelled comment datasets. Due to these reasons, We compiled a new corpus comprised of the Slashdot[1] dataset collected by FitzGerald et al[5], and conversations from five other blog websites namely, DailyKos[2], AndroidCentral[3], BusinessInsider[4], Macrumors[5] and TSN[6]. These blog sites cover a variety of genres such as technology, business. politics and sports; they also have different conversation structure (sequential or tree). For example, Slashdot conversations have a tree-like structure; users can directly reply to a given comment, and their reply will be placed underneath that comment in a

[1] http://slashdot.org
[2] http://www.dailykos.com
[3] www.androidcentral.com
[4] www.businessinsider.com
[5] www.macrumors.com
[6] www.tsn.ca

nested structure. On the contrary, comments in conversations from AndroidCentral always form a single linear thread. We will be able to test the generality of our approach on very different blogs; and verify how the performance of our proposal varies with respect to the key properties of these blogs. Since our dataset covers several different domains, we will be able to test domain adaptation techniques in the future work.

- In the previous work done in the field of comments classification using CRFs, the authors use a linear-chain CRFs (LCCRFs) to detect the informative blog comments through the exploration of conversational and topical features. However, their approach has two main limitations: (1) it ignores many hierarchical dependencies between the output tags/labels and (2) the common internal nodes fall in multiple paths, which cause them to be classified multiple time, and possible inconsistent classifications have to be combined heuristically. Our proposed approach applies a TCRF model to better handle the dependencies across the hierarchically structured comments in a blog conversation.

- There has not been a comparison of different CRFs approaches for blog comments quality classification. We have therefore created a framework where we can compare various CRFs models. Through our experiments, we found that our TCRFs approach works better than LCCRFs model. TCRFs outperformed LCCRFs due to the inherent tree structure which captures the semantic and syntactic dependencies better.

- In order to build Tree structured CRFs in GRMM, we created our own FactorTemplate class and supplied a a structure file containing the parent-child pairs information of a given blog conversation trees to the modified Arbitrary CRF classes.

Chapter 2

Related Work

CRFs have been successfully applied to many NLP tasks including Part of Speech (POS) tagging, chunking, syntactic parsing, word segmentation and meeting utterances classification [6]. The application most relevant to our work is presented in [5], where the authors use a linear-chain CRF (LCCRF) to detect informative comments in a blog conversation through the exploration of conversational and topical features. They apply the LCCRF model to each path (i.e., from the root to the leave) of the conversation's tree structure. However, this approach has two main limitations: (1) it ignores many hierarchical dependencies between the output tags/labels and (2) the common internal nodes fall in multiple paths, which cause them to be classified multiple time, and possible inconsistent classifications have to be combined heuristically. Our use of Tree-structured CRFs (TCRFs) is inspired by [15] and [3]. [15] applies a TCRF model to better handle the dependencies across the hierarchically laid-out information on the web in the task of *semantic annotation* and show that TCRF outperforms Support Vector Machines (SVMs) and LCCRFs. According to the work done by [3], in the task of *semantic role labelling*, TCRFs outperform LCCRFs because they better capture the tree-structured semantic and syntactic dependencies between a sentence constituents. Other forms of CRFs such as Dynamic Conditional Random Fields (DCRFs) and Skip-Chain Conditional Random Fields (SCCRFs) are also introduced in this chapter. We only used TCRFs and LCCRFs models in our work. It would be interesting to see if other forms of CRFs can be applied to our project as future work.

2.1 Conditional Random Fields

Conditional random fields (CRFs) is a framework introduced by [7] for building probabilistic model on sequence data to perform tasks such as segmentation sand labelling. CRFs offer serval advantages over other machine learning models like hidden Markov models (HMMs) and maximum entropy Markov models (MEMMs) for these tasks, due to their ability to relax strong independence assumption and avoiding the label bias problem. Two independence assumptions are made by HMMs model: first, it assumes that each state depends only on its immediate predecessor, that is, each state y_t is independent of all its ancestors y_1, y_2, $\cdots$, y_{t-2} given its previous state y_{t-1}. Second, an HMM assumes that each observation variable x_t depends only on the current state y_t. Label bias problem exists in MEMMs model: the transitions leaving a given state compete only against each other, rather than against all other transitions in the model. This causes a bias toward states with fewer outgoing transitions. In the worst case, a state with a single outgoing transition effectively ignores the observation.

2.1.1 Linear Chain Conditional Random Fields

Linear Chain Conditional Random Fields (LCCRFs) is a an special case of CRFs. It defines conditional probability distributions of label sequence given input sequence. These sequences have chain like format.

Linear-chain CRFs have been applied to many natural language processing tasks such as named-entity recognition (NER) [10], feature induction for NER [9], Chinese word segmentation Peng et al [12]. In most recent work, Matthieu and Isabelle [4] created a CRF-based Multiword Segmenter and Part-of-Speech Tagger.

The work most related to our project is done by FitzGerald et al [5]. They applied LCCRFs on the task of blog comments quality prediction. They also compiled a new corpus comprised of articles and their subsequent user comments from the science and technology news aggregation website Slashdot. Slashdot comments are displayed in a threaded conversation-tree type layout. Users can directly reply to a given comment, and their reply

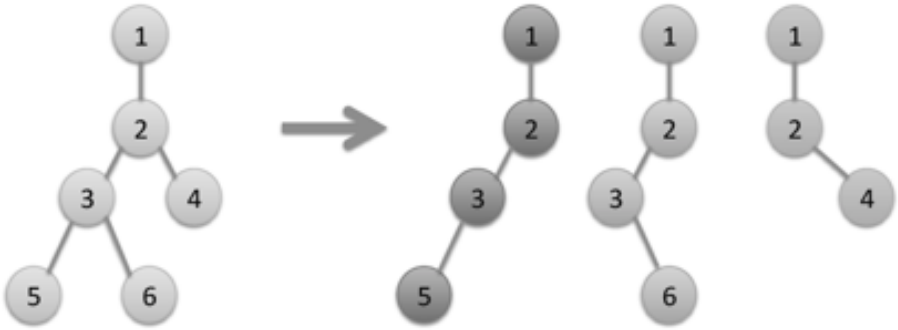

Figure 2.1: The transformation from blog conversation tree to threads

will be placed underneath that comment in a nested structure. However, LCCRFs can only deal with linear sequence data, this required the authors to transform the tree-like structure of comment conversations into sequences. To solve this problem, each conversation tree is transformed into multiple Threads, one for each leaf-comment in the tree. The blog's conversation tree has to be transformed into multiple linear sequences before the authors apply LCCRFs to perform both learning and inference . An example is shown by Figure 2.1. On the left side, we have a sample conversation initiated by the post 1, which received one comment 2, and it get two comments 3 and 4. Comment 3 receives two comments 5 and 6. These comments form a tree structure. On the right side, we can see the corresponding three linear threads decomposed from this tree, they are: (a) comment 1,2,3,5; (b) comment 1,2,3,6 and (c) comment 1,2,4.

By applying sequence tagging using LCCRFs , they were able to achieve much higher accuracy than the majority class baseline of assigning GOOD to all comments, which yielded a precision of 29.7%. The results are shown in table 2.1.

There is one problematic issue with this approach. We can see that one given conversation trees is decomposed into multiple threads in order to cast the problem in the form of sequence labelling. The result of this is that

	Not GOOD	**GOOD**
NOT GOOD	4160	467
GOOD	862	1090
Precision	0.700	
Recall	0.558	
F-Score	0.621	

Table 2.1: Blog Comments Classification Result by FitzGerald et al [5]

after classification, each non-leaf thread has been classified multiple times, equal to the number of sub-comments of that comment. These different classifications may not be the same. For example, as shown in Figure 2.2, a given comment might well have been classified as GOOD in one sequence and NOT-GOOD in another. Notice that, in Figure 2.2, blue nodes are the comments classified as GOOD comments, while red nodes are classified as NOT-GOOD comments. We can see that node 2 is classified as GOOD in the first and last thread, while classified as NOT-GOOD in the middle one. The author choose to mark a comment as GOOD in the final tree (right side of the arrow) if it was classified as GOOD at least in one of the thread. The limitation of this method is that it chooses the classification result of a non-leaf node in an heuristic fashion. More importantly, it ignores the dependency between two nodes with a common parent node in the learning phrase.

2.1.2 Bayesian Conditional Random Fields

Bayesian Conditional Random Fields (BCRFs) are proposed by [16]. BCRF is a Bayesian approach to training and inference for conditional random fields. The motivation of applying the Bayesian framework is to address over-fitting, model selection and many other aspects of the problem. Unlike Maximum likelihood (ML), Maximum a posteriori (MAP) or large-margin approaches, BCRFs are trained by by estimating the posterior distribution of the model parameters. Subsequently, the posterior distribution is averaged for BCRF inference.

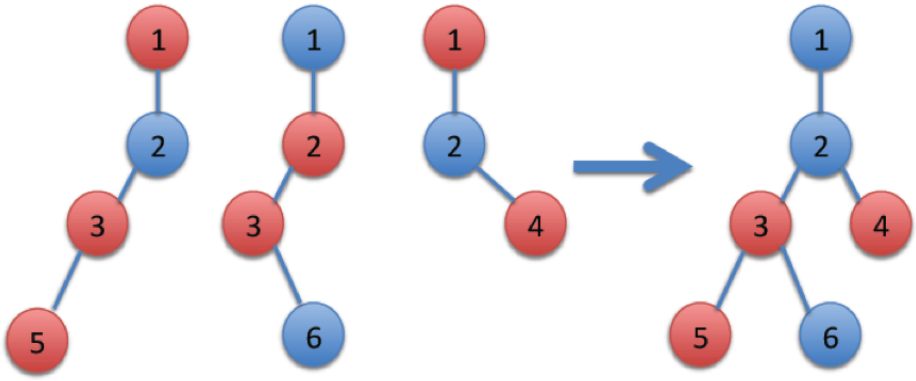

Figure 2.2: Remerging Threads

In training, BCRFs approximate the posterior distribution of the parameters using a variant of the power EP method. Also, BCRFs flatten approximation structures to increase the algorithmic stability, efficiency, and prediction accuracy. In testing, BCRFs use approximate model averaging. On synthetic data and FAQ files, they compared BCRFs with ML and MAP-trained CRFs. In almost all the experiments, BCRFs outperformed ML and MAP-trained CRFs significantly.

2.1.3 Dynamic Conditional Random Fields

Dynamic conditional random fields (DCRFs) is a generalization of linear-chain conditional random fields (CRFs) introduced by [2]. In DCRFs, each time slice contains a set of state variables and edges, which is a distributed state representation as in dynamic Bayesian networks (DBNs)and parameters are tied across slices. They performed approximate inference using several schedules for belief propagation, including tree-based reparameterization (TRP). The test on a natural-language chunking task showed that a DCRF performs better than a series of linear-chain CRFs, achieving comparable performance using only half the training data.

As described in [2], a dynamic CRF (DCRF) is a conditional distribution

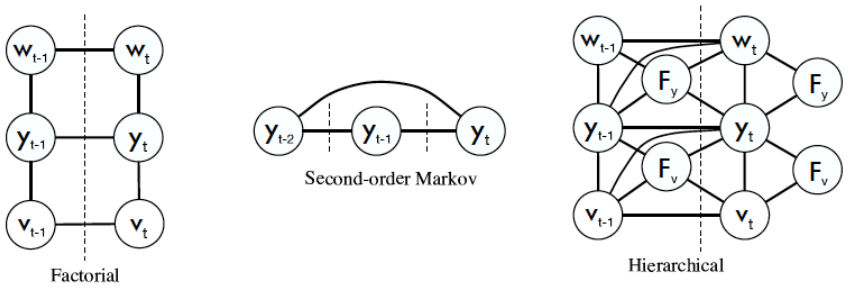

Figure 2.3: Examples of DCRFs

that factorizes according to an undirected graphical model whose structure and parameters are repeated over a sequence. As with a DBN, a DCRF can be specified by a template that gives the graphical structure, features, and weights for two time slices, which can then be unrolled given an input X. The same set of features and weights is used at each sequence position, so that the parameters are tied across the network. Several example templates are given in Figure 2.3. The dashed lines indicate the boundary between time steps. The input variables X are not shown.

The unrolling process is defined as the following: let Y = { $y_1 \cdots y_T$} be a sequence of random vectors $y_i = (y_{il} \cdots y_{I'm})$, where y_i is the state vector at time i, and y_{ij} is the value of variable j at time i. To give the likelihood equation for arbitrary DCRFs, a way to describe a clique in the unrolled graph independent of its position in the sequence is introduced. Given a time t, any variable y_{ij} in Y is denoted by two integers: its index j in the state vector y_i, and its time offset Δt = i-t. A set c = {(Δt,j)} of such pairs is called a clique index, which denotes a set of variables $y_{t,c}$ by $y_{t,c}$ = $\{y_{t+\Delta,j}|((\Delta t,j) \in c\ \}$. That is, $y_{t,c}$ is the set of variables in the unrolled version of clique index c at time t.

The formal definition of DCRFs is the following:

$$p(y|x) = \frac{1}{Z(x)} \prod_t \prod_{c \in C} exp\left(\sum_k \lambda_k f_k(y_{t,c}, x, t)\right) \tag{2.1}$$

where C is a set of clique indices, F = $\{f_k(y_{t,c}, x, t\}$ is a set of feature functions and $\Lambda = \{\lambda_k\}$ is a set of real-valued weights. Then the Then the distribution p is a dynamic conditional random field. $Z(x) = \sum_y \prod_t \prod_{c \in C} exp\left(\sum_k \lambda_k f_k(y_{t,c}, x, t)\right)$ is the normalizing factor.

In summary, dynamic CRFs are conditionally-trained undirected sequence models with repeated graphical structure and tied parameters. They combine the best of both conditional random fields and the widely successful dynamic Bayesian networks (DBNs). DCRFs address difficulties both of DBNs, by easily incorporating arbitrary overlapping input features, and of previous conditional models, by allowing more complex dependencies between labels.

2.1.4 Skip-Chain Conditional Random Fields

The Skip Chain CRF is essentially a linear-chain CRF with additional long-distance edges between similar words. These additional edges are called skip edges. The features on skip edges can incorporate information from the context of both endpoints of a sequence, so that strong evidence at one endpoint can influence the label at the other endpoint. This model was introduced in the work done by [6] and [13].

In the work done by [13], a Skip-Chain CRF is applied to an information extraction problem: building a database automatically from unstructured text. Skip-Chain CRF is used to model certain kinds of long-range dependencies between entities. In this case, one important type of dependency within information extraction occurs on repeated mentions of the same field. For example, if the same entity is mentioned more than once in a document, such as Steve Jobs, in many cases all mentions have the same label, such as Conference-Speaker. We can take advantage of this fact by favouring labelings that treat repeated words identically, and by combining features from all occurrences so that the extraction decision can be made based on global

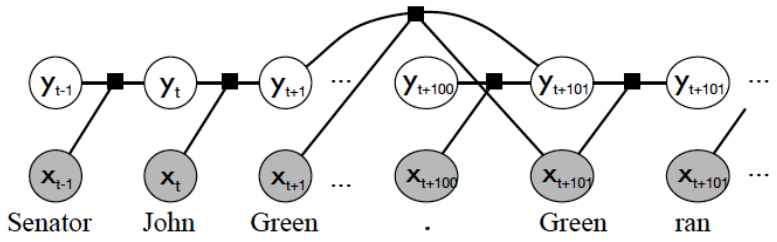

Figure 2.4: Skip Chain CRFs: Identical words are connected because they are likely to have the same label.

information. Furthermore, identifying all mentions of an entity can be useful in itself, because each mention might contain different useful information.

To perform collective labelling, dependencies between distant terms in the input should be represented. Sequence models make a Markov assumption among labels, that is, that any label y_t is independent of all previous labels given its immediate predecessors $y_{t-k} \cdots y_{t-1}$. This represents dependence only between nearby nodes. For example, between bigrams and trigramsand cannot represent the higher-order dependencies that arise when identical words occur throughout a document.

Skip-Chain CRF is used in [13] to relax this assumption. Skip-Chain CRF is a conditional model that collectively segments a document into mentions and classifies the mentions by entity type, while taking into account probabilistic dependencies between distant mentions. These dependencies are represented in a skip-chain model by augmenting a linear-chain CRF with factors that depend on the labels of distant but similar words. This is shown graphically in Figure 2.4 (Adopted from [13]).

According to [13], the skip-chain CRF is defined as a general CRF with two clique templates: one for the linear-chain portion, and one for the skip edges. For an sentence x, let I = {(u,v)} be the set of all pairs of sequence positions for which there are skip edges. Then the probability of a label sequence y given an input x is modelled as

$$p_\theta(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, x) \prod_{(u,v)\in I} \Psi_{uv}(y_u, y_v, x) \tag{2.2}$$

where Ψ_t are the factors for linear-chain edges, and Ψ_{uv} the factors over skip edges. These factors are defined as

$$\Psi_t(y_t, y_{t-1}, x) = exp\left\{\sum_k \lambda_{1k} f_{1k}(y_t, y_{t-1}, x, t)\right\} \tag{2.3}$$

$$\Psi_{uv}(y_u, y_v, x) = exp\left\{\sum_k \lambda_{2k} f_{2k}(y_u, y_v, x, u, vt)\right\} \tag{2.4}$$

where $\theta_1 = \{\lambda_{1k}\}_{k=1}^{K_1}$ are the parameters of the linear-chain template, and $\theta_2 = \{\lambda_{2k}\}_{k=1}^{K_2}$ are the parameters of the skip template. The full set of model parameters are $\theta = \{\theta_1, \theta_2\}$. Both the linear-chain features and skip-chain features are factorized into indicator functions of the outputs and observation functions.

The observation functions for the skip edges are chosen to combine the observations from each endpoint. Formally, we define the feature functions for the skip edges to factorize as:

$$f_k^{'}(y_u, y_v, x, u, v) = 1_{\{y_u = \widetilde{y}_u\}} 1_{\{y_v = \widetilde{y}_v\}} q_k^{'}(x, u, v) \tag{2.5}$$

This choice allows the observation functions $q_k^{'}(x, u, v)$ to combine information from the neighbourhood of y_u and y_v. For example, one useful feature is $q_k^{'}(x, u, v)$ if and only if $x_u = x_v =$ "Ford" and $x_{v-1} =$ "Speaker:". This can be a useful feature if the context around x_u, such as "Tom Ford is manager of control engineering. . . ," may not make clear whether or not Robert Booth is presenting a talk, but the context around x_v is clear, such as "Speaker: Tom Ford."

The experiments conducted in [13] showed that a Skip-Chain CRF model brings significant improvement over a LCCRF model in the task of identifying speakers, locations and capitalized words that occur multiple times in a seminar announcement.

2.1.5 Tree Structured Conditional Random Fields

The key advantage of a tree Structured Conditional Random Field (TCRF) model is that it can better incorporates dependencies.

It has been used in many tasks such as semantic annotation [15], semantic role labelling [3], and image labelling [1].

In the work done by Jie at al [15], they applied TCRFs to the task of annotating large volume of web pages. More specifically, they tried to label the semantic meaning of the instances of webpages based on a given ontology. Some examples of the labels are: Email Address, Registered Address, Phone Number and so on. They compared the proposed TCRFs method to LCCRFs and SVM (Support Vector Machine). The test results show that the proposed TCRF method significantly outperforms both the SVMs based method and the linear-chain CRFs based method.

For the semantic role labelling task, Trevor Cohn and Philip Blunsom [3] defined a random field over the structure of each sentences syntactic parse tree. For each node of the tree, the model must predict a semantic role label, which is interpreted as the labelling for the corresponding syntactic constituent. They showed how modelling the task as a tree labelling problem allows for the use of efficient CRF inference algorithms, while also increasing generalization performance when compared to the equivalent maximum entropy classifier.

TCRFs has been also successfully applied to several non-NLP tasks. For example, in the work done in [1], the authors presented a discriminative framework based on conditional random fields for stochastic modelling of images in a hierarchical fashion. The main advantage of the proposed framework is its ability to incorporate a rich set of interactions among the image sites. They achieved this by inducing a hierarchy of hidden variables over the given label field. The proposed tree like structure of their model eliminates the need for a huge parameter space and at the same time permits the use of exact and efficient inference procedures based on belief propagation. The authors demonstrated the generality of this approach by applying it to two important computer vision tasks, namely image labelling and object de-

tection. The model parameters are trained using the contrastive divergence algorithm. The results show that TCRF achieved the best classification accuracy in both objects detecting and image labelling tasks.

More details on TCRFs will be introduced in Chapter 4.

2.1.6 Semi-supervised CRFs

One practical difficulty in applying CRFs is that training requires obtaining true labels for potentially many sequences. This can be expensive because it is more time consuming for a human labeller to provide labels for sequence labelling than for simple classification. For this reason, it would be very useful to have techniques that can obtain good accuracy given only a small amount of labeled data.

Semi-supervised CRF is used for achieving this goal. In the work done by [8], the authors present a new semi-supervised training procedure for conditional random fields that can be used to train sequence segmentors and labellers from a combination of labeled and unlabelled training data. Their approach is based on extending the minimum entropy regularization framework to the structured prediction case, yielding a training objective that combines unlabelled conditional entropy with labeled conditional likelihood. This semi-supervised approach is applied to the problem of identifying gene and protein mentions in biological texts.

An example on this approach is described in [8]: assume we have a set of labeled examples, $D^l = \left((x^{(1)}, y^{(1)}), \cdots, (x^{(N)}, y^{(N)})\right)$, and unlabelled example, $D^u = \left(x^{(N+1)}, \cdots, x^{(M)}\right)$. A CRF model is built over sequential input and output data x and y like the following:

$$p_\theta(y|x) = \frac{1}{Z_\theta(x)} exp\left(\sum_{k=1}^{K} \theta_k f_k(x, y)\right) = \frac{1}{Z_\theta(x)} exp\left(\langle\theta, f(x, y)\rangle\right) \qquad (2.6)$$

where $\theta = (\theta_1, \cdots, \theta_K)^T$ (T stands for Transpose)

The goal is to learn a model from the combined set of labelled and unlabelled examples, $D^l \cup D^u$. The standard supervised CRF training procedure

is based upon maximizing the log conditional likelihood of the labeled examples in D^l

$$CL(\theta) = \sum_{i=1}^{N} logp_\theta(y^{(i)}|x^{(i)}) - U(\theta) \tag{2.7}$$

where $U(\theta)$ is any standard regularizer on θ, e.g. $U(\theta) = \|\theta\|^2/2$. Regularization can be used to limit over-fitting on rare features and avoid degeneracy in the case of correlated features. Equation 2.7 ignores the unlabelled examples in D^u.

To make full use of the available training data, a semi-supervised learning algorithm is introduced to exploit a form of entropy regularization on the unlabelled data. Specifically, for a semi-supervised CRF, the following objective is maximized:

$$RL(\theta) = \sum_{i=1}^{N} logp_\theta(y^{(i)}|x^{(i)}) - U(\theta) + \gamma \sum_{i=N+1}^{M} \sum_{y} p_\theta(y|x^{(i)}) logp_\theta(y|x^{(i)}) \tag{2.8}$$

where the first term is the penalized log conditional likelihood of the labeled data under the CRF defined in equation 2.7, and the second line is the negative conditional entropy of the CRF on the unlabelled data. In the equation 2.8, γ is a tradeoff parameter that controls the influence of the unlabelled data.

The experiments' results in [8]'s work show that their semi-supervised CRF approach shares all of the benefits of the standard CRF training, including the ability to exploit arbitrary features of the inputs, while obtaining improved accuracy through the use of unlabelled data. The main drawback of Semi-CRF approach is that training time is increased compared to standard CRF model. Nevertheless, the algorithm is sufficiently efficient to be trained on unlabelled data sets that yield a notable improvement in classification accuracy over standard supervised training.

2.2 Related Datasets

Datasets of actual blog conversations are needed for training (in supervised approaches) and testing. To be useful for this research, the blog comments we are using have to be rated, in the sense that good quality comments have been awarded a good rating by human users. The more data is available the better the algorithms can perform and the results of the evaluations become more accurate. The following datasets have been used in related work done by [5].

It is often very difficult to get access to real blog comments conversations as the selection of the blog comments conversations need to be very careful. For example, the comments of a blog site need to have ratings and should be possible to crawl them so that we have a sufficient enough amount of data to perform training and testing on it. For these reasons there are no publicly available corpora for our blog comment research. The only resource we had to start with is the dataset complied by FitzGerald et al [5], which is Slashdot corpus.

2.2.1 The Slashdot Dataset

In [5], the authors compiled a new corpus comprised of articles and their subsequent user comments from the science and technology news aggregation website Slashdot. This site was chosen for several reasons. Comments on Slashdot are moderated by users of the site, meaning that each comment has a scores from -1 to +5 indicating the total score of moderations assigned, with each moderator able to modify the score of a given comment by +1 or -1. The authors of [5] took this score to be a gold-standard metric for comment quality. These user moderation classes provide ready-made classifications which can be used as training and testing data for supervised approaches to identifying high-quality comments. Since the default score for an unmoderated comment is either +1 or 0, depending on whether the user is registered or unregistered respectively, they treat comments with scores of 0 or +1 as having no moderation class (class NOT-GOOD). Furthermore, to be able to identify high-quality comments, most of their experiments were

conducted with comments grouped into two categories: GOOD (all comments which fall into the four "Good Comment" classes) and NOT-GOOD (the rest).

Slashdot comments are displayed in a threaded conversation-tree type layout. Users can directly reply to a given comment, and their reply will be placed underneath that comment in a nested structure. This is in contrast to other commenting schemes which are merely linear temporal sequences, and conversational links between individual comments must be inferred from context or by the user indicating the comment or user to which they are replying (ie. @23 ...). The size of the total collection in Slashdot corpus is 425,853 comments on 4320 articles.

Chapter 3

Our Dataset

Datasets are fundamental for the evaluation, as well as for the training of statistical machine learning methods. Research done in blog comments classification has suffered from a lack of publicly available rated blog comments corpora. In this chapter we describe our solution to this problem. More specifically, we developed a new blog comments corpus that was used for classification. It consists of the Slashdot corpus developed by [5] along with five other blog websites namely DailyKos, AndroidCentral, BusinessInsider, Macrumors and TSN.

3.1 Background

The corpus construction work presented in this project expands on the basis of previous work done by [5]. Following the same requirements applied to select and compile the Slashdot corpus, we chose five additional blog sites to build our datasets. Namely, DailyKos (www.dailykos.com), AndroidCentral (www.androidcentral.com), BusinessInsider (www.businessinsider.com) Macrumors (www.macrumors.c-om) and TSN (www.tsn.ca).

These blog sites cover a variety of genra such as technology, business. politics and sports. The motivation behind this is that we want to see the performance of our TCRF model under blogs with different contents, size, conversation structure and different rating system. One common feature of these blog sites is the comments of each blog is rated by their users. In the following sections, we provide a detailed description of each blog site.

Figure 3.1: Slashdot

3.2 Slashdot

Slashdot (Figure 3.1) is a technology-related news website, the summaries of stories and links to news articles are submitted by Slashdot's own readers, and each story becomes the topic of a threaded discussion among users.

The Slashdot corpus was collected by [5]. This site was chosen for several reasons. The most important one is that this site has an ideal rating system: 1. Comments already had scores assigned by the blog users. 2. Comments on Slashdot are moderated by users of the site, meaning that each comment has a scores from -1 to +5 indicating the total score of moderations assigned, with each moderator able to modify the score of a given comment by +1 or -1. These user moderation scores provide ready-made classifications which can be used as training and testing data for supervised approaches to identifying high-quality comments. We treat comments with scores of 0 or -1 as having no moderation class (class NOT-GOOD). Others are treated as GOOD comments. The collection totalled 425,853 comments on 4320 articles.

3.3 DailyKos

DailyKos (Figure 3.2) is an American political blog that publishes news and opinions about political events. It has a comment conversation structure similar to the one in Slashdot. However, Its comment rating system is different: Each comment can be assigned points of either -1 or +1 by the users. There is no limit of the score a comment can get. These scores can also be used in supervised learning to classify a comment as GOOD or

Figure 3.2: DailyKos

NOT-GOOD. The size of our collection is 1000 posts and 60,713 comments.

3.4 AndroidCentral

AndroidCentral (Figure 3.3) is a website providing news and reports about product based on the Android system. The motivations of collecting its comments data are: (a) The comments of AndroidCentral are structured as linear sequences. This can be used to test our model's performance in different conversation structures (both tree and linear). (b) The rating system is different than other blogs: each comments is rated by users by assigning stars to it. The range of score is from 0 to 3 stars. This can be used to test our model's performance with respect to different rating methods. The size of data collected from AndroidCentral is 1000 posts and 8,277 comments.

Figure 3.3: AndroidCentral

Figure 3.4: BusinessInsider

3.5 BusinessInsider

BusinessInsider (Figure 3.4) is a U.S. business/entertainment news website. The site provides and analyzes business news and acts as an aggregator of top news stories from around the web. Its business content along with rated comments (in the form of Thumbs up(+1) and Thumbs down (-1)) provides us with an additional, different scenario to train and test our model. The size of the data we get from BusinessInsider is 1000 Post with 10,856 Comments.

3.6 Macrumors

Macrumors (Figure 3.5) is a website that aggregates Mac and Apple related news, rumours, and reports. Macrumors is also home to one of the largest Mac-focused forum sites, with over 400,000 members and over 10,000,000 forum posts as of May 2010. We collected 1000 posts and 14,120 comments from this site.

3.7 TSN

TSN (The Sports Network, in Figure 3.6) is a website presenting highlights and score updates about sports such as NHL Hockey, NFL and NBA etc. The reason for getting comments from TSN is the same as for the other blogs in our corpus: rated comments. Furthermore, for diversity, it is nice

Figure 3.5: Macrumors

Figure 3.6: Macrumors

to cover sports topic in our dataset. At last, the easiness of getting the comments is another reason: all the comments of a given post are included in a JSON (JavaScript Object Notation) object, which makes it very easy to process and store in a universal format. The size of the data collected is 1000 posts with 54,235 comments.

3.8 Data Processing Framework

3.8.1 Pre-processing

A web-crawler was built to collect articles and comments from these blog sites. To handle the issue that different websites have their own way of storing and displaying the content of their comments, a conversion step was developed to store all these data in a common representation. More

cid	comment_title	comment_author	date	pid	text	url	score
41796337	As far as I kno...	commonmass	Wed Jun 01, 2...	981080	As far as I kno...	http://www.d...	8
41796377	You cut me to...	blue aardvark	Wed Jun 01, 2...	981080	You cut me to...	http://www.d...	10
41796396	Thanks, jotter!	Hedwig	Wed Jun 01, 2...	981080	Thanks, jotter...	http://www.d...	11
41796409	Too funny!	smileycreek	Wed Jun 01, 2...	981080	Too funny! Es...	http://www.d...	13
41796450	I feel terrible f...	commonmass	Wed Jun 01, 2...	981080	I feel terrible f...	http://www.d...	11
41796457	Actually, one...	navajo	Wed Jun 01, 2...	981080	Actually, one...	http://www.d...	11
41796480	Hey, when do...	navajo	Wed Jun 01, 2...	981080	Hey, when do...	http://www.d...	9
41796490	Thanks jotter!	boran2	Wed Jun 01, 2...	981080	Thanks jotter!...	http://www.d...	11
41796516	Oh, you can n...	navajo	Wed Jun 01, 2...	981080	Oh, you can n...	http://www.d...	8
41796527	I put at least a...	blue aardvark	Wed Jun 01, 2...	981080	I put at least a...	http://www.d...	9
41796630	smileycreek	navajo	Wed Jun 01, 2...	981080	smileycreek M...	http://www.d...	9
41796650	boran2	navajo	Wed Jun 01, 2...	981080	boran2 meet...	http://www.d...	11
41796691	Yeah, no pres...	navajo	Wed Jun 01, 2...	981080	Yeah, no pres...	http://www.d...	8
41796713	Thanks Jotter	blueoregon	Wed Jun 01, 2...	981080	Thanks Jotter	http://www.d...	8
41796879	I see no basis...	blue aardvark	Wed Jun 01, 2...	981080	I see no basis...	http://www.d...	6
41796918	This is clearly...	navajo	Wed Jun 01, 2...	981080	This is clearly...	http://www.d...	5
41796988	good to meet...	smileycreek	Wed Jun 01, 2...	981080	good to meet...	http://www.d...	7

Figure 3.7: Sample table

specifically, we process the data crawled and store them into a relational database tables with the same scheme. The scheme of the table consists of comment's id [cid], comment title, comment author, date, post id [pid], content of the comment [text], the web link of the comment [url], rating of the comment [score], parent comment of given comment [parent] and so on. An example of such database table can be seen in Figure 3.7.

3.8.2 Transformation

The next step of our data processing framework is to transform the data stored in database into XML (Extensible Markup Language) like format. The motivation of this is to convert the data into the linear sequence and tree structures needed by the tool we use to train and test our LCCRFs and TCRFs models. A sample formatted file is shown in Figure 3.8.

3.8.3 Feature Extraction

The last step in processing the data in to extract the feature values needed for training and testing in our CRFs models. Since we are modifying the GRMM (GRaphical Models in Mallet) toolkit to perform our training and testing tasks, the training and testing files feeder to GRMM should look like the format in Figure 3.9.

```
<comment>
        <cid> comment_id</cid>
        <pid> post_id of this comment </pid>
        <title> comment_title </title>
        <url> comment_url </url>
        <date> comment_date </date>
        <author> comment_author </author>
        <score> comment_score </score>
        <text> post_body </text>
</comment>
```

Figure 3.8: XML format data

```
LABEL11 LABEL12 ... LABEL1k ---- feature11 feature12 ...
LABEL21 LABEL22 ... LABEL2k ---- feature21 feature22 ...
LABEL32 LABEL32 ... LABEL3k ---- feature31 feature32 ...
....
```

Figure 3.9: Feature Format

3.8.4 Dataset Summary

In Table 3.1, we present a summary of key properties and basic statistics of the data we have collected from these blog sites. These blog sites cover a variety of genres such as technology, business. politics and sports; they also have different conversation structure (sequential or tree) and rely on different rating methods. This kind of diversity provides us with a comprehensive research target. We will be able to test the generality of our approach on very different blogs; and verify how the performance of our proposal varies with respect to the key properties of these blogs. Notice that since our dataset covers several different domains, in future work, we will be able to test domain adaptation techniques. Furthermore, these real life data makes our research more applicable and interesting.

Name	Structure	Rating Method	Genre	Articles	Comments
Slashdot	Tree	-1 to 5	Technology	4320	425853
DailyKos	Tree	positive - negative	Politics	1000	60713
AndroidCentral	Sequential	0 - 3 Stars	Technology	1000	8277
BusinessInsider	Tree	positive - negative	Business News	1000	10856
Macrumors	Sequential	positive - negative	Technology	1000	14120
TSN	Tree	positive - negative	Sports	1000	54235

Table 3.1: Dataset description

Chapter 4

Our Approach: Tree structured Conditional Random Fields

This chapter presents our approach to classifying blog comments using Tree-structured Conditional Random Fields. As described in Chapter 2 (Related Work), there are mainly two ways to apply CRFs to classification problem: LCCRFs and TCRFs. In this work, we focus on expanding the work done by [5]. Instead of applying LCCRFs as the authors of [5] did, we try to explore the results of blog comments classification using TCRFs to better handle the dependencies across the hierarchically structured comments in a blog conversation.

An outline of the chapter is as follows. In Section 4.1 we describe the LCCRF model used in previous work done by [5]. Section 4.2 introduces the TCRFs model we proposed in our work.

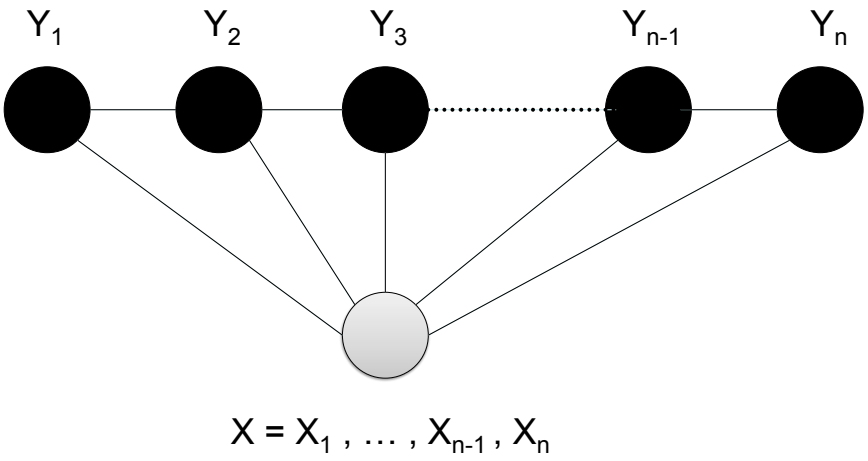

Figure 4.1: Linear Chain CRFs

4.1 Linear Chain Conditional Random Fields

Conditional Random Fields introduced by [7] are discriminative probabilistic models which have gained much popularity in Natural Language Processing and Bio-informatics applications. Like Markov Random Fields, CRFs can be defined on an arbitrary undirected graph. However, for sequence-labelling tasks, linear-chain models are often used. These Linear-Chain CRFs have been shown to provide superior performance on many of the same tasks traditionally handled by Hidden Markov Models, their generative counterpart.

A CRF is a random field globally conditioned on the observation X. Linear-Chain CRFs were first introduced by Lafferty et al [7]. The graphical structure of linear-chain CRFs is shown in Figure 4.1.

Let X and Y be random vectors, $\theta = \{\theta_k\} \in R^k$ be a parameter vector, and $\left\{f_k(y, y^{'}, X_t)\right\}$be a set of real-values feature functions. Then a linear-chain conditional random field is a distribution $P(y|x)$ that takes the form:

$$p(y|x) = \frac{1}{Z(X)} \prod_{t=1}^{T} exp\left\{\sum_{k=1}^{L} \theta_k f_k(y_t, y_{t-1}, X_t)\right\} \tag{4.1}$$

where Z(x) is an instance-specific normalization function:

$$Z(X) = \sum_{y} \prod_{t=1}^{T} exp \left\{ \sum_{k=1}^{L} \theta_k f_k(y_t, y_{t-1}, X_t) \right\} \tag{4.2}$$

The parameter values, θ_k, can be estimated from training data by several methods. In [5], they applied Linear-Chain CRFs to the problem of detecting high-quality blog comments in sequences of comments. One benefit of using linear chain CRFs over more traditional linear classification algorithms is that the sequence of labels is considered. However, this approach has two main limitations: (1) it ignores many hierarchical dependencies between the output tags/labels and (2) the common internal nodes fall in multiple paths, which cause them to be classified multiple time, and possible inconsistent classifications have to be combined heuristically (see Section 2.1.1 for details). These limitation inspire the use of TCRFs model which will be introduced in the next section.

4.2 Tree Structured Conditional Random Fields

A Tree-structured Conditional Random Field (TCRF) model is a particular variation of the CRFs framework. Since the structure of blog conversations is often a tree, TCRFs are applied to take advantage of the dependencies across hierarchically laid-out information in the blog conversation. The structure of the tree CRF will mirror the structure of the conversation as shown in Figure 4.3. On the left side, we have a sample conversation initiated by the post x_0, which received two comments x_1 and x_2 etc. On the right side, you can see the corresponding TCRF, in which all the posts are represented as the observed variables x_i and their labels as the corresponding hidden variables y_i. The TCRF models the parent-child dependencies for all the comments with their children (e.g., y_0-y_1 and y_0-y_2).

The conditional probability for the labels given the observation in a CRF is defined as:

$$p(y|x) = \frac{1}{Z(x)} exp(\sum_{c \in C} \sum_{k} \lambda_k f_k(c, y_c, x)) \tag{4.3}$$

where C is the set of cliques, λ_k are the feature weights and $f_k(c, y_c, x)$ are

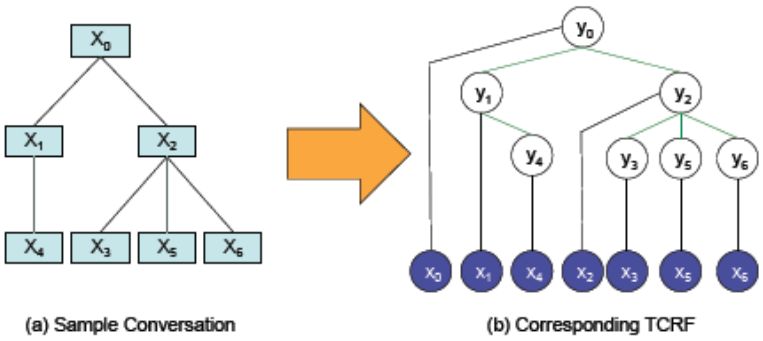

Figure 4.2: Sample Conversation and corresponding TCRF

the feature function for the clique. Z(x) is the normalization function. The probability distribution can be rewritten to separate the node cliques and from edge cliques (between different nodes) as:

$$p(y|x) = \frac{1}{Z(x)} exp(\sum_{u \in C_1} \sum_{k} \lambda_k g_k(u, y_u, x)) + exp(\sum_{v \in C_2} \sum_{k} \lambda_k h_j(u, v, y_u, y_v, x)) \tag{4.4}$$

where C_1 is the set of node cliques and C_2 is the set of edge cliques. In the TCRF model, edge cliques are edges between parent and child nodes. We used the dependency structure in the blog conversations to model the TCRF.

4.3 GRMM Modification

GRMM (GRaphical Models in Mallet) is a general machine learning software package for implementing probabilistic graphical models providing various fitting/learning algorithms like Limited Memory BFGS (LBFGS) and inference algorithms like Belief Propagation, Tree-based re-parameterization (TRP). We slightly modified the software to model arbitrary tree structures and suit our experimental needs.

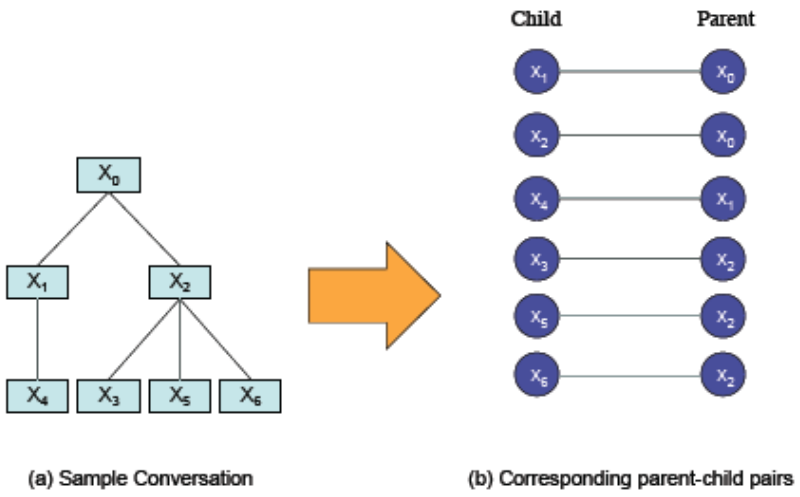

Figure 4.3: Parent-Child pairs from a given conversation tree

In order to build Tree structured CRF in GRMM, we create our own FactorTemplate class that: (1) define how the tree structure should be created from our FeatureVectorSequence (2) define the parameter tying (all factors created by the same Template have tied parameters)

Another change is that we need to supply a structure file containing the parent-child pairs information of a given blog conversation trees to the modified Arbitrary CRF (ACRF) classes. An example of such process is shown in Figure 4.3. Where on the left side, we have a sample conversation initiated by the post x_0, which received two comments x_1 and x_2 etc, just the same as we have in Figure 4.2. On the right side, you can see the corresponding parent-child pairs from the given conversation tree, in which all the child nodes in the conversation tree are mapped to their parent. For example, x_1 is mapped to x_0 and x_3 is mapped x_2.

4.4 Feature Selection

Each comment in a given blog conversation tree was represented as a series of features. In our approach, we consider the same features which are used in previous work done in [5]: unigrams, lexical similarity, and conversational features. These are described below:

Unigram Features

Unigrams are simply the words present in a given comment. Each unigram is a binary feature, indicating either the presence or absence of the given word in the comment in question. The intuition behind using these features is that good comments might be more likely to include certain words, or that replies to good comments might be more likely to include certain words (such as words which indicate agreement).

Similarity Features

Three features were used which capture the overlap of words between two comments: TF-IDF, LSA and Lexical Cohesion. For each comment, each of these three scores was calculated for both the preceding and the following comment (0 if there was no comment before or after), giving a total of six similarity features. All of these metrics are weighting schemes applied to each word in the document. These weights then form a vector representing the comment, indexed by terms in the corpus. A given term is zero if the word does not appear in the document, or equal to the weight for the word if it does appear. The distance between two comments is then the cosine-similarity between the two vectors. The intuition behind the use of these features is that good comments should be ones that tend to use similar language to those around it. This should indicate comments which are relevant and on-topic. In our work, the "document" referred to in each metric's description is the set of all comments for the article in whose conversation the comment appears.

TF-IDF (term frequency-inverse document frequency) is a weighting

scheme commonly used in computational linguistics applications. The overall TF-IDF weight is the product of the Term Frequency and Inverse Document Frequency scores. The Term Frequency is simply the number of occurrences of the term in the document, divided by the total number of terms in the document:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_{k \in j} n_{k,j}} \tag{4.5}$$

The Inverse Document Frequency (IDF) is the inverse frequency of the appearance of the term in all documents in the corpus:

$$idf_i = \frac{|D|}{|\{d : t_i \in d\}|} \tag{4.6}$$

The intuition behind this metric is that the most important words to determine similarity between two documents are those which are frequent in the documents, but infrequent in the language as a whole. This means that very common words which are likely to appear in all comments (like and and the) will not contribute much to the similarity score.

LSA (Latent Semantic Analysis) is a matrix-based approach to document similarity. First we form a matrix X where (i, j) is the occurrence of term i in document j. Now we decompose X as $X = U\Sigma V^T$, where U and V are orthogonal matrices and Σ is a diagonal matrix. The columns of V^T are now $d_j^{'}$, and the similarity between two documents can be calculated as the cosine-distance between $d_i^{'}$ and $d_j^{'}$.

LCSeg (Lexical Chain Segmenter) works by first forming chains of word repetitions. These chains are then ranked on two criteria: the length of the chain (number of repetitions of the term) and the compactness of the chain (number of words the chain spans in the sequence). The chain-weight for each term then form the document vectors, which again are compared with cosine-similarity.

Conversational Features

The conversational features capture information about where the comment is situated in the conversation as a whole. The list is as follows:

- **NumReplies**
  The number of replies to this comment. This includes all comments which are children of this comment in the conversation tree.
- **WordLength**
  The length of this comment in words.
- **SentenceLength**
  The length of this comment in sentences.
- **AvgReplyWordLength**
  The average length of replies to this comment in words length.
- **AvgReplySentLength**
  The average length of replies to this comment in sentences length.
- **TotalReplyWordLenth**
  The total length of all replies to this comment in words length.
- **TotalReplySentLength**
  The total length of all replies to this comment in sentences length.

In addition, we also modify and apply some of the features introduced in the work done by [11] to our experiments. They are listed as following:

$$idf_i = \frac{|D|}{|\{d : t_i \in d\}|} \tag{4.7}$$

- **TPOS1**
  The time from the post time from original post to the time of this comment.
- **TPOS2**
  The time from the post time of this comment to the time of last comment in this conversation tree.

- **PPAU**
  The time between the post time of this comment and its parent's time.
- **BEGAUTH**
  The comment is the first comment to the given article or not.
- **DOM**
  A measure to see how dominant the current participant is in terms of words in the conversation.

Chapter 5

Experiments and Results

5.1 Experiments

5.1.1 Experiments Setup

Dataset

Our dataset is described in details in Chapter 3. The size of the entire corpus is 8,400 blog conversations with 304,500 comments.

Software Package

We used GRMM (Graphical Models in Mallet) toolkit [14] to implement our TCRF model. GRMM is a general machine learning software package for implementing probabilistic graphical models providing various fitting/learning algorithms like Limited Memory BFGS (LBFGS) and inference algorithms like Belief Propagation, Tree-based re-parameterization (TRP). We slightly modified the software to model arbitrary tree structures and suit our experimental needs.

Classifiers

In our experiment, we used three classifiers for blog comments classification. The first one is a majority-class baseline classifier that assign to each testing comment the most frequent class in the training test. For example, if the training corpus contains 60% GOOD comments and 40% NON-GOOD comments, the classifier will always classify a testing comment as GOOD comment. The second is a CRF classifier using linear chain as its graphical

model [5]. The third is our new approach, a CRF classifier using dependency tree as its graphical model.

5.2 Training and Testing

For the training and testing, the popular Natural Language Machine Learning toolkit MALLET[7] and its package GRMM[8] are used. MALLET is used for the LCCRF model, while GRMM's ACRF class is modified to create the TCRF model.

10-fold cross-validation is used for our training and testing of each blog's corpus under both LCCRFs and TCRFs model. In 10-fold cross-validation, the original sample is randomly partitioned into 10 subsamples. Of the 10 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 9 subsamples are used as training data. The cross-validation process is then repeated 10 times (the folds), with each of the 10 subsamples used exactly once as the validation data. The 10 results from the folds then can be averaged (or otherwise combined) to produce a single estimation. The advantage of this method over repeated random subsampling is that all observations are used for both training and validation, and each observation is used for validation exactly once.

To compare the performance difference under LCCRF model and TCRF model, the blogs in our corpus with tree structured conversation layout (Slashdot, DailyKos, BusinessInsider and TSN) are trained and tested under both models.

5.3 Results

5.3.1 Classification

Our goal in this experiments is to be able to detect comments which had been identified as GOOD by blog users. This suggests that the problem

[7] http://mallet.cs.umass.edu/index.php
[8] http://mallet.cs.umass.edu/grmm/index.php

should be formulated as a binary classification problem, where we need to classify good comments.

Our experiment is to train the LCCRF and TCRF using data where the full set of moderation labels had been grouped into GOOD comments and NON-GOOD. These two Conditional Random Field classifiers were trained on the full set of features presented in Section 5.1.3. The result of our experiment is presented in Table 5.1.

LCCRF	**Precision**	**Recall**	**F-Score**
BusinessInsider	62.78%	**62.72%**	62.75%
DailyKos	69.40%	63.60%	66.40%
Slashdot	70.45%	**69.42%**	69.93%
TSN	64.31%	59.22%	61.66 %
TCRF	**Precision**	**Recall**	**F-Score**
BusinessInsider	**65.86%**	61.58%	**63.65%**
DailyKos	**75.20%**	**63.82%**	**72.30%**
Slashdot	**73.48%**	68.93%	**71.13%**
TSN	**72.33%**	**66.31%**	**68.81%**
Baseline	**Precision**	**Recall**	**F-Score**
BusinessInsider	54.22%	53.47%	53.84%
DailyKos	59.23%	56.31%	57.73%
Slashdot	60.23%	63.12%	61.64%
TSN	56.75%	58.12%	57.43%

Table 5.1: Comparison of LCCRF, TCRF and Baseline

From Table 5.1 we can see that TCRF performs better than LCCRF for all four tree-structured blog corpus in terms of F-Score. The improvement is especially noticeable for DailyKos and TSN. However, for Slashdot and BusinessInsider, the improvement is very limited because of a slight decrease in Recall performance. The analysis of this result is presented in next section.

5.3.2 Result Analysis

Statistical Analysis

For the statistical analysis, we performed the Paired t-Test to see if the improvement we obtained using TCRF compared to LCCRF model was statistically significant.

We calculated the results for each conversation tree in the blog using TCRF and LCCRF. For example, the Dailykos has 1000 conversation trees (articles). We calculated the F-Score for each conversation tree using both TCRF and LCCRF so we got 1000 scores under each model. Then we performed the Paired t-Test with the following hypotheses: (1) H_0: $Mean_{tcrf} <= Mean_{lccrf}$ (2)H_a: $Mean_{tcrf} > Mean_{lccrf}$. The chosen significance level is 5% for all tests.

In our tests for all four blogs, TCRF model showed statistically significant improvement over LCCRF approach. For the comparisons, one-tailed t-tests were always performed and the null hypothesis was rejected with 95% confidence.

Tree Complexity vs Performance

In this section we are focusing on analyzing the correlation between the performance of our TCRF model and the complexity of the tree structure of each blog corpus.

The complexity of a tree structure is decided by its *Branching Factor* and *Height*.

The Branching Factor of a tree is the number of children at each node in a tree. In our case, since the number of children at each node is not uniform for a given blog conversation tree, an average branching factor is calculated for each tree.

The Height of a tree is the height of this tree's root node. The height of the root node is the length of the longest path to a leaf from that node. In other words, *the Height* of a tree is the "number of levels" in this tree.

In order to represent *The Branching Factor* and *The Height* of a tree

using one value so we can see the correlation between the complexity of tree and the performance of our TCRF model more easily, we define the complexity of a tree are computed as following:

$$Complexity(Tree) = Branching_Factor(Tree) * Height(Tree) \quad (5.1)$$

The tree complexity information of all four blogs with tree-structured conversation is shown in Table 5.2.

Blogs	**Branching Factor**	**Height**	**Complexity**
BusinessInsider	3.67	1.86	6.83
DailyKos	4.33	2.79	12.08
Slashdot	6.47	2.47	15.98
TSN	3.92	2.92	11.45

Table 5.2: Tree Complexity of Blogs

From Table 5.2, we can observe that Slashdot has the highest complexity score at 15.98 while BusinessInsider's 6.83 is the lowest. DailyKos and TSN are very close in terms of complexity score.

Blogs	**Complexity**	**Performance Improvement**
BusinessInsider	6.83	1.43%
DailyKos	12.08	8.89%
Slashdot	15.98	1.72%
TSN	11.45	11.60%

Table 5.3: Tree Complexity vs Performance Improvement in percentage

Table 5.3 and Figure 5.1 together show the relationship between blogs' tree complexity and the improvement they got using TCRFs over LCCRFs. The observation is that if the tree complexity of a given blog is low (BusinessInsider), the improvement we got using the TCRF model comparing to LCCRF is limited compared to the blogs with higher tree complexity (DailyKos and TSN). However, if the tree complexity is over some limit (Slashdot), then the performance of TCRF is decreased due to over-complex tree

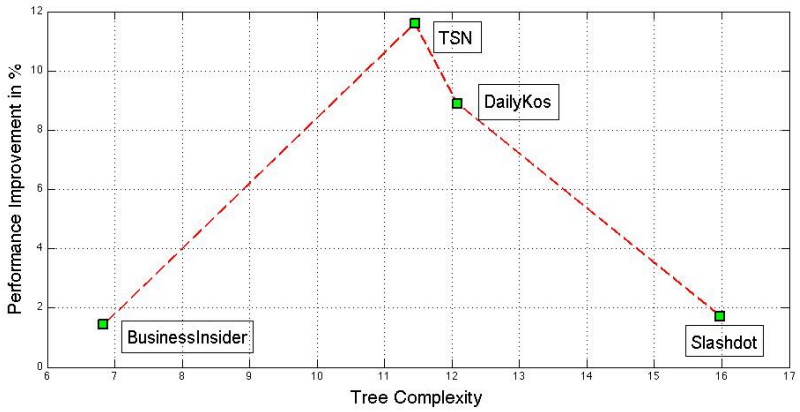

Figure 5.1: Tree Complexity vs Performance Improvement in percentage

structure of the blog conversation.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis we described how to classify blog comments using TCRF model. Our classification approach is an extension of the previous work done by [5], which applied LCCRF approach to training the classifier. Since we used machine learning, we also required a dataset. To our knowledge there were no publicly available datasets for blog comments. We therefore collected blog comments conversation from a variety range of different blogs. We compared Tree-structured Conditional Random Fields and Linear-chain Conditional Random Fields approaches and found TCRF model has statistically significant improvement over LCCRF approach.

As a summary we outline the main achievements of this thesis here:

- We compiled a new corpus comprised of the Slashdot dataset collected by FitzGerald et al [5], and conversations from five other blog websites namely, DailyKos, AndroidCentral, BusinessInsider, Macrumors and TSN. These blog sites cover a variety of genres such as technology, business, politics and sports; they also have different conversation structure (sequential or tree). For example, Slashdot conversations have a tree-like structure; users can directly reply to a given comment, and their reply will be placed underneath that comment in a nested structure. On the contrary, comments in conversations from AndroidCentral always form a single linear thread. We were able to test the generality of our approach on very different blogs; and verify how the

performance of our proposal varies with respect to the key properties of these blogs. Notice that since our dataset covers several different domains, we will be able to test domain adaptation techniques in the future work.

- In the previous work done in the field of comments classification using CRFs, the authors use a linear-chain CRFs (LCCRFs) to detect the informative blog comments through the exploration of conversational and topical features. However, their approach has two main limitations: (1) it ignores hierarchical dependencies between the output tags/labels and (2) the common internal nodes fall in multiple paths, which cause them to be classified multiple time, and possible inconsistent classifications have to be combined heuristically. Our proposed approach applies a TCRF model to better handle the dependencies across the hierarchically structured comments in a blog conversation.

- For building Tree structured CRFs in GRMM, we created our own FactorTemplate class and supplied a a structure file containing the parent-child pairs information of a given blog conversation trees to the modified Arbitrary CRF classes.

- We compared our proposed TCRFs model with the LCCRFs approach previously applied in [5] and we found that our TCRFs approach works better than LCCRFs model. TCRFs outperformed LCCRFs due to the inherent tree structure which captures the semantic and syntactic dependencies better.

- In this thesis we also wanted to see if there was any correlation between the performance improvement of our TCRF model and the tree's complexity of a given blog. We observed that higher complexity could bring performance gain for the TCRF model. However, over-complicated tree structure could hurt the performance of our model.

6.2 Future Work

Classification of blog comments is a relatively new field. There are many areas that have yet to be explored. Research can also be done on similar but newer forms of communication such as Twitter or Facebook discussions. The following are some topics that would be interesting to explore:

- Extending this research into other forms of communication would be interesting. Although online chats like Twitter or Facebook chats are less structured than blog conversations, it would be interesting to see if the same techniques can be applied in this field.

- It would be preferable to have finer-grained classification than just GOOD and NON-GOOD. For this task, some of the finer grained classes provided in the Slashdot corpus would be a good starting point. Models other than CRFs may be needed since LCCRFs didn't perform well on this task in the work done by [5].

- Since our dataset covers several different domains, it would be interesting to test domain adaptation techniques using our CRFs models.

- Exploring more ways to refine our ability to classify comments. For example, some blogs have the author information available for each comment. We can take advantage of this kind of information in our model to see if there is better outcome.

- Incorporating comments classification into an abstractive summarization system. It would be useful to have such a summarization system to summarize blog comments for users.

Bibliography

[1] Pranjal Awasthi, Aakanksha Gagrani, and Balaraman Ravindran. Image modelling using tree structured conditional random fields. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2060–2065, 2007.

[2] Andrew McCallum Charles Sutton and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labelling and segmenting sequence data. *Journal of Machine Learning Research*, 2007.

[3] Trevor Cohn and Philip Blunsom. Semantic role labelling with tree conditional random fields. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 169–172, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[4] Matthieu Constant and Isabelle Tellier. Evaluating the impact of external lexical resources into a crf-based multiword segmenter and part-of-speech tagger. In *International Conference on Language Resources and Evaluation (LREC'12)*, 2012.

[5] Nicholas FitzGerald, Giuseppe Carenini, Gabriel Murray, and Shafiq R. Joty. Exploiting conversational features to detect high-quality blog comments. *Canadian Conference on AI*, pages 122–127, 2011.

[6] Michel Galley. A skip-chain conditional random field for ranking meeting utterances by importance. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 364–372, 2006.

[7] A. McCallum J. Lafferty and F. Pereira. Conditional random fields: probabilistic models for segmenting and labelling sequence data. *International Conference on Machine Learning*, 2001.

[8] Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, 2006.

[9] Andrew McCallum. Efficiently inducing features of conditional random fields. *Conference on Uncertainty in Artificial Intelligence*, 2003.

[10] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. *Seventh Conference on Natural Language Learning (CoNLL)*, 2003.

[11] Gabriel Murray and Giuseppe Carenini. Summarizing spoken and written conversations. In *In Proc. of EMNLP 2008*, 2008.

[12] Fuchun Peng, Fangfang Feng, and Andrew McCallum. Chinese segmentation and new word detection using conditional random fields. In *COLING*, 2004.

[13] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[14] Charles Sutton. Grmm: Graphical models in mallet, 2006.

[15] Jie Tang, Mingcai Hong, Juanzi Li, and Bangyong Liang. Tree-structured conditional random fields for semantic annotation. In *In Proceedings of 5th International Conference of Semantic Web (ISWC2006*, pages 640–653, 2006.

[16] Martin Szummer Yuan Qi and Thomas P. Minka. Bayesian conditional random fields. In *AISTATS 2005*, 2005.