Don't Be Spoiled by Your Friends: Spoiler Detection in TV Program Tweets *

Sungho Jeon †
The Attached Institute of Electronics and Telecommunications Research Institute
Daejeon, South Korea
sdeva14@gmail.com

Sungchul Kim
Dept. of Computer Science and Engineering
POSTECH
Pohang, South Korea
subright@postech.ac.kr

Hwanjo Yu ‡
Dept. of Computer Science and Engineering
Dept. of Creative IT Engineering
POSTECH
Pohang, South Korea
hwanjoyu@postech.ac.kr

Abstract

Providing a convenient mechanism for accessing the Internet, smartphones have led to the rapid growth of Social Networking Services (SNSs) such as Twitter and have served as a major platform for SNSs. Nowadays, people are able to check conveniently the SNS messages posted by their friends and followers via their smartphones. As a consequence, people are exposed to spoilers of TV programs that they follow. So far, there are two previous works that explored the detection of spoilers in texts, not SNS: (1) keyword matching method and (2) machine-learning method based on Latent Dirichlet Allocation (LDA). The keyword matching method evaluates most tweets as spoilers; hence its poor recall performance. The other method based on LDA, although successful on large text, works poorly on short segments of text such as those found on Twitter and evaluates most tweets as non-spoilers. This paper presents four features that are significant in the classification of spoiler tweets. Using those features, we classified spoiler tweets pertaining to a reality TV show ("Dancing with the Stars"). We experimentally compared our method with previous methods, with our method achieving substantially higher precision compared to the keyword matching and LDA-based methods while maintaining comparable recalls.

1. Introduction

Motivation: A spoiler is anything that reveals facts such as crucial events, important reversal, or ending of the whole story. Spoilers can destroy peoples enjoyment of a movie or a TV program, particularly in the cases of thriller movies, reality TV shows, or sports broadcasts wherein the winner and loser of a contest and the final score of a match are critical to those who did not watch the program. If everyone interested in a program watches it at the originally aired time, spoilers are not a problem. Scheduling and time differences between other countries, however, mean that this is implausible particularly in the context of sports events. To try to avoid spoilers, people may stay away from certain sites on the Internet or avoid the news. Some people will even avoid talking to their friends to minimize the chances of receiving a spoiler. Unfortunately, avoiding spoilers is becoming more difficult due to the prevalence of smartphones and Social Networking Services (SNSs). SNSs are now a part of our lives; people check their SNSs and communicate with their friends even while working. The popularity of smartphones has accelerated this phenomenon. People can share their emotions and opinions anytime, anywhere via their smartphone; hence the increasing possibility of receiving a spoiler. Compared to news articles, which can often be determined to contain a spoiler before they are clicked, SNS messages are posted without any filtering; thus exposing people who use SNS to the danger of encountering spoilers. For such reason, people wishing to avoid spoilers must consider avoiding SNSs.

Our goal is to detect spoilers to allow people to avoid them when checking their SNSs to communicate with their friends. Specifically, we collected tweets related to the reality TV show Dancing with the Stars US Season 13 over a seven-week period. Using this data, we proposed distinct features to distinguish spoilers from non-spoilers by constructing a classification model for spoiler detection.

Challenges: To the best of our knowledge, there has been no spoiler detection research on tweets; existing spoiler detection research has covered long text, not short segments of text that are common in SNSs. There are two existing strands of research: a keyword matching method and a machine learning method based on Latent Dirichlet Allocation (LDA). Note, however, that the keyword matching method requires manual effort yet still results in a low success rate. In contrast, the LDA-based method does not require manual effort, but its performance on tweet data is much worse than that for long text such as movie reviews.

Contributions: We manually inspected tweets to select spoiler tweets from all tweets (total of 176,426 tweets), allowing us to split our data set into 5,618 spoiler tweets and non-spoiler tweets for the rest. Based on a manual study on spoiler and non-spoiler sets, we discovered four distinct features: *Named Entity, Frequently Used Verb, Objectivity, and Main Tense of Tweet*. Using the features, we classified spoiler tweets for a reality TV show using SVM. In our evaluation, simple comparison by accuracy was meaningless due to the imbalance between the spoiler and non-spoiler sets. Accuracy was determined by how many non-spoiler tweets were found, because spoiler tweets accounted for only 3.2%

*This work was supported by IT Consilience Creative Program of MKE and NIPA (C1515-1121-0003)

†This work is conducted while Sungho Jeon is at POSTECH

‡Corresponding author

of our working data set. To address this problem, we used f-score, which considers both recall and precision and compares the results with previous methods for spoiler detection. Our method achieved substantially higher precision while maintaining comparable recalls.

2. Model

For the sake of our goal – to detect spoiler in tweets, we consider this problem as a classification problem; i.e., to classify spoiler given a set of features. We used LibSVM which is popular open-source SVM library to achieve our goal.

2.1 Feature1: Named Entity

The name of the person leaving or surviving plays the most important role in the spoilers of DWTS. This information is the most common cause of spoiler tweets for reality TV shows. In our manual study, however, all spoiler tweets that we found had "named entity," nickname, or job, unlike non-spoiler tweets. We used the Twitter Named Entity Recognizer (Ritter et al. 2011).

If we only use frequency as a score of named entity, its influence can be changed as a function of the data size; thus, we divided the sum of the total frequency of the named entity.

2.2 Feature2: Frequently Used Verb

We observed that frequently used verbs differed in the spoiler set and non-spoiler set based on our manual study. Words related to survival or failure such as "wins", "eliminated", and "voted" are common in spoilers, whereas words such as "watch", "love", and "think" top the frequency list in the non-spoiler set. Similarly, with the named entity score, we calculate the score of frequently used verbs as the frequency of verb divided by the sum of the total frequency of frequently used verbs having a frequency of greater than five.

2.3 Feature3: Objectivity and URL

We also noticed that objective spoilers containing only the title of spoiler news articles were more common than subjective spoiler tweets, which expressed the writer's emotion. For example, an objective spoiler tweet would be, "Chaz Bono gets eliminated from Dancing with the Stars Sad news for Chan Bono fans. He was eliminated from Dance *URL"; a subjective spoiler tweet is, "Yaaaayyy!!!! Hope and Max went home on Dancing with the Stars. Now the real finals begin. #HASHTAG." We chose 50 objective spoiler tweets and subjective tweets from the spoiler set to train the LingPipe subjectivity tool. When we conducted a subjectivity analysis, we verified our hypothesis, i.e., the ratio of objective tweets in the spoiler set is higher than that in the non-spoiler set. These objective spoiler tweets usually carry a URL linking to a news article, so we considered whether a tweet has a URL. Therefore, the purpose of this feature is to detect news spoiler tweets that have only a news article title and a link without emotion.

2.4. Feature4: Main Tense of Tweet

We believe both "named entity" and "frequently used verb" play a critical role, but they are not enough. There is further consideration – in tweets such as "Kadashian will survive", "Kadashian survives", and "Kadashian survived", the same "named entity" and "frequently used verb" are used, but the "the tense of the tweet" determines whether the tweet is a spoiler. Of course, people mix up or hardly care about present and past tenses on Twitter, but the future tense is rarely used in this way. Tweets using future tense cannot be spoiler tweets since they represent wishes or desires, as with "hope" or "want".

To improve the performance of statistical machine translation, (Gong et al. 2012) identified the main tense of document. We modified the their method used to identify the tense of each document. When we applied the original work to our method, we only found about 2,000 tweets using the future tense. We see two problems. First, flawed syntax and misspelled words in tweets cause false recognition of POS tags. We modified the method to enable the detection of that false recognition because the original work only focused on verbs. Second, the original work was built on the rule-based method, yet it used very few rules to catch diverse words and phrases commonly used in daily conversation. To solve this problem, we added additional rules; our method remembers previous POS tags for recognizing "be" verbs such as is and are to catch more future representations. Furthermore, the representation of desire – such as hope or wish – involves future tense, and these representations are used to indicate future meaning; thus, we regarded these representations as future tense. Consequently, future tense tweets had frequency of 83 in the spoiler set and 13,619 in the non-spoiler set.

3. Experiment

3.1 Data

We collected tweets related to the reality TV show, "Dancing with the Stars, US. Season 13" (DWTS) during the period 2011-10-14 $\sim$ 2011-12-01. DWTS was aired 12 times during this period, and the number of collected tweets was 176,426. We first identified 5,618 spoiler tweets manually, and then investigated the differences between spoiler and non-spoiler tweets to determine the features that should be used to classify spoilers.

3.2 Stepwise Regression

We evaluated the strength and usefulness of our method on a real-world data set. First, we performed stepwise-regression to verify the usefulness of our proposed features (Table 1). The P-value of four features – along with each positive coefficient of terms in the regression equation – proves that our method is meaningful. Features were added to the regression in the order of "named entity" (NE), "frequently used verb" (Fverb), "objectivity + URL" (Obj), and "tense of tweet" (Tense).

3.3 Parameter Tuning

As we mentioned earlier, accuracy is not appropriate as a performance measure, since the size of the spoiler set ac-

Table 1: Stepwise Regression Results

Step	Variable entered	Co-efficient	β	Standard error	P-value
0	-	-0.036	-	-	-
1	NE	1.85	1.85	0.04	0
2	Fverb	3.05	3.05	0.02	0
3	Obj	0.03	0.03	< 0.001	0
4	Tense	0.03	0.03	0.002	< 0.001

Table 2: Classification Experiment Results

Step	Entered feature	Recall	Precision	F-score
1	Named Entity	0.3084	0.8291	0.4496
2	+Verb	0.5653	0.8611	0.6826
3	+Objectivity	0.7699	0.7957	0.7825
4	+Tense	0.7697	0.7987	0.7839

counted for only 3% of the entire data set, the accuracy of the model could be over 90% without referring to any spoiler tweet. To address this problem, we decided to use f-score instead as a measure of accuracy and to report recall and precision as well. The classification results were influenced by training and testing data sets; we reported the result of three-fold cross validation. The performance of Support Vector Machine (SVM) is significantly influenced by the kernel type and parameters, so we performed our experiment with four kernel types and various parameters. We modified two parameters – cost and γ – in our experiment as changed parameters: cost = $2^{-5} - 2^{15}$, and $\gamma = 2^{-15} - 2^{3}$. Based on our results, the performance of SVM culminated with f-score of 0.7912.

3.4 Comparison of Features

We observed a change in performance when we added features in the order of the result of stepwise regression (Table 2). The f-score was 0.4496 when we experimented using only "named entity", persistently increasing as we added additional features. Finally, our model had an f-score of 0.7839 when we used all features.

Additionally, we measured the performance of each feature to test which one wields the most significant influence (Table 3). We thought that "named entity" or "frequently used verb" is most powerful, because the coefficients of these features in stepwise regression were higher than others. Moreover, based on our intuition, these two features are most important. Contrary to our expectation, however, we could get the highest f-score (0.6776) when measuring with only tense and second highest f-score (0.6078) with objectivity + URL. We attribute this to the binary scoring of "tense" and "objectivity + URL" features, unlike other features that used scoring as frequency divided by total frequency. Although we normalized the score of "named entity" and "frequently used verb", binary scoring wields a more powerful influence when used individually similar to the keyword matching filtering method.

Table 3: Performance When Each Feature Was Used Individually

Feature	Recall	Precision	F-score
Named Entity	0.3084	0.8291	0.4496
Verb	0.4306	0.8726	0.5765
Objectivity	0.5844	0.6333	0.6078
Tense	0.9852	0.5164	0.6776

4. Comparision of Baseline Methods

Despite the absence of research studies on spoiler detection in Twitter at present, there are two fields of research related to spoiler detection in other domains: 1) keyword-matching method 2) machine-learning method based on LDA.

4.1 Keyword-Matching Method

First, the simplest method, a keyword-matching method, simply filters out according to important words in their domain, such as the actor name in a drama or the match score at a sporting event (Nakamura and Tanaka 2007). Note, however, that this method is not realistic because the method of choosing these important words is not determined. We implemented this method by choosing important words such as "named entity" and "frequently used verb" to filter out spoilers from tweets. The advantage of this method is the high recall performance, i.e., blocking all tweets that carry even a small possibility of being a spoiler. Paradoxically, however, this advantage also makes for a significant disadvantage – it has very low precision performance because it regards most tweets as spoilers; that is, it flags many false positives. When we compared this method with ours, we found it to have higher recall performance, but much lower precision than ours; hence the lower f-score as well. As such, this method – although the simplest – is also an impractical method.

4.2 Machine-Learning Method Based on LDA

The other relevant research is a machine-learning method based on LDA (Guo and Ramakrishnan 2010). They ranked 1,000 IMDB movie comments via predictive perplexity and filtered out spoilers when they regarded the top N comments as spoilers and considered report performance of recall and precision. Note, however, that ordinary LDA is not applicable to short texts such as tweets (Rosa et al. 2011); performance in terms of recall and precision is not good even for movie comments, which generally consist of several sentences. One problem with LDA is that we have to investigate the appropriate number of topics. Various methods were proposed to solve this problem – we used perplexity, which is widely used in the language-modeling community, as well as the original work to predict the best number of topics. Calculating the average per-word held-out likelihood, predictive perplexity measures how the model fits with new documents; lower predictive perplexity means better fit. As with the original work, we used the classic perplexity method, which can be calculated as: $P(W|M) = \exp -\frac{\sum_{m=1}^{M} \log p(\tilde{w_m}|M)}{\sum_{m=1}^{M} N_m}$. We increased the topic number

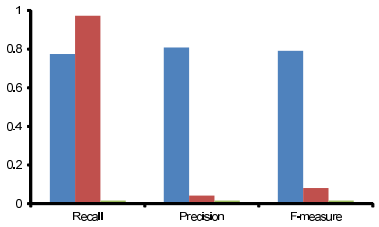

Figure 1: Comparison between our model, baseline1, and baseline2. Blue bar (Left): Our model; Red bar (Middle): Baseline1; Green bar (Right): baseline2.

from 50 to 650 in steps of 50 to find the appropriate number of topics; thus, we came up with 450 as the most appropriate number of topics. We conducted the baseline2 experiment using three-cross validation as our model. (we used 0.1 as document topic prior and 0.01 as the topic word prior.)

4.3 Comparison: Recall, Precision, F-Score

We compared with baseline1 which was implemented using named entity and frequently used verb to filter out spoilers by keyword matching and baseline2 and our model (Figure 1). Baseline1 had the highest recall, but much lower precision than in our model; thus our model had the highest f-score. Compared to our model, the recall was higher and the precision was 0.76 lower in baseline1. The f-score of our model was also 0.71 higher than baseline1.

Baseline2 based on LDA delivered poorer results in all aspects of performance compared to our model. We attribute such to the fact that our target is different from that of the original work, i.e., movie comments consisting of several sentences, unlike tweets where one sentence usually makes up the entire document. Thus, their method is not applicable to tweets. The performance of baseline2 was significantly lower than others, requiring much more time to execute.

4.4 Comparison: Processing Time

Lastly, we compared the processing time of our method and baseline methods (Table 4). Practically, this system can be used as plugin of web browser or SNS. To do that, processing time is one of the most important factors; it is worth sacrificing performance to get better processing time performance if we consider making a plugin form. Obviously, baseline 1 is fastest and simplest; it just reads a tweet once and blocks the tweet if it has a danger keyword. In contrast, baseline 2 needs more than 1 day. One of the disadvantages of LDA is slow processing time given the bigger data size, so baseline 2 cannot be applied as a plugin form. The processing time of SVM is also significantly influenced by parameters aside from performance. As a result of our parameter tuning, the best parameters are cost = 2^{15} and $\gamma = 2^3$, but it takes a very long time with these parameters. We could get faster processing time at the expense of performance by changing parameters. As a result, we could get a relatively high f-score in 1 minute by sacrificing of f-score when cost = 2 and $\gamma = 2$. Thus, if we adjust the parameters, we can get a relatively high f-score and a fast processing time which are enough to apply the plug-in form.

Table 4: Comparsion of Processing Time Between Our Model and Basline Methods with Three-fold Cross Validation

Method	Processing Time	F-score	$\log_2 C$	$\log_2 \gamma$
Baseline1	$<$ 1 min	0.0807	-	-
Our Model	1.1 min	0.7568	1	1
Our Model	14.5 min	0.7668	10	3
Our Model	192 min	0.7912	15	3
Baseline2	$>$ 1440 min	0.0165	-	-

5. Conclusion

This paper has proposed a method for detecting spoilers in reality TV show tweets by SVM. Our main goal is to detect spoilers using distinguishable features. We made spoiler sets manually and propose four major features, “named entity”, “frequently used verb”, “objectivity + URL”, and “tense” based on our manual study. These features show a distinct distribution in spoiler sets and non-spoiler sets. There is currently no spoiler detection research focusing on tweets, but there are two works related to spoiler detection. The first method, a keyword matching method, can achieve high recall performance but considers most tweets to be spoilers; hence its very low f-score. The second method, which is based on LDA, is not applicable to short texts as used in our experiments. According to our experiments, we found that our method is superior in terms of f-score. We compared baseline 1 and baseline 2 in terms of recall, precision, and f-score. In particular, our model achieved significantly better performance in terms of precision.

References

Gong, Z.; Zhang, M.; Tan, C.; and Zhou, G. 2012. N-gram-based tense models for statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 276–285. Association for Computational Linguistics.

Guo, S., and Ramakrishnan, N. 2010. Finding the storyteller: automatic spoiler tagging using linguistic cues. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 412–420. Association for Computational Linguistics.

Nakamura, S., and Tanaka, K. 2007. Temporal filtering system to reduce the risk of spoiling a user’s enjoyment. In *Proceedings of the 12th international conference on Intelligent user interfaces*, 345–348. ACM.

Ritter, A.; Clark, S.; Etzioni, O.; et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1524–1534. Association for Computational Linguistics.

Rosa, K.; Shah, R.; Lin, B.; Gershman, A.; and Frederking, R. 2011. Topical clustering of tweets. *Proceedings of the ACM SIGIR: SWSM*.