Spoiler Alert: Machine Learning Approaches to Detect Social Media Posts with Revelatory Information

Jordan Boyd-Graber
University of Maryland
iSchool and UMIACS
jbg@umiacs.umd.edu

Kimberly Glasgow
JHU Applied Physics Lab
UMD iSchool
kimberly.glasgow@jhuapl.edu

Jackie Sauter Zajac
University of Maryland
iSchool
jsauter@umd.edu

ABSTRACT

Spoilers—critical plot information about works of fiction that "spoil" a viewer's enjoyment—have prompted elaborate conventions on social media to allow readers to insulate themselves from spoilers. However, these solutions depend on the conscientiousness and rigor of Internet posters and are thus an imperfect system. We create an automatic alternative that could alert users when a piece of text contains a spoiler. An automated spoiler detector serves not only as an additional protection against spoilers, but it also contributes to important problems in computational linguistics. We develop a new dataset of spoilers gathered from social media and create automatic classifiers using machine learning techniques. After establishing baseline performance using lexical features, we develop metadata-based features that substantially improve performance on the spoiler detection task.

Author Keywords

Information Organization and Representation: supervised learning, classification, feature engineering, social media

THE NEED FOR AUTOMATIC SPOILER DETECTION

"Spoilers" have the potential to kill the joy of an entertainment experience: Darth Vader is Luke's father. Andy Dufresne escapes from Shawshank prison. Michael Corleone becomes the next Godfather. Defined as essential information about a work of fiction (e.g., a movie, television show, or book), spoilers—if known—can decrease the audience's anticipation and excitement about the work, diminish enjoyment of the work when consumed, or deter them from consuming the media at all (Tsang and Yan, 2009). Pleasurable uncertainty and curiosity about what will happen is replaced with mundane knowledge (Wilson et al, 2005; Lowenstein, 1994).

Avoiding spoilers has become a common problem for social media users who have not yet viewed or read about the plot twist. Because fans enjoy talking about these works (especially on social media), we shield others who have not yet viewed or consumed the work. There is a perceived—if not actual—harm to hearing spoilers before experiencing a work (Leavitt and Christenfeld, 2011).

In the age of the "second screen"—when users tweet, post, or blog about television shows in real time—and where private Netflix viewings replace "appointment television" and synchronized cinema first releases, elaborate mechanisms have evolved to protect viewers from spoilers. IMDb explicitly defines spoilers and its message boards etiquette states "If your message includes a spoiler you must announce it in the subject of your post" (Amazon.com, 2013). Failure to follow this policy can result in loss of posting privileges and deletion of posts. Similarly, users who post to the websites *Rotten Tomatoes* and *Reddit* include "Spoiler" in the subjects of their posts. *Lostpedia*, a wiki for the television show *Lost*, adopted an extensive spoiler policy, banning spoilers from all portions of the wiki (articles, talk pages, user pages, theory pages, user blogs, and comments). Images from upcoming episodes are banned, and even official press releases and podcast transcripts that could potentially contain spoilers are marked with a bright red template.

Beyond specific sites adopting policies regarding spoilers, others have weighed in on the issue. Some authors and bloggers have proposed statutes of limitations on sensitivity to spoilers. More broadly, many good Internet citizens have developed a personal convention of issuing "spoiler alerts" to protect others.

But despite these rich conventions, spoilers remain a real problem in online conversations. In part, this is because social motivations might run counter to the conventions of spoiler alerts. Additionally, the technological context of online interaction often removes the ability of recipients to protect themselves from spoilers. We discuss both of these aspects of spoilers below.

First, the motivations of posters in social media. Not all users of social media are conscientious or sensitive. In particular, users' motivations (Sanders, 1987) might be at odds with the conventions that have led to spoiler alerts. They may wish to influence others' reactions, manage interpersonal relationships, or establish self-image. Participants in communicative exchanges thus may have different, conflicting motivations (Tracy, 2001). While sharing one's knowledge and experiences regarding a creative work through writing a review on social media can be characterized as informational gift giving, which is associated with motives such as altruism and

ASIST 2013, November 1-6, 2013, Montreal, Quebec, Canada.

reciprocity, less altruistic exchanges can feature attempts to achieve greater status (Lampel and Bhalla, 2007), commonly by displaying superior knowledge. The latter in particular may clash with the review reader's goals and motivations to learn enough about a movie to decide whether to see it, without being exposed to spoilers.

The motivations of social media users then interact with the technological context. Social media distorts the traditional conventions of discourse. Because social media are technologically-mediated, traditional techniques to ward off spoilers are unavailable—a recipient cannot interrupt a partner to say, "wait, I haven't seen that yet". Instead, new mechanisms (Herring, 2013) must protect personal and social goals.

The goal of our work is to add these new interaction mechanisms to improve users' interactions online by providing automatic methods to detect text that is a spoiler. Identifying potential spoilers adds to the human repertoire of response in social media, preserving the ability to shape the nature of interaction in an asynchronous context.

In the remainder of the paper, we review existing approaches for detecting spoilers and show how detecting spoilers fits within the larger computational linguistics research area. We then describe our new dataset that allows us to discover classifiers that detect spoilers. This dataset was assembled from a large wiki site devoted to popular culture, TV Tropes. We next develop machine learning algorithms that generalize from examples of spoilers to new spoilers from media that the algorithm has never seen before.

We show that simple metadata about the work in question can dramatically improve the performance of the spoiler detection algorithms. We conclude the paper with error analysis and a discussion of applications and extensions.

TRANSITIVITY, NARRATIVE, AND SPOILERS

Previous work has looked at automatically detecting spoilers. Guo and Ramakrishnan (2010) used manual annotation and supervised classification to detect spoilers in movie summaries. In contrast, our classifiers are trained on large, diverse data created by many heterogeneous, untrained annotators collaboratively building an online resource.

Other work has taken a more heuristic approach and focused on high-recall solutions. Golbeck (2012) created a "Twitter Mute" button that used a knowledge resource to discover terms related to a topic of interest and then intercept relevant messages.

More broadly, the problem of detecting "spoiler" text is connected to core problems in linguistics and natural language processing: Transitivity and narrative. We first discuss Transitivity (Hopper and Thompson, 1980),[1] a property of a clause measuring how impactful, deliberate and complete the action it describes is. An action like "kill" is far more Transitive than an action like "think" would be. "Kill" is kinetic, completed, volitional, done with agency, real (not imagined), and powerfully affects its object. An action like "think" is far less Transitive. People you think about may not know you are thinking about them; they are not affected by your thinking about them; your thinking about them may not have a clear starting point and stopping point, and you may not have made a conscious decision to think about them. A central component of spoilers is that they are inherently Transitive. Transitive actions advance the narrative, causally linking actors, actions, and outcomes in recognizable schemata (Schank and Abelson, 1977).

While there have been attempts to automatically detect Transitivity (Greene and Resnik, 2009; Madnani et al, 2010), these techniques have faltered because of a lack of extensive training data for traditional machine learning techniques. Machine learning approaches typically require large amounts of annotated data to be successful (Halevy et al, 2009), which is difficult to obtain for such a niche linguistic concept. However, the ubiquity of spoilers and the explicit annotation of spoilers in social media (e.g., users providing a "spoiler alert") allows researchers to build larger datasets to capture this linguistic phenomenon.

Another key concept from natural language processing that is connected to spoilers is automatic narrative detection. A spoiler is, by definition, an event that is *later* than the viewer's knowledge of the current work. Thus, spoiler detection depends on discovering which sentences refer to later events. While supervised (Bethard et al, 2012) and unsupervised systems (Chambers and Jurafsky, 2011) have shown promise for extracting narrative events, such techniques have again suffer from a paucity of data.

CROWDSOURCED SPOILER ANNOTATION

What sets our approach apart is that, rather than relying on manual annotation (Guo and Ramakrishnan, 2010) or naïve wordlists (Golbeck, 2012), we rely on the insights of thousands of web users to generate examples of spoilers from the web site TV Tropes.

The name of the site TV Tropes comes from "tropes", from the Greek word for "turn", that characterize the common plot twists, character types, or recurring narrative structures that recur in many works of fiction. The goal of TV Tropes is to define a set of common tropes (e.g., ("There's No Place Like Home", "Delayed Explosion") and meticulously assign examples from film, books, video games, and other media to each trope. For example, the trope "There's No Place Like Home" references the film *The Wizard of Oz*, from which the trope is named. In addition, the that trope page also includes examples of the trope from other works, such as the novel *Alice in Wonderland*, the film *Back to the Future*, and the television show *Gilligan's Island*.

Because TV Tropes focuses on describing the plots of works of fiction, it has developed an ingenious system to allow users to browse the site while protecting themselves from spoilers. When users browse the site, spoilers are covered by an opaque box. These annotations are added to the site by the users

[1]Following convention in linguistics, we write the Hopper and Thompson definition of Transitivity with a capital letter. This is to distinguish *grammatical* transitivity (whether a verb takes a direct object or not) with the more general Transitivity we describe here.

who curate the user-generated content. While not directly comparable to academic annotation protocols, TV Tropes has developed a comprehensive system to define what a spoiler is. The site defines spoilers as:[2]

- **Important**: only important plot developments warrant being declared a spoiler,
- **Fictional**: events that are a part of "real" history are not a spoiler, and
- **Not Universally Known**: mythology, religion, and out-of-copyright works (e.g., Shakespeare and Goethe) are assumed to be known by any literate reader.

TV Tropes divides its pages into two main types: trope pages and work pages. Trope pages serve as a concordance of all of the examples of a particular trope (e.g., "There's No Place Like Home", mentioned above). Work pages go in the other direction; they list all of the tropes that appear in a *single work*. We focus on the latter, as spoilers are applied more uniformly within work pages (some tropes particularly associated with spoilers—Death, Love, Betrayal, and Twist Endings—are populated almost entirely by spoilers, so TV Tropes has elected not to hide spoilers on those trope pages).

The content of TV Tropes is shared under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 license.[3]

SUPERVISED CLASSIFIERS FOR DETECTING SPOILERS

As mentioned above, we focus on *works* pages that list, for a given work of fiction, the tropes contained in that work. We collect all of the pages associated with a single television show and then extract individual entries—examples of an instance of a trope within that television show. Some of these entries are marked as containing a spoiler.

Spoilers are marked in TV Tropes pages by enclosing a substring of an entry in a HTML span tag that marks words that should be hidden to prevent a reader from being spoiled. These annotations can conceal entire sentences, single words, or phrases. The following are examples of entries from the work page associated with the U.S. television show *The Office*, with spoilers underlined:

- When it comes up that his girlfriend Katy was a high school cheerleader, he briefly doesn't believe it and overall seems to see it as a negative, in preference to Pam the "art geek". He dumps her near the end of the episode.
- Dwight, who manages a one night stand with one of Pam's friends at her and Jim's wedding and handles it with a level of expertise that suggests experience ...
- Partway through Season 7, Holly realizes her relationship with AJ isn't going to work out and she gets together with Michael. The two are Put on a Bus after getting engaged.

What is marked as a spoiler emerges over time. If the first author of an entry is overzealous or too conservative in marking spoilers, subsequent users will edit the post until a consensus annotation is achieved. Because users do not always agree on exact spans that should be marked as a spoiler, we focus on identifying sentences that **contain** a spoiler.

We then remove all of the entries associated with a show that are not a well-formed sentence. While this is a difficult problem, we found that applying a part of speech tagger (Bird et al, 2009) to an entry and removing any entry that did not have a word tagged as a verb resulted in a relatively effective first-pass filter to ensure that each entry has an associated verb.

We pause for a moment to consider how removing these sentences would affect our goal of building an automatic spoiler detector for use in social media. First, it is unreasonable to assume that all social media posts will be complete sentences. However, the same properties of non-sentences that frustrate our automatic methods also mean that these spoilers are often inscrutable without additional context. While "Stay-Puft" is indeed a spoiler for the film *Ghostbusters*, without additional context it's a fairly benign spoiler.

Creating a Balanced Dataset

Thus, we consider any sentence that has any overlap with a spoiler to contain a spoiler tag. This results in nearly 10 thousand spoilers. We randomly select an equal number of non-spoiler sentences to create a balanced dataset (an equal number of spoiler and non-spoiler sentences). We then divide these sentences into training (70%), test (10%), and two development sets (10% each) (Resnik and Lin, 2010).

Sentences are allocated to folds such that a single work only appears within a single fold. Thus, no sentences from *Doctor Who* will appear in both the training and test split. Sentences from a work only appear in a single fold. This forces our technique to generalize across works and not learn—for example—proper nouns associated with spoilers *within a single work*.

We balance the data in this way to prevent classifiers from learning that **nothing** is a spoiler (as most statements are not). This allows us to balance precision and recall, and provides convenient random guessing for subsequent experiments (guessing a single class would get 50% accuracy).

Data Representation and Classifier

Our goal is to take examples of spoilers and from those examples create a system that can determine, given a new example sentence, whether it is a spoiler or not. This is a text classification task. Text classification as an approach dates to the 1960s (Maron, 1961), and machine learning techniques provide the mathematical foundation for effective automatic text classification. In contexts where the quantities of text to be classified are large, the requisite response time is short, or the cost and availability of human raters prohibit a manual approach, machine learning techniques are the only cost-effective solution (Sebastiani and Ricerche, 2002). We train a classifier that, given a representation of a document $\boldsymbol{x}$ can produce a prediction $f(\boldsymbol{x})$ which is either "spoiler" or "not spoiler". If we had such a system, a user could use this preprocessing technique to filter their social media stream. Like a spam filter, our spoiler filter could detect sentences likely to detract from a user's enjoyment.

[2]We summarize their definitions here, but the full policy is available at **http://tvtropes.org/pmwiki/pmwiki.php/Main/HandlingSpoilers**

[3]Our dataset available from **http://umiacs.umd.edu/~jbg/downloads/spoilers.tar.gz**.

This results in two questions: how do we represent a candidate sentence $\boldsymbol{x}$, and how do we learn the function f that tells us whether a sentence is a spoiler or not? We represent sentences as a point in a d-dimensional vector space and learn our function f as a support vector machine (Joachims, 1999, **SVM)**.

Each example $\boldsymbol{x}_n$ is represented as a vector with d-dimensions (one dimension for each feature). Each dimension is associated with a real number $x_{d,i}$ that represents how much that example is associated with that feature. We will develop many of these features in subsequent sections, but for the moment we focus on the ubiquitous unigram feature, which encodes how often a word type appears in a document. Each dimension i of our example vector $\boldsymbol{x}_n$ is

$$x_{d,i} = \begin{cases} \text{count}(d,i) & \text{if word}_i \text{ in sentence } d \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $\text{count}(d,i)$ is the number of times word i appears in the d^{th} document. We then normalize each vector to ensure that we are not affected by the number of words in a sentence and to ensure convergence of the methods used to discover classifier. We create a new normalized feature vector for each sentence,

$$\dot{x}_d \equiv \frac{x_d}{||x_d||_2}, \quad (2)$$

which divides each dimension by the length of the original vector; features with a value of zero are left unchanged. Thus, if an example only contains the words "he" and "died", the vector would have two entries that were non-zero ("he" and "died") both with weight around 0.71 (because the length of the original vector, $||x_d||$, was $\sqrt{2}$).

We then use examples, represented as in Equation 2, to train a linear-kernel SVM (Ben-Hur and Weston, 2009). This produces a function f that can take an arbitrary sentence (encoded into the same vector space, as in Equation 2) and output whether the sentence is a spoiler or not.

In practice, additional parameters must be set during SVM classification. These include the cost parameter and the number of active features to use during recursive feature elimination (Guyon et al, 2002). We select these parameters using grid search on development data (for each feature set) and apply those values to the final classifier. All numbers are reported on the test dataset.

In the next sections, we describe the features needed to learn effective classifiers.

BASELINE FEATURES

The most straightforward features we can extract are the words present in the sentence. We tokenize the sentences (Kiss and Strunk, 2006) to extract word tokens and remove words found in an English stoplist (Bird et al, 2009).

We consider three types of baseline features, which use only the information available in the sentences:

- **unigram** uses only *single word* tokens from the original sentence;
- **stem** uses only single word tokens, but reduces each token to a *stem* (Porter, 1980) of each unigram (i.e., "runs" becomes "run" and "quicker" becomes "quick"); and
- **bigram** uses all *consecutive pairs* of tokens from the original sentence. This results in a much larger feature space, but can capture important contextual information lost to the "bag of words" model.

Unigram	Bigram
show	turns out
finale	the end
killed	the show
death	end of
turns	the season
s	out that
season	one episode
end	at the
back	he s
revealed	to kill
dead	that he
kill	in the

Table 1: Most useful unigram and bigram features, sorted by information gain.

Feature Set	Number of Features	Accuracy
Unigram	20,269	0.53
Stems	16,683	0.56
Bigrams	125,523	0.60

Table 2: Baseline classifiers trained on text alone. Bigram features do better than either raw words or stemmed words.

For both the **unigram** and **bigram** features, we show the most useful features sorted by the information gain for each feature in Table 1. The information gain of a feature is the difference between the original label entropy (in this case over whether a sentence has a spoiler S) minus conditional label entropy given the value of a feature w,

$$IG(S,w) \equiv H(S) - H(S \mid w), \quad (3)$$

where the entropy is

$$H(S) = \sum_{v \in S} p(S=v) \lg p(S=v), \quad (4)$$

summed over the event space of the spoiler random variable—in this case the presence or absence of a spoiler. If seeing a feature allows you to be more certain whether a sentence is a spoiler or not, it will have a larger information gain (Quinlan, 1986).

For both the unigrams and bigrams, we see two common patterns in the useful features. Both highly transitive verbs "kill", "revealed", "to kill" and temporal expressions are common ("the end", "in the season", "end of"). Both are evidence of the strong connection between spoiler detection and the computational linguistics tasks discussed earlier: temporal ordering and detecting the Transitivity of verb expressions.

To evaluate the classifier, we take a classifier trained on the "training" fold and present it examples from the "test" fold. We present accuracy as the fraction of sentences classified as spoiler or not a spoiler. The higher this number, the better. The classifiers trained using basic feature sets are only marginally better than the most-frequent-class baseline; a classifier that always said that a sentence was not a spoiler would only have an accuracy of 0.5 (recall that we balanced the dataset during preprocessing).

Table 2 presents these results. Unigram features perform slightly better than the baseline, with an accuracy of 0.53. Stemming the words allows the classifier to generalize from the sparse training data, improving performance to 0.56. However, more information is available in bigram features, which improves the performance to 0.60.

The unigram features are able to capture many of the Transitive properties we expect spoilers to contain. Words like "kill" and "death" imply an irrevocable change of state, a hallmark of highly Transitive constructions. Unigram features are able to capture these properties, learning which events should be hidden behind a spoiler alert. Stemming unigrams improves performance by clustering different forms of verbs such as "killing", "kills", and "killed" into a single feature, "kill", which is easier for the classifier to represent.

In contrast, bigram features allow the system to discover more of the temporal aspects of the dataset. This includes both relative time markers such as "end of" and "the end" and absolute markers such as "one episode" and "the season". By adding bigram features, we are able to further improve the classifier's ability to recognize when a sentence refers to something later in a work's chronology.

FEATURES FOR DETECTING SPOILERS

In contrast to previous approaches that focused on lexical features, additional metadata can dramatically improve a classifier prediction of whether a sentence is a spoiler or not. In this section, we describe features that improved our ability to detect whether a sentence is a spoiler.

Obtaining this information required us to match entries in the TV Tropes dataset. We did this by matching television show names with two additional resources: Episode Guides and the Internet Movie Database.

Episode Guides[4] is an Internet clearing house for the airdates of television shows. It combines information from public listings, contributor information, and other websites to create a comprehensive listing of when shows aired. The authors make their assembled information freely available to interested users. Episode Guides provides us with our first air date, country of origin, and length features.

The Internet Movie Database[5] (henceforth IMDb) contains information on the individuals associated with works of media. Previous research has used IMDb's data to explore sentiment analysis (Pang and Lee, 2004) and deanonymization (Brickell and Shmatikov, 2008). While IMDb contains a wealth of data, we focus on the IMDb's classification of media into genres.

Because we have used the show names to look up these metadata, an important question is whether this is a realistic assumption for our envisioned applications. This is a problem, because if we have a sentence but do not know what show it is talking about, we cannot use these metadata features. We leave these issues for future work, as determining the associated show is traditional topic classification, a well-studied problem computational linguistics and natural language processing. Moreover, in many contexts the topic / show is known (e.g., on a board dedicated to a particular show) or can be more easily determined from wider contexts (e.g., it is easy to know if a whole conversation is about *The Walking Dead*, but the spoiler classification must be done sentence-by-sentence.

While the previous section's features were discrete, many of the features presented in this section are continuous. Rather than being present or absent (e.g., whether a word is in a document or not), these features are associated with a real number. To ensure the magnitude of these new features do not overwhelm existing features, we rescale a continuous feature i for observation n as a z-score

$$z_{n,i} = \frac{x_{n,i} - \mu_i}{\sigma_i}, \tag{5}$$

where μ_i is the average value of this feature (across all observations) and σ_i is the standard deviation of this feature (again, across all observations). For example, if the average length of shows is forty minutes and the standard deviation is twenty minutes, a sixty minute show will have a z-score of 1.0 and a twenty minute show will be -1.0.

This also has the added advantage of making the average value of a feature zero. For any work for which we lack a feature, we represent that feature value of zero and set a "feature_missing" feature indicator to be 1.0. Thus, the classifier recovers gracefully when the feature is missing (e.g., if we do not know how long a given show is, we pretend that it is the average length—forty-four minutes).

Genre

Media are often sorted by genre; IMDb categorizes every movie and television show into a set of overlapping categories. This can be a reflection of the target audience ("Family" or "Adult"), the subject ("Crime" or "Sci-Fi"), or the way the subject is presented ("Comedy" or "Short").

Table 3 shows the information gain when IMDb genre categories are incorporated as binary features into a sentence's feature vector. We set a feature for a document n to be 1.0 if and only if the show associated with the sentence is classified by IMDb as having that genre (afterward, feature vectors are normalized via Equation 2).

Length

The length of a work serves as a useful indicator of whether it is likely to have a spoiler. As seen in Figure 1a, longer works are more likely to have a spoiler than shorter works. This is a function both of format and genre. First, length is often

[4] `http://epguides.com`
[5] `http://imdb.com`

Genre	Spoilers outside genre	Spoilers inside genre	Information Gain
Mystery	0.46	0.69	0.03
Drama	0.43	0.62	0.03
Thriller	0.47	0.67	0.02
Crime	0.49	0.66	0.02
Short	0.49	0.61	0.01
Sci-Fi	0.50	0.64	0.01
Comedy	0.57	0.47	0.01
Action	0.50	0.60	0.01
Family	0.54	0.45	0.01

Table 3: The usefulness of genre features at predicting whether a sentence discussing a show contains a spoiler or not. The column "spoilers outside genre" gives the proportion of sentences with spoilers given that they are labeled as **not** being associated with a genre (by row). The column "spoilers inside genre" gives the same quantity for sentences **labeled** with that genre. Recall that the dataset average is 0.5. Some genres are more likely to have spoilers ("mystery") than others ("family").

correlated with genre; comedies are more likely to be thirty minutes, while dramas are more likely to be one hour.

However, the format on its own is also critical. Longer shows are more likely to be plot-centric, increasing the probability of spoilers.

First Aired

Because TV Tropes concerns all media, not just recent productions, there is a diverse range of works discussed on the site. If the algorithm knows when the work was created, it can better replicate the collective decision about which sentences are spoilers or not.

Figure 1b shows the relationship between the distribution over air date for sentences without spoilers and those with spoilers. More recent shows are more likely to have spoilers. This is because users are more careful about labeling these spoilers but also because of a shift from comedy to more spoiler-heavy genres.

Other Features

In addition to these features, we also added features based on

- **Country** We thought that the nation of origin might help determine if a work was likely to have a spoiler or not. However, this was the least useful feature we added.
- **Episodes** The number of episodes in a television series helped the algorithm distinguish long-running series from shorter ones. Longer series are likely to have a richer mythology and thus have spoilers.

The results of adding each of these features individually to a feature set containing both stemmed unigrams and unstemmed bigrams (Table 4). All of these features were individually able to raise baseline performance. Many of these features seem to be similar, as each individually improve accuracy by about the same amount. This is often reasonable, as the length of a television show is correlated with the genre of the television show.

Feature Set	Additional Features	Accuracy
All Features	20	0.67
Length	1, *cont*	0.64
Genre	13	0.64
Episodes	1, *cont*	0.64
First Aired	1, *cont*	0.64
Country	4	0.61
Baseline (unigram and bigram)	0	0.60

Table 4: Performance of classifiers compared to baseline performance of classifier with text features only. All of the new features improve performance, although length, genre, the number of episodes, and the date of first airing do best. For discrete features, the number of levels (and thus number of features) is given. For continuous features, only one feature is added. When all features, accuracy rises to 0.67.

However, by combining all of the features together into a single classifier, we improve accuracy to 0.67, higher than any of the features individually. This shows that while some of the information contained in the features is complementary, not all of the information is. For example, while you can tell a show is more likely to be a genre based on its run time, more recent works are more likely to have a higher concentration of spoilers.

ERROR ANALYSIS

Our classifier with additional features is better than a baseline model but still makes errors. This section analyzes those errors to explain what the system is doing, what features might improve performance further, and the intrinsic difficulty of detecting spoilers.

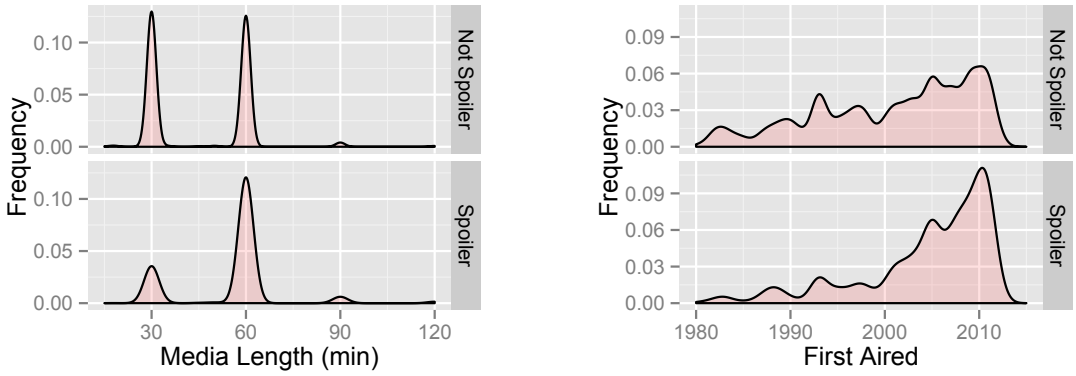
Distracted by Murder

> **Hell, pretty much the only villain there's no Foe Yay with is Annie's murderer Owen, which is ironic, as he was her fiancee.**

It is understandable why our system might label this sentence as a spoiler. The prediction is grounded in both Transitive text-based features ("murderer") and metadata features such as genres (Drama, Thriller), length, and country. However, this sentence is not a spoiler for two reasons: Annie's murder happens before the timeline of the work (in this case, the UK series *Being Human*), and this entry is focused on the relationship, not the actual murder.

The character Annie is a ghost, so her murder is not a surprise. It is part of the premise of the show. Because many recent shows have supernatural elements, it may be useful to distinguish "true death" from "fantasy death"; this could be accomplished by using kernel functions to allow implicit feature products.

Beyond this timeline issue, the focus is not on her death. The focus is on "Foe Yay", a trope meaning sexual—often homoerotic—tension between characters who are bitter enemies. The existence or lack of such tension would be unlikely



(a) Show Length

(b) Start Date

Figure 1: For show length (left), entries about longer shows (particularly hour-long) are more likely to contain spoilers than short format shows. Some of this can be explained by genre (sitcoms are more likely to be thirty minutes long); however, hour-long formats are more likely to involve spoilers even within a genre. For when the show first aired (right), more recent shows are more likely to be tagged with spoilers. There are multiple reasons this could be the case. Newer shows have pages that are more carefully curated; there has been an upsurge in epic, arc-centered dramas; and entries about older shows are more likely to be considered "common knowledge". (Media produced before 1980 are in the dataset but excluded in this plot because of low frequency.)

to constitute a spoiler. A classifier able to leverage trope information contained within sentence text might correctly predict this sentence.

We have thus far ignored trope-based information because tropes are an idiosyncratic property of TV Tropes; outside of this setting, it's unlikely that individuals would use the same trope-laden lexicon to describe works of fiction. While we might be able to improve performance *on this dataset*, it would unlikely generalize to social media in general.

Low-Transitivity Syntax

Sam's confrontation with his father is much darker than in the British version.

Our classifier incorrectly predicted this sentence was a spoiler because of lexical features (primarily "confrontation" and "father"; family relationships are often a part of spoilers). There was no metadata available for the show associated with the sentence. In the absence of that information we must rely on features derived from content. Our discussion of Transitivity pointed out that indicators of Transitive action (kinetic, completed, volitional, done with agency, real (not imaginary), and affecting an object) correlate with spoilers. Syntactic constructs such as comparatives (e.g., "is darker than") lack many of these properties. Additional features recognizing low-Transitivity constructions such as predicate adjectives could improve the model.

Figurative Language

Then his mother finally points out the elephant in the room.

This sentence is not classified as a spoiler by our system, in contradiction to human judgment. This sentence might appear to have been mistagged by a human annotator, in that referring to an "elephant in the room" appears to effectively conceal the plot rather than spoil it. However, in this episode, a ceramic statue of an elephant actually is in the room, and becomes a point of controversy. This is clear in the larger context, but cannot be understood from a single sentence. Our classifier operates at the sentence level and cannot predict this. Collocations, metaphorical language, and limited context are long-standing natural language processing problems (Fellbaum, 2007; Geyken and Boyd-Graber, 2003).

Annotator Error

Eleanor gets Simon to seduce Abigail in order to learn the King's secrets.

Our classifier classifies this sentence as a spoiler, though it is not annotated as a spoiler in TV Tropes. We assert that the classifier was correct, and the crowdsourcing efforts of TV Tropes, though extensive, have not yet thoroughly reviewed this statement. While crowdsourced ground truth data has many strengths, it may be less accurate than ground truth judgments made by trained annotators applying a standard, well-defined protocol. In TV Tropes, raters affirmatively annotate spoilers, but do not explicitly label non-spoilers. Thus the occasional unrated statement cannot be distinguished from a non-spoiler.

CONCLUSION

This work approached the problem of detecting "spoilers" in social media. We produced a new dataset of spoilers and use these data to contribute to the problem of spoiler detection, a fun task connected with more serious computational linguistics tasks such as detecting Transitivity and recovering temporal relations from text. This work also develops features than can effectively discriminate spoilers from non-spoilers in natural text.

The connection of spoilers to deeper linguistic tasks is important. Many of these related tasks have faltered because

of the lack of effective training data. By connecting these fundamental tasks to individuals' interactions on social media that they already self-annotate, we allow big data techniques to be harnessed for what had been restricted to the realm of "armchair linguists".

Our features improved over text-only baselines, but we expect that additional features would further improve the ability of automatic classifiers to detect spoilers. Previous research has shown that topic models (Blei et al, 2003) provide useful features, as do syntactic features (Guo and Ramakrishnan, 2010). While our genre features serve as a proxy for topics, other properties of a work—sub-genre and "weighting" works that straddle multiple genres—might be better captured by topic models. Moreover, detecting writers' sentiment (Pang and Lee, 2008) may also improve the algorithm's performance.

One limitation of our work is we assume that we know which work is being discussed. In a longer discussion or document, it is relatively easy to determine which work is being discussed; even without machine learning algorithms, matching proper nouns is a reasonable heuristic. However, a real-world system would need to determine how errors on this easier pre-processing step (identifying a work) affects the more complicated downstream task (determining whether a sentence is a spoiler).

We also hope to investigate the interface implications of integrating spoiler detection in a user's online experience. How aggressive should the algorithm be, and what are effective strategies for alerting users of potential spoilers and allowing them to continue to see the message once they understand the risk? Together with effective automatic classification techniques, these can make spoiler alerts more consistent and more effective as users obtain information from the web and social media. Further, if we consider both author and reader in a social media context to be engaged in communication, how best can we provide feedback to the author of the spoiler in order to constructively advance dialog? Our work, based in the seemingly lighthearted task of identifying spoilers, offers new mechanisms to enrich social media interactions.

Acknowledgments

We thank the anonymous reviewers for their insightful suggestions and Santiago Vanegas for his initial exploration of these data. Jordan Boyd-Graber is supported by NSF grant #1018625. Any opinions, conclusions, or recommendations are the authors' and not those of the sponsors.

References

Amazoncom (2013) Message boards etiquette. **http://pro.imdb.com/help/show_leaf?boardsetiquette**

Ben-Hur A, Weston J (2009) Biological Data Mining, Springer, chap A User's guide to Support Vector Machines

Bethard S, Kolomiyets O, Moens MF (2012) Annotating story timelines as temporal dependency structures. In: International Language Resources and Evaluation

Bird S, Klein E, Loper E (2009) Natural Language Processing with Python. O'Reilly Media

Blei DM, Ng A, Jordan M (2003) Latent Dirichlet allocation. Journal of Machine Learning Research 3:993–1022

Brickell J, Shmatikov V (2008) The cost of privacy: destruction of data-mining utility in anonymized data publishing. In: Knowledge Discovery and Data Mining

Chambers N, Jurafsky D (2011) Template-based information extraction without the templates. In: Proceedings of the Association for Computational Linguistics

Fellbaum C (2007) Idioms and collocations: corpus-based linguistic and lexicographic studies. Studies in corpus and discourse, Continuum

Geyken A, Boyd-Graber J (2003) Automatic classification of multi-word expressions in print dictionaries. Linguisticae Investigationes 26(2)

Golbeck J (2012) The twitter mute button: a web filtering challenge. In: International Conference on Human Factors in Computing Systems

Greene S, Resnik P (2009) More than words: Syntactic packaging and implicit sentiment. In: NAACL

Guo S, Ramakrishnan N (2010) Finding the storyteller: automatic spoiler tagging using linguistic cues. In: Proceedings of International Conference on Computational Linguistics

Guyon I, Weston J, Barnhill S, Vapnik V (2002) Gene selection for cancer classification using support vector machines. Machine Learning 46(1-3):389–422

Halevy A, Norvig P, Pereira F (2009) The unreasonable effectiveness of data. IEEE Intelligent Systems 24(2):8–12

Herring SC (2013) Handbook of Computational Linguistics and Natural Language Processing, Georgetown University Press, chap Discourse in Web 2.0: Familiar, reconfigured, and emergent

Hopper PJ, Thompson SA (1980) Transitivity in grammar and discourse. Language (56):251–299

Joachims T (1999) Making large-scale SVM learning practical. In: Advances in Kernel Methods - Support Vector Learning, Cambridge, MA, chap 11

Kiss T, Strunk J (2006) Unsupervised multilingual sentence boundary detection. Computational Linguistics 32(4):485–525

Lampel J, Bhalla A (2007) The role of status seeking in online communities: Giving the gift of experience. Journal of Computer-Mediated Communication 12(2):434–455

Leavitt JD, Christenfeld NJS (2011) Story Spoilers Don't Spoil Stories. Psychological science URL **http://dx.doi.org/10.1177/0956797611417007**

Lowenstein G (1994) The psychology of curiosity. Psychological Bulletin 116(1):75–98

Madnani N, Boyd-Graber J, Resnik P (2010) Measuring transitivity using untrained annotators. In: NAACL Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk

Maron ME (1961) Automatic indexing: An experiment inquiry. Journal of the Association for Computing Machinery 8(3):404–417

Pang B, Lee L (2004) A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: ACL

Pang B, Lee L (2008) Opinion Mining and Sentiment Analysis. Now Publishers Inc

Porter MF (1980) An algorithm for suffix stripping. Program: electronic library and information systems 14(3):130–137, DOI 10.1108/eb046814, URL **http://dx.doi.org/10.1108/eb046814**

Quinlan JR (1986) Induction of decision trees. Machine Learning 1(1):81–106

Resnik P, Lin J (2010) Handbook of Computational Linguistics and Natural Language Processing, Wiley Blackwell, chap Evaluation of NLP Systems

Sanders R (1987) Cognitive foundations of calculated speech: Controlling understandings in conversation and persuasion. SUNY Press

Schank RC, Abelson RP (1977) Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures. Lawrence Erlbaum

Sebastiani F, Ricerche CND (2002) Machine learning in automated text categorization. ACM Computing Surveys 34:1–47

Tracy K (2001) Discourse analysis in communication. The handbook of discourse analysis pp 725–749

Tsang AS, Yan D (2009) Reducing the spoiler effect in experiential consumption. Advances in Consumer Research 36:708–709

Wilson T, Centerbar D, Kermer D, Gilbert D (2005) The pleasures of uncertainty prolonging positive moods in ways people do not anticipate. Journal of Personality and Social Psychology 88(1):5–21