

# Exploiting Conversational Features to Detect High-Quality Blog Comments

No Author Given

No Institute Given

**Abstract.** In this work, we present a method for classifying the quality of blog comments using Linear-Chain Conditional Random Fields. This approach is found to yield high accuracy on binary classification of high-quality comments, with conversational features contributing strongly to the accuracy. We also present a new corpus of blog conversations in conversational form, complete with user-generated quality moderation labels from the science and technology news blog Slashdot.

## 1 Introduction

As the amount of content available on the Internet continues to increase exponentially, the need for tools which can analyze and summarize large amounts of data has become increasingly pronounced. This is particularly the case for so-called grey-intelligence, information which exists not in well-structured databases, but in unstructured formats such as text. One method of aggregating information from collections of text is *abstractive summarization*, where important content from the source texts is extracted and expressed in a concise summary. Recent work in [8] and [9] has focused on summarization of conversations, in which several participants trade contributions. One key problem in this task is distinguishing high-quality contributions from low-quality ones, so that the summary can focus on information extracted from the best, and therefore most influential comments.

In this paper, we present our work on detecting high-quality comments in blogs using Conditional Random Fields, which is arguably a key step towards abstractive blog summarization. In Section 2, we give background on Automatic Summarization and Conditional Random Fields. In Sect. 3, we present a new corpus of blog conversations collected for this experiment, and describe the approach taken to identifying high-quality comments. In Sect. 4, we describe a series of experiments which show the effectiveness of our approach for identifying high-quality comments, and explore more fine-grained analysis. Sect. 5 presents our conclusions and considerations for future work.

## 2 Background

### 2.1 Automatic Summarization

Traditionally, most work on automatic summarization has focused on extractive methods, where representative sentences are chosen from the input corpus ([4]).

This approach is conceptually simple, and has the major advantage of being easily cast as a binary classification problem, which allows for the easy adaptation of popular machine-learning techniques. In contrast, recent work (eg. [9], [2]) has taken an abstractive approach, where information is first extracted from the input corpus, and this information is then expressed through novel sentences created with Natural Language Generation techniques. This approach, though more difficult, has several advantages over extractive methods. Since content from the corpus is freed from the specific sentence structure in which it appeared in the input documents, the system has greater flexibility in terms of the order and grouping of expressed information. This results in summaries which are judged to be more coherent and concise than those created by extractive systems.

[9] describes a system for the automatic generation of abstractive summaries of meeting conversation transcripts. The approach is to use a series of classifiers to identify different types of messages in the transcripts—for example, utterances expressing a decision being made, or a positive opinion being expressed. The summarizer then selects a set of messages which maximize a function encompassing information about the sentences in which messages appear.

In this work, we focus on the problem of identifying the high-quality contributions from a conversation of blog comments. In future work, this will be combined with classification on other axes—for instance that of the message’s rhetorical role (ie. Question, Response, Criticism etc.)—to provide the messages for an abstractive summarization system. Blogs are a particularly challenging domain for automatic summarization, because the conversations often span a wide range of topics, and the language used is often ungrammatical and contains frequent use of verbal irony and humour.

## 2.2 Conditional Random Fields

Conditional Random Fields ([6]) are a discriminative probabilistic model 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 like the experiment in this study, 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 ([10]), their generative counterpart. The specification of a Linear-Chain CRF is as follows (from [12]):

Let  $X$  and  $Y$  be random vectors,  $\theta = \{\theta_k\} \in \mathfrak{R}^k$  be a parameter vector, and  $\{f_k(y, y', \mathbf{x}_t)\}_{k=1}^K$  be a set of real-valued feature functions. Then a linear-chain conditional random field is a distribution  $p(\mathbf{y} | \mathbf{x})$  that takes the form:

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

where  $Z(\mathbf{x})$  is an instance-specific normalization function:

$$Z(x) = \sum_y \prod_{t=1}^T \exp\left\{ \sum_{k=1}^L \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\}$$

The parameter values,  $\theta_k$ , can be estimated from training data by several methods.

We apply 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. Several works have shown the effectiveness of CRFs on similar Natural Language Processing tasks which involve sequential dependencies. [11] uses Linear-Chain CRFs to classify summary sentences to create extractive summaries of news articles, showing their effectiveness on this task. [5] test CRFs against two other classifiers (Support Vector Machines and Naive-Bayes) on the task of classifying dialogue acts in live-chat conversations. They also show the usefulness of structural features, which are similar to our conversational features (see Sect. 3.3). [1] and [3] use CRFs to categorize sentences based on their rhetorical roles in scientific abstracts. Again, these experiments show the dominance of CRFs over non-sequential classifiers.

In contrast to this previous work, we seek to classify the perceived quality of entire comments within a conversation. Furthermore, we are working within the domain of blog comments, which are asynchronous, tree-structured conversations with multiple participants.

## 3 Experiment

### 3.1 The Slashdot Corpus

We compiled a new corpus comprised of articles and their subsequent user comments from the science and technology news aggregation website Slashdot<sup>1</sup>. 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. We take this score to be a gold-standard metric for comment quality. Furthermore, each moderation assigns a classification to the comment: for good comments, the classes are: *Interesting*, *Insightful*, *Informative* and *Funny*. For bad comments, the classes are: *Flamebait*, *Troll*, *Off-Topic* and *Redundant*. 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, we treat comments with scores of 0 or +1 as having no moderation class (class *NONE*). Furthermore, since the goal of

<sup>1</sup> <http://slashdot.org>

this work was to identify high-quality comments, most of our experiments were conducted with comments grouped into two categories: *GOOD* (all comments which fall into the four "Good Comment" classes) and *NOT* (the rest).

Secondly, 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 conversational structure of Slashdot comments allows us to use Conversational Features in our classification approach (see Sect. 3.3).

A final more pragmatic consideration was that the HTML design of the website made it easier to collect comments using a traditional web-crawler than other candidate sites which serve the majority of their data dynamically.

A web-crawler was built to collect articles and comments from Slashdot. Due to the inherently messy nature of trying to pull text from a website, some comments were not successfully crawled, which meant that some comments in the corpus referred to parent comments which had not been collected. In order to prevent this, comments whose parents were missing were excluded from the corpus. After this cleanup, the collection totalled 425,853 comments on 4320 articles.

### 3.2 Transformation into Sequences

As mentioned above, Slashdot commenters can reply directly to other comments, forming a tree-like structure of comment conversations, where each comment can have multiple children. Each article will have several such conversation trees, comprised of root comments, and the tree of subsequent replies. This creates a problem for our use of Linear-Chain Conditional Random Fields, which require linear sequences.

In order to solve this problem, each conversation tree is transformed into multiple Threads, one for each leaf-comment in the tree. The Thread is the sequence of comments from the root comment to the leaf comment. Each Thread is then treated as a separate sequence by the classifier. One consequence of this is that any comment with more than one reply will occur multiple times in the training or testing set. This makes some intuitive sense for training, as comments higher in the conversation tree are likely more important to the conversation as a whole, as the earlier a comment appears in the thread the greater effect it has on the course of conversation down-thread. We describe the process of re-merging these comment threads in Sect. 4.3.

### 3.3 Features

Each comment in a given sequence was represented as a series of features. For this experiment, we experimented with three different sets of features: 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 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 documents 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:

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

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

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

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  $\Sigma$  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.

These features were previously shown in [13] to be useful in the task of topic-modelling in email conversations. However, in contrast to [13], where similarity was calculated between adjacent sentences, these metrics were adapted to calculate similarity between adjacent comments. Where similarity metrics are used, each comment has one feature for each metric calculated between the previous comment and the next comment in the sequence, yielding 6 similarity features in all.

**Conversational Features** The conversational features capture information about the how the comment is situated in the conversation as a whole. The list is as follows:

*ThreadIndex* The index of the comment in the current thread. 0 indicates the first comment in the thread, which is the root comment of the conversation tree.

*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* and *AvgReplySentLength* The average length of replies to this comment in words and sentence length.

*TotalReplyWordLength* and *TotalReplySentLength* The total length of all replies to this comment in words and sentence length.

### 3.4 Training

For the CRF training and testing, the popular Natural Language Machine Learning toolkit MALLET<sup>2</sup> was used. A 1000-article subset of the entire Slashdot corpus was divided 90%-10% between the training and testing set. The training set consisted of 93,841 Threads from 900 articles, while the testing set consisted of 10,053 Threads from 100 articles.

## 4 Results

### 4.1 Classification

Our goal in this experiment was to be able to detect comments which had been identified as GOOD by Slashdot moderators. This suggests that the problem should be formulated as a binary classification problem, where we only seek to

---

<sup>2</sup> <http://mallet.cs.umass.edu/index.php>

classify good comments, rather than the full set of moderation classes provided by the corpus (see Sect. 3.1). Experiment 1 was to train the CRF using data where the full set of moderation labels had been grouped into *GOOD* comments and *NOT*. The Conditional Random Field classifier was trained on the full set of features presented in Sect. 3.3. The Confusion-Matrix of this experiment is presented in Tab. 4.1. We can see that the CRF performs well on this formulation of the task, with a precision of 0.818 and recall of 0.839. This compares very favourably to a baseline of assigning *GOOD* to all comments, which yields a precision score of 0.563.

**Table 1.** Confusion-Matrix for CRF trained for binary classification of *GOOD* comments (Experiment 1) trained on the full feature set. The y-axis is the correct (“gold-standard”) label, whereas the x-axis is the label selected by the classifier. Underneath are the precision, recall and f-score values for classification of *GOOD* comments.

	NOT	GOOD
NOT	5991	1965
GOOD	1426	8814
P:	0.818	
R:	0.861	
F:	0.839	

## 4.2 Feature Analysis

To investigate the relative importance of the 3-types of features (unigrams, similarity, and conversational) we experiment with training the classifier with different groupings of features. The results of this feature analysis is presented in Tab. 4.2. Interestingly, all three sets of features can provide relatively good results by themselves, but the similarity and conversational features greatly out-perform the unigram features. Similarity features have a slight edge in terms of recall and f-score, while the Conversational features provide the edge in precision, seeming to dominate Similarity features when both are used. In fact, the results of this analysis seem to show that whenever the conversational features are used, they dominate the effect of the other features, since all sets of features which include Conversational features have the same results as using the Conversational features alone. Adding unigrams features, similarity features or both has no effect on the results of the conversational features alone. This would seem to indicate that most relevant factors in deciding the quality of a given comment are conversational in nature, including the number of replies it receives and the nature of those replies. This effect could be reinforced by the fact that comments which have previously been moderated as good are more likely to be read by future readers, which will naturally increase the number of comments they receive in reply. However, since the unigram- and, more notably, similarity-features can

still perform quite well without use of the conversational features, our method is not overly-dependent on this effect.

**Table 2.** Feature analysis of binary classification task. Each row consists of a selection from the 3 classes of features (unigrams, similarity and conversational). Also provided for comparison is the baseline of labelling all *GOOD*, and the post-hoc grouping which we present in Sect. 4.4.

	P	R	F
all_good	0.563	1.000	0.720
fg_posthoc	0.608	0.718	0.658
uni	0.708	0.699	0.703
sim	0.802	<b>0.900</b>	<b>0.848</b>
conv	<b>0.818</b>	0.855	0.836
uni_sim	0.780	0.847	0.812
uni_conv	<b>0.818</b>	0.855	0.836
sim_conv	<b>0.818</b>	0.855	0.836
uni_sim_conv	<b>0.818</b>	0.855	0.836

### 4.3 Re-Merging Conversation Trees

As described in Sect. 3.2, conversation trees were decomposed into multiple threads in order to cast the problem in the form of sequence labelling. The result of this is that after classification, each non-leaf thread has been classified multiple times, equal to the number of sub-comments of that comment. These different classifications need not be the same, ie. A given comment might well have been classified as *GOOD* in one sequence and *NOT-GOOD* in another. We next recombined these sequences, such that there is only one classification per comment. Comments which appeared in multiple sequences, and thus received multiple classifications, were marked *GOOD* if they were classified as *GOOD* at least once.

$$class(c_i) = \begin{cases} GOOD & \text{if } |\{c_i \in C : c_i = good\}| \geq 1 \\ NOT & \text{otherwise} \end{cases}$$

where  $C$  is the set of classifications of comment  $i$ . This was compared to similar metrics such as a majority-vote metric ( $GOOD$  if  $|\{c_i \in C : c_i = good\}| \geq |\{c_i \in C : c_i = bad\}|$ ), and performed the best (though the difference was negligible).

There are two ways to evaluate the merged classifications. The first way is to reassign the newly-merged classifications back onto the thread sequences. This preserves the proportions of observations in the original experiments, which allows us to determine whether merging has affected the accuracy of classification. Doing so showed that there was no significant effect on the performance of the classifier; precision and recall remained .818 and .861, respectively.

The other method is to look at the comment-level accuracy. This removes duplicates from the data, and gives the overall accuracy for determining the classification of a given comment. The results of this are given in Table 4.3. The precision and recall in this measure are significantly lower than in the thread-based measure, which indicates that the classification of "leaf" comments tended to be less accurate than that of non-leaf comments which subsequently appeared in more than one thread. The precision of .700 is still much greater than the baseline of assigning *GOOD* to all comments, which would yield a precision of .297. This indicates that our approach can successfully identify good comments.

**Table 3.** Confusion-Matrix for the re-merged conversation trees, yielding one classification per comment.

	NOT GOOD	
NOT	4160	467
GOOD	862	1090
P:	0.700	
R:	0.558	
F:	0.621	

#### 4.4 Finer-Grained Analysis

The first experiment was conducted with the full range of comment moderation labels. A confusion-matrix of the results on the test set is given in Tab. 4.4. The overall accuracy for this experiment (being the frequency with which the correct label was chosen) was 0.342. This is very low, lower even than the baseline of selecting the most common label (*NONE*) for all comments, which would yield an accuracy of 0.392. Clearly, the fine-grained differences between the categories are not captured by the features provided.

By performing a post-hoc grouping of these fine-grained classes into *GOOD* and *NOT-GOOD*, we can compare these results to the earlier Experiment 1 (sect 4.1). The confusion matrix for this post-hoc grouping is presented in Table 4.4. Note that this does not represent results trained on data which has been grouped, but a post-hoc grouping of the results from the first experiment which used all categories. Here we see somewhat much more promising results. The precision for identifying good comments is 0.608, with a recall of 0.718. However, the baseline of assigning the most-common category (which is now *GOOD*) yields a precision of 0.563. The classifier beats the baseline, but only just.

## 5 Conclusion and Future Work

In this work, we have presented an approach to identifying high-quality comments in blog comment conversations. By casting the problem as one of binary

**Table 4.** Confusion-Matrix for Experiment 1, using all user-generated comment labels. Under each column is the precision and recall value for that label.

	NON	INS	INT	INF	FUN	FLA	RED	OFF	TRO
NON	2850	1649	680	713	1098	31	5	26	79
INS	1239	2297	786	291	277	25	0	1	108
INT	723	1008	415	101	149	3	0	8	14
INF	444	530	211	206	102	1	2	0	26
FUN	262	412	85	39	440	4	1	2	28
FLA	77	172	14	12	13	2	0	0	11
RED	20	15	5	1	5	0	0	0	0
OFF	44	24	14	22	26	0	0	1	2
TRO	58	148	14	36	72	1	0	6	10
P:	0.499	0.367	0.187	0.145	0.202	0.030	0.000	0.023	0.036
R:	0.400	0.457	0.171	0.135	0.346	0.007	0.000	0.008	0.029

**Table 5.** Confusion-Matrix for post-hoc grouping of data from Experiment 1, trained on the full label set.

	NOT GOOD
NOT	3223 4733
GOOD	2891 7349
P:	0.608
R:	0.718
F:	0.658

classification, and applying sequence tagging by way of a Linear-Chain Conditional Random Field, were able to achieve high accuracy. We tried classifying on the fine-grained moderation classes provided by Slashdot, but were unable to achieve acceptably high accuracy on this task. Also presented was a new corpus of blog comments, which will be useful for future research.

Future work will focus on refining our ability to classify comments, and incorporating this into an abstractive summarization system. Now that we have a reliable means to identify high-quality comments, this can form a component of future work on Abstractive Summarization of blog comment conversations. However, in order to be useful for this task, it would be preferable to have finer-grained classification than just *GOOD* and *NOT-GOOD*. Therefore, we will work on ways to retain some of the finer grained classes provided in the Slashdot corpus. The groupings described in this work took any comment with a positive moderation class to be a *GOOD* comment, regardless of the moderation score assigned (as long as the score was at least +2). This resulted in approximately 56% of comments being labelled as *GOOD*. It would be useful to investigate how accuracy is affected if the threshold for a Good comment is increased to, perhaps +3 or +4. This might increase the "certainty" that a comment belongs to the label with which it was moderated. We could also try classifying with the moderation scores as labels, rather than the moderation classes.

## References

1. Chung G.: Sentence retrieval for abstracts of randomized controlled trials. In: BMC Medical Informatics and Decision Making. 2009, 9:10.
2. FitzGerald N., Carenini G., Ng R.: ASSESS: Abstractive Summarization System for Evaluative Statement Summarization, The Pacific Northwest Regional NLP Workshop (NW-NLP), 2010 (extended abstract), Feb 2010
3. Hirohata K, Okazaki N, Ananiadou S, Ishizuka M: Identifying Sections in Scientific Abstracts using Conditional Random Fields. In: Proceedings of the Third International Joint Conference on Natural Language Processing: January 2008; Hyderabad, India 2008:381-388.
4. Jurafsky D., Martin J.: Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Upper Saddle River, NJ: Pearson Prentice Hall, 2009. Print.
5. Kim S., Cavedon L., Baldwin T.: Classifying dialogue acts in one-on-one live chats. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10) Cambridge, MA, USA.
6. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. 18th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (2001) 282-289
7. McCallum, A.: MALLET: A Machine Learning for Language Toolkit. <http://mallet.cs.umass.edu>.
8. Murray G., Carenini G.: Summarizing Spoken and Written Conversations. In: EMNLP 2008, Waikiki, Hawaii.
9. Murray G., Carenini G., Ng R.: Generating Abstracts of Meeting Conversations: A User Study. International Conference on Natural Language Generation, (INLG 2010), 2010

10. Poole D., Mackworth A.: *Artificial Intelligence: Foundations of Computational Agents*. New York: Cambridge UP, 2010. Print.
11. Shen D., Sun J., Li H., Yang Q., and Chen Z.: Document Summarization using Conditional Random Fields. In: *IJCAI '07: International Joint Conferences on Artificial Intelligence*.
12. Sutton C., McCallum A.: *An Introduction to Conditional Random Fields*.
13. Joty S., Carenini G., Murray G., and Ng, R.: Exploiting Conversation Structure in Unsupervised Topic Segmentation for Emails. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, MIT, Massachusetts, USA.