A Bootstrapping Approach to Identifying Relevant Tweets for Social TV

Ovidiu Dan*
Lehigh University
Bethlehem, PA, USA
ovd209@lehigh.edu

Junlan Feng
AT&T Labs Research
Florham Park, NJ, USA
junlan@research.att.com

Brian D. Davison
Lehigh University
Bethlehem, PA, USA
davison@cse.lehigh.edu

Abstract

Manufacturers of TV sets have recently started adding social media features to their products. Some of these products display microblogging messages relevant to the TV show which the user is currently watching. However, such systems suffer from low precision and recall when they use the title of the show to search for relevant messages. Titles of some popular shows such as *Lost* or *Survivor* are highly ambiguous, resulting in messages unrelated to the show. Thus, there is a need to develop filtering algorithms that can achieve both high precision and recall. Filtering microblogging messages for Social TV poses several challenges, including lack of training data, lack of proper grammar and capitalization, lack of context due to text sparsity, etc.

We describe a bootstrapping algorithm which uses a small manually labeled dataset, a large dataset of unlabeled messages, and some domain knowledge to derive a high precision classifier that can successfully filter microblogging messages which discuss television shows. The classifier is designed to generalize to TV shows which were not part of the training set. The algorithm achieves high precision on our two test datasets and successfully generalizes to unseen television shows. Furthermore, it compares favorably to a text classifier specifically trained on the television shows used for testing.

1 Introduction

We address the problem of filtering social media messages for use in Social TV applications. Some television sets and set-top boxes which integrate data from social networks can display messages about the TV show the user is currently watching. These messages are displayed next to the video or overlayed on top of the image. They are shown individually or in some cases as part of aggregate statistics on the TV show. Current Social TV applications search for these messages by issuing queries to social networks with the full title of the television program. Unfortunately, this naïve approach leads to low precision and recall.

Our main goal is to retrieve microblogging messages relevant to any given TV show with high precision. Our work makes several contributions. First, we propose an iterative bootstrapping approach to training a classifier using a limited amount of training data and a large dataset of unlabeled messages. The classifier is designed to generalize to TV shows that were not part of the training set. Second, based on our intuition and experiments, we develop features which can be used to filter messages for Social TV applications. Third, we evaluate our bootstrapped classifier on a combination of two manually labeled datasets. We show that our classifier achieves high precision, successfully generalizes to unseen television shows, and matches or surpasses the baseline which is trained on specific television shows.

Online social networks such as Twitter have attracted much interest from the academic community in the last few years (Kwak et al. 2010). Social TV projects have used audio, video, and text chat links to test interaction between users watching TV in separate rooms. More recently there has been work on combining these two fields by displaying messages from social networks in Social TV interfaces (Mitchell et al. 2010). Unfortunately such attempts used the naïve method of searching for the title of the TV show. While this paper is the first one to approach the task of filtering messages about TV shows, we found two papers that discuss the somewhat similar task of determining if tweets refer to a given company. In (Tellez et al. 2010) the authors used four term expansion approaches to improve the performance of K-Means clustering on short texts, with K set to two. Their approaches outperformed the baseline, which was the normal K-Means algorithm. (Yerva, Miklós, and Aberer 2010) attempted to build a generic classifier with the same goal. They achieved a precision of 71% for the positive class. To the best of our knowledge our work is the first to identify messages for Social TV.

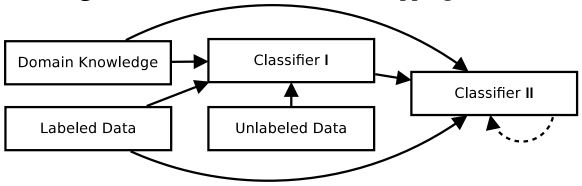
2 Bootstrapping Approach

Labeling training data for hundreds or thousands of television shows is a costly and time consuming process. Instead, we propose a bootstrapping method which uses a small set of labeled data, some domain knowledge, and a large unlabeled dataset to train a classifier which generalizes to hundreds of television programs. Figure 1 illustrates the iterative process of this approach.

First, we retrieve candidate messages from the social media website. This retrieval step is needed because most social media websites offer search APIs which require queries. We

*Part of this work was performed while the first author was visiting AT&T Labs Research.

Figure 1: Overview of our bootstrapping method



collect a list of TV show titles from IMDB[1] and TV.com[2]. For some shows these websites list several variations of the main title. We use each title in the list as a query to the search API provided by Twitter and retrieve candidate messages for each show.

Second, we train an initial binary classifier, which we refer to as *Classifier-I*, using a small dataset of manually labeled messages. The features used for training are described in Section 4. Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of n television shows and let k_i be the set of queries we use to search for candidate messages which might be relevant to show s_i, $\forall i = 1 \ldots n$. For the purpose of this paper we will define k_i to only contain the title variants of the show s_i which we collected in the first step. However, some of the features described in Section 5 could be used to extend this set, greatly improving overall recall. Let m be a message retrieved by using a query in k_i. Then, we train the classifier:

$$f(i,m) = \begin{cases} 1, & \text{if } m \text{ makes a reference to show } s_i \\ 0, & \text{if } m \text{ does not make a reference to show } s_i \end{cases}$$

Note that i is an input of f, along with the message m. Since we know m was retrieved by searching for k_i, we can use this context when we compute the values of the features. For a new unlabeled message, we compute a list of IDs of candidate TV shows by matching the text of the message with the keywords in k_i. We can test each of these candidate IDs against the new message by using the classifier.

Third, we train an improved classifier, *Classifier-II*. To achieve this, we use *Classifier-I* to assign labels to a large corpus of unlabeled messages. From this corpus of messages we derive more features which are combined with the ones from *Classifier-I* to train *Classifier-II*. Optionally, we can iterate this step using the second classifier to further increase the quality of the extra features. We will elaborate on the features used for *Classifier-II* in Section 5.

3 Datasets

We used three datasets for training, testing, and deriving new features for the classifiers. Dataset $D1$ was labeled using the Amazon Mechanical Turk Service and dataset $D2$ contains messages labeled by our team. Table 1 shows statistics about $D1$ and $D2$. Furthermore, a third dataset, DU, consists of 10 million unlabeled Twitter messages collected in October 2009 using the Streaming API provided by Twitter.

[1]http://www.imdb.com/

[2]http://www.tv.com/

Table 1: Summary of manually labeled datasets

	Show	**Yes**	**No**	**N/A**	**Total usable**
$D1$	Fringe	634	227	139	861
	Heroes	541	321	138	862
	Monk	317	589	94	906
$D2$	Survivor	325	175		500
	Cheaters	101	49		150
	Firefly	68	102		170
	House	80	20		100
	Weeds	97	65		162

4 Classifier-I

We developed features which capture the general characteristics of messages which discuss TV shows. Each of these features have one single value between 0 and 1 (normalized when needed).

Terms related to watching TV. We developed three features based on the observation that TV-related microblogging messages contain general terms commonly associated with watching TV. *tv_terms* and *network_terms* are short manually compiled lists of keywords, such as *watching, episode, hdtv, netflix,* and *cnn, bbc, pbs*. When classifying messages we check if they contain any of these terms. If they do, we set the corresponding feature to 1.

Some users post messages which contain the season and episode number of the TV show they are currently watching. Since Twitter messages are limited in length, this is often written in shorthand. For instance, *"S06E07"*, *"06x07"* and even *"6.7"* are common ways of referring to the sixth season and the seventh episode of a particular TV show. The feature *season_episode* is computed with the help of a limited set of regular expressions which can match such patterns.

General Positive Rules. We observed that many Twitter messages about TV shows follow certain language patterns. Table 2 shows such patterns. <*start*> means the start of the message and <*show_name*> is a placeholder for the real name of the show in the current context s_i. When a message contains such a rule, it is more likely to be related to television shows. We use these rules in the feature *rules_score*.

Table 2: Examples of general positive rules

<*start*> watching <*show_name*>
episode of <*show_name*>
<*show_name*> was awesome

We developed an automated way to extract such general rules and compute their probability of occurrence. We started from a manually compiled list of ten mostly unambiguous TV show titles, which contains titles such as *"Mythbusters"*, *"The Simpsons"*, *"Grey's Anatomy"*, etc. We searched for these titles in all 10 million messages from DU. For each message which contained one of these titles, the algorithm replaced the title of TV shows, hashtags, references to episodes, etc. with general placeholders, then computed the occurrence of trigrams around the keywords. The result is a set of general rules such as the ones shown in Table 2, along with their occurrences count in the messages which matched the titles. If we find one of these rules in an unlabeled message, we use the normalized popularity of the rule as the value of the feature *rules_score*.

Features related to show titles. Although many social media messages lack proper capitalization, when users do capitalize the titles of the shows this can be used as a feature. Consequently, our classifier has a feature called *title_case*, which has the value 1 *iff* the title of the show is capitalized.

Another feature which makes use of our list of titles is *titles_match*. Some messages contain more than one reference to titles of TV shows. If any of the titles mentioned in the message (apart from the title of the current context s_i) are unambiguous, we can set the value of this feature to 1. For the purpose of this feature we define *unambiguous title* to be a title which has zero or one hits when searching for it in WordNET (Fellbaum 1998). WordNET contains both single words and compound expressions.

Features based on domain knowledge crawled from online sources. One of our assumptions is that if a message contains names of actors, characters, or other keywords strongly related to the show s_i, the probability of $f(i, m)$ being 1 increases. To capture this intuition we developed three features: *cosine_characters*, *cosine_actors*, and *cosine_wiki*, which are based on data crawled from TV.com and Wikipedia. For each of the crawled shows, we collected the names of actors and their respective characters. We also crawled their corresponding Wikipedia page. Using the assumptions of the vector space model we compute the cosine similarity between a new message and the information we crawled about the show for each of the three features.

5 Classifier-II

We used *Classifier-I* to label the messages in DU. We then derived new features for *Classifier-II*.

Positive and negative rules. Features *pos_rules_score* and *neg_rules_score* are natural extensions of the feature *rules_score*. Whereas *rules_score* determined general positive rules, now that we have an initial classifier we can determine positive and negative rules for each show in S individually. For instance, for the show *House* we can now learn positive rules such as *episode of house*, as well as negative rules such as *in the house* or *the white house*.

Related users and hashtags. Using messages labeled by *Classifier-I*, we can determine commonly occurring hashtags and users which often talk about a particular show. We use messages labeled as "Yes" by *Classifier-I* to calculate the occurrences of users and hashtags for each individual show. These features, which we call *users_score* and *hashtags_score*, can be used to expand the set k_i for each show, thus improving the overall recall of the system. For unlabeled messages, the value of these features is given by the normalized popularity of the users or hashtags for the current show s_i.

Tweet volume. As users of social media websites often discuss a show around the time it airs, we develop the feature rush_period. We keep a running count of the number of times each show was mentioned in every 10 minute interval. If the number of mentions is higher than a threshold equal to twice the mean of the mentions of all previous 10 minute windows, we set the feature to 1. Otherwise, we set it to 0. An important advantage of this method is that it does not require information on TV listings or timezones.

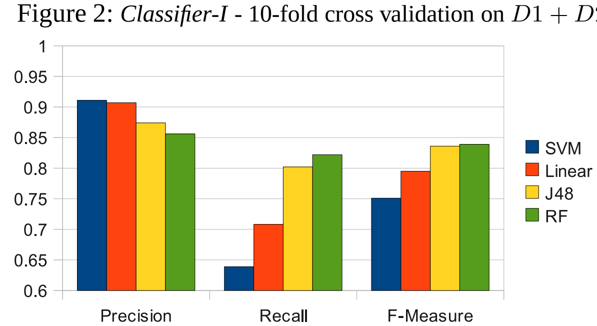
6 Evaluation

Unless otherwise stated, all experiments in this section were carried out on the $D1 + D2$ dataset using 10-fold cross validation, and all results show the performance of the *Yes* label.

Evaluation of Classifier-I

The results for *Classifier-I* are shown in Figure 2. The classifier labeled *SVM* is the Java implementation of *LIBSVM*, *Linear* corresponds to *LIBLINEAR*, *J48* is an implementation of the *C4.5* decision tree classifier, and *Rotation Forest (RF)*, is a classifier ensemble method. Please refer to (Hall et al. 2009) for more information. We ran all classifiers with their default settings. Figure 2 shows that we can achieve a precision of over 90% with both the *SVM* and *Linear*. In fact, all four classifiers achieve a precision over 85%. Unfortunately, the results suffer from poor and inconsistent recall. *SVM* and *Linear*, which achieve the best precision, have especially bad results. Overall, the best classifier for this experiment is Rotation Forest, which achieves an F-Measure of 83.9%.

Figure 2: *Classifier-I* - 10-fold cross validation on $D1 + D2$

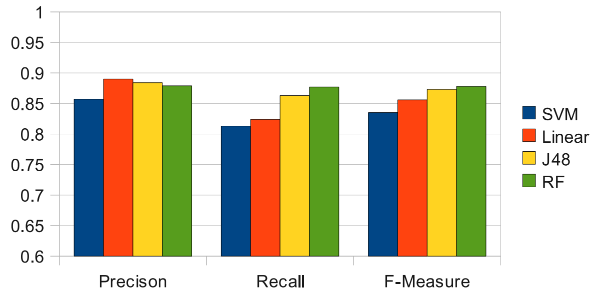


Evaluation of Classifier-II

10-fold Cross Validation. Next, we turn our attention to the results for *Classifier-II*, which are shown in Figure 3. We can easily see that the recall has improved significantly for all four classifiers. Again, the best performing classifier is Rotation Forest, with an F-Measure of 87.8%. Also, the Linear classifier yields the best precision of 89% while still maintaining a respectable F-Measure of 85.6%.

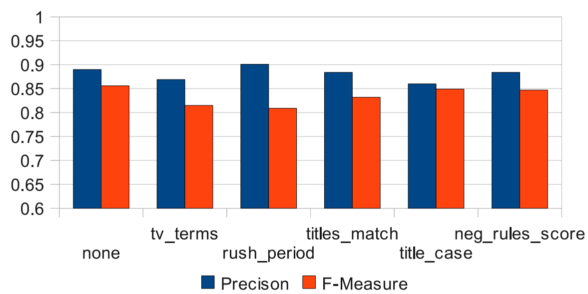
Leave one feature out. The purpose of this experiment is to observe what are the effects of leaving one of the features out when training *Classifier-II*. We run 14 experiments, one for each feature. Each time we remove one feature and run

Figure 3: *Classifier-II* - 10-fold cross validation on $D1 + D2$



the experiment with the remaining ones. The classifier we used was *Linear* (LIBLINEAR). Figure 4 shows the change in Precision and F-Measure. The bars labeled "none" show the results with all fourteen features and are our basis for comparison. Because of lack of space we only show the top five features which had the largest drop in F-Measure. The largest drop is only 3.6% absolute, which suggests there is no feature which vastly dominates the others, and that they are used in ensemble by the classifier.

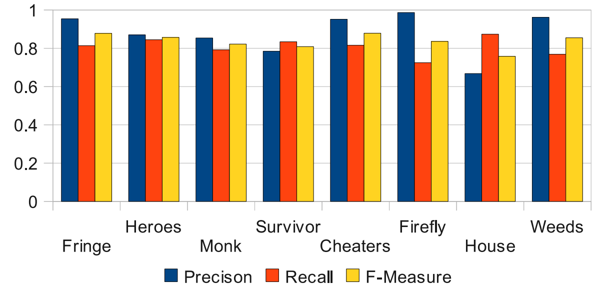
Figure 4: Precision and F-Measure when removing a feature - $D1 + D2$



Leave one show out. We argued that one major advantage of this classifier is that it generalizes to TV programs on which it has not been directly trained. To test this claim we ran eight experiments where we trained the classifier on seven of the shows and tested on the eighth. Figure 5 shows a summary of the results. As before, we used the *Linear* classifier. Except for the show *House*, all F-Measure values are above 80%, with a mean of 83.6%. The F-Measure computed on all shows for the Linear classifier is 85.6%. At an absolute difference of 2% we can conclude that our classifier successfully generalizes to new shows.

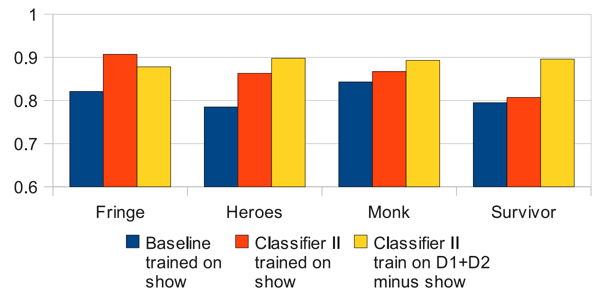
Comparison against a baseline Finally, we compare our classifier against a baseline built by training a text classifier on one show at a time. Our goal here is to show that our more general classifier can come close to or even surpass one specifically trained on a particular show. This was in fact an important argument in our motivation. The features of the baseline classifier are the terms of each message normalized by the maximum word occurrence in the message. Often in such types of messages each term has an occurrence of 1, as

Figure 5: Precision, Recall, and F-Measure when leaving one show out



they are limited to 140 characters. We will again use the *Linear* classifier to run the experiments. We use the four shows from $D1$ and $D2$ for which we have at least 500 samples. Figure 6 shows that our classifier achieves better precision than the baseline. F-Measure results, not shown here due to lack of space, are similar. In conclusion, we achieved our goal of being comparable to or surpassing the results of a baseline specifically trained on a particular show.

Figure 6: Comparison to baseline - Precision for 10-fold cross-validation



References

Fellbaum, C. 1998. *WordNet: An electronic lexical database*. The MIT press.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. 2009. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter* 11(1):10–18.

Kwak, H.; Lee, C.; Park, H.; and Moon, S. 2010. What is Twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, 591–600. ACM.

Mitchell, K.; Jones, A.; Ishmael, J.; and Race, N. 2010. Social TV: toward content navigation using social awareness. In *Proceedings of the 8th international interactive conference on Interactive TV&Video*, 283–292. ACM.

Tellez, F. P.; Pinto, D.; Cardiff, J.; and Rosso, P. 2010. On the difficulty of clustering company tweets. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, SMUC '10, 95–102. New York, NY, USA: ACM.

Yerva, S. R.; Miklós, Z.; and Aberer, K. 2010. It was easy, when apples and blackberries were only fruits. In *Third Web People Search Evaluation Forum (WePS-3), CLEF*, volume 2010.