Filtering microblogging messages 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 Davison
Lehigh University
Bethlehem, PA, USA
davison@cse.lehigh.edu

ABSTRACT

Social TV was named one of the ten most important emerging technologies in 2010 by the MIT Technology Review. Manufacturers of set-top boxes and televisions have recently started to integrate access to social networks into their products. Some of these systems allow users to read microblogging messages related to the TV program they are currently watching. However, such systems suffer from low precision and recall when they use the title of the show as keywords when retrieving messages, without any additional filtering.

We propose a bootstrapping approach to collecting microblogging messages related to a given TV program. We start with a small set of annotated data, in which, for a given show and a candidate message, we annotate the pair to be relevant or irrelevant. From this annotated data set, we train an initial classifier. The features are designed to capture the association between the TV program and the message. Using our initial classifier and a large dataset of unlabeled messages we derive broader features for a second classifier to further improve precision.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval—*Information Filtering*

General Terms

Theory, Algorithms, Experimentation

Keywords

Social TV, Twitter, microblogging, filtering, classification

1. INTRODUCTION

This paper tackles the problem of filtering social media messages for use in Social TV applications. The users of such applications, which run on TV sets or set-top boxes, can choose to receive microblogging messages relevant to a given TV program. The messages are displayed either alongside the video or overlayed on top of the image. Current Social TV applications search for these messages by issuing queries to social networks with the full title of the TV program. This naive approach can lead to low precision and recall.

[*]Parts of this work was performed while the first author was visiting AT&T Labs - Research.

The popular TV show *House* is an example that results in low precision. Searching for the title of the show often yields results unrelated to the show. Table 1 shows such examples. The word *house* has multiple senses depending on the context, including *White House, House of Representatives, building, home, etc.* In some cases the query is part of the title of another show, as can be seen in the last example. Another problem is low recall. Continuing with our example for the show *House*, there are many messages which do not mention the title of the show but make references to users, hashtags, or even actors and characters related to the show. The problem of low recall is more severe for shows with long titles.

Our task is to retrieve microblogging messages relevant to a given TV show with high precision. Filtering messages from microblogging websites poses several challenges, including:

- Microblogging messages are short and often lack context. For instance, Twitter messages (tweets) are limited to 140 characters and often contain abbreviated expressions such as hashtags and short URLs.
- Many social media messages lack proper grammatical structure. Also, users of social networks pay little attention to capitalization and punctuation. This makes it difficult to apply natural language processing technologies to parse the text.
- Many social media websites offer access to their content through search APIs, but most have rate limits. In order to filter messages we first need to collect them by issuing queries to these services. For each show we require a set of queries which provides the best tradeoff between the need to cover as many messages about the show as possible, and the need to respect the API rate limits imposed by the social network. Such queries could include the title of the show and other related strings such as hashtags and usernames related to the show. Determining which keywords best describe a TV show can be a challenge.
- In the last decade alone, television networks have aired more than a thousand new TV shows. Obtaining training data for every show would be prohibitively expensive. Furthermore, new shows are aired every six months.

We propose a bootstrapping approach to automatically classifying a candidate Twitter message as relevant or irrelevant to a given show. Our robust filtering method can be used for several applications, including displaying messages related to particular TV shows, measuring the popularity of TV programs, displaying accounts and hashtags related to a show, and further mining such as sentiment analysis and other aggregate statistics.

The rest of the paper is organized as follows: Section 2 gives an overview of our bootstrapping approach, Section 3 discusses the

Table 1: Example messages for the ambiguous query *house*

driving back to my **house, i really hope @VampireRoland likes his suit, i love this dress i got**
@blogcritics White **House**, Fox News Feud Heats Up Over the Weekend http://bit.ly/mi5tg
Someone may be in my **house**... And im a little scared.
Election 2010 **House** of Representatives 33rd District http://bit.ly/cv0l48
Watching Clean **House**

two datasets we use for training and testing, Section 4 discusses the features for the Initial Classifier, Section 5 describes the features of the Improved Classifier, then Section 6 shows a detailed evaluation of the two classifiers and a baseline. We conclude with Previous Work and References.

2. OVERVIEW OF OUR BOOTSTRAPPING APPROACH

Hundreds of new television shows are created each year in the United States alone. Creating training data for each show individually would be costly and inefficient. Instead, we propose a bootstrapping method which is built upon 1) a small set of labeled data, 2) a large unlabeled dataset, and 3) some domain knowledge, to form a classifier that can generalize to an arbitrary number of TV shows.

Our approach starts from a list of TV show titles which can be obtained by crawling popular websites such as IMDB[1] or 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. Later in the bootstrapping process we can automatically expand the list of keywords for each show by adding relevant hashtags, user accounts or other keywords which the algorithm determines are related to the show.

First, we train a binary classifier using a small dataset of manually labeled messages (dataset DT). The input of the classifier is the new message which needs to be classified, along with the unique ID of a TV show. It outputs 1 if the message is relevant to the television show, or 0 otherwise. For a new message we can get a list of possible TV shows by matching the text of the message with the keywords we use for each show in the first step. We can test each of these possible IDs against the new message by using the classifier. The features used by the classifier are described in Section 4.

Second, we run the Initial Classifier on a large corpus of unlabeled Twitter messages (dataset DL). These newly labeled messages are then used to derive more features. The new features are combined with the features of the Initial Classifier to train an Improved Classifier. This step can be iterated several times to improve the quality of the features. The features of this classifier are described in Section 5.

3. DATASETS

DT Dataset We used workers from Amazon Turk [11] to label the training dataset. We picked three TV shows with ambiguous names: $Fringe$, $Heroes$, and $Monk$. For each of these shows we randomly sampled 1000 messages which contained the title of the show. The messages were sampled from the DL dataset described below. The workers were asked to assign one of three labels to each message: *"Yes, the message is relevant to the show"*, *"No, it is not relevant"*, and *"Not sure / Foreign language"*. The results of the labeling process are summarized in Table 2. After discarding messages which received the third label, we are left with 2,629 labeled messages.

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

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

Table 2: Summary of the training/testing dataset

	Show	Yes	No	N/A	Total usable
DL	Fringe	634	227	139	861
	Heroes	541	321	138	862
	Monk	317	589	94	906

DL Dataset The bootstrapping method described in Section 2 makes use of a large amount of unlabeled data to improve features used by the Improved Classifier. We will refer to this large corpus as DL. The dataset was collected in October 2009 using the Streaming API provided by Twitter. This is a push-style API with different levels of access which constantly delivers a percentage of Twitter messages over a permanent TCP connection. We were granted the *Gardenhose* level access which the company describes as providing a "statistically significant sample" of the messages. We collected over 10 million messages, roughly equivalent to 340,000 messages per day. Apart from its textual content, each message has metadata attached to it, which includes the author and the time when the message was originally posted.

4. INITIAL CLASSIFIER

We developed features which capture the general characteristics of messages which discuss television shows.

4.1 Terms related to TV watching

While studying TV-related microblogging messages we noticed that some of them contain general terms commonly associated with watching TV. Table 3 contains a few examples of such messages. Starting from this observation we developed three features: *tv_terms*, *network_terms*, and *season_episode*.

Table 3: Messages containing TV-related terms

True Blood 3rd **season finale**, here I come.
If **CNN**, **C-SPAN** & **Fox News** will be at Stewart Sanity/Fear rally, why not NPR? Come on, lighten up.
S06E07 - Teamwork (**watching** House via @gomiso)

tv_terms and *network_terms* are two short lists of keywords compiled manually. tv_terms are general terms such as *watching, episode, hdtv, netflix,* etc. The *network_terms* list contains names of television networks such as *cnn, bbc, pbs,* etc.

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.

These three features described above are binary with values of 0 or 1. For example, if a message matches one of the patterns in *season_episode*, this feature will have the value 1. Otherwise, it will have the value 0. Also, throughout this paper we will assume that all features are normalized when needed.

4.2 General Positive Rules

The motivation behind the *rules_score* feature is the fact that many messages which discuss TV shows follow certain patterns. Table 4 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. When a message contains such a rule, it is more likely to be related to TV shows.

Table 4: 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 start from a manually compiled list of ten unambiguous TV show titles. It contains titles such as *"Mythbusters"*, *"The Simpsons"*, *"Grey's Anatomy"*, etc. We searched for these titles in all 10 million messages from DL. 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 Section 4. Next, we computed the occurrences of these rules in dataset DL to determine which ones have a higher chance of occurring. Using these rules we can then give a value between 0 and 1 for the feature *rules_score* to each new message.

4.3 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 is set to 1 if the title of the show is capitalized, otherwise it has the value 0. We consider multi-word titles to be capitalized if at least the first letter of the first word 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. Some examples are listed in Table 5. 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 [1].

4.4 Features based on domain knowledge crawled from online sources

One of our assumptions is that messages relevant to a show often contain names of actors, characters, or other keywords strongly related to the show. To capture this intuition we developed three features: *cosine_characters*, *cosine_actors*, and *cosine_wiki*, which

Table 5: Examples of messages which mention the titles of several shows

If I'm sick call **HOUSE**, if I'm dead call **CSI**
grey's anatomy & supernatural
Lets see - **Jericho**, **Heroes**, and now **Caprica**. Don't tell me to watch a series you like. If I like it, it'll get the axe for sure :-/ #fb

are based on data crawled from TV.com and Wikipedia. For each of the crawled shows, we collected the names of actors which play in the show, and the name of 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. IMPROVED CLASSIFIER

We applied our Initial Classifier to automatically label the messages in DL and derive new features. Two such 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 separately. 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*.

Using messages labeled by Classifier #1, we can determine commonly occurring hashtags and users which often talk about a particular show. We refer to these features as *users_score* and *hashtags_score* respectively. Furthermore, these features can also help us expand the set of queries for each show, thus improving the recall by searching for hashtags and users related to the show, in addition to the title. While we have not tested this hypothesis here, we plan to do so in future work.

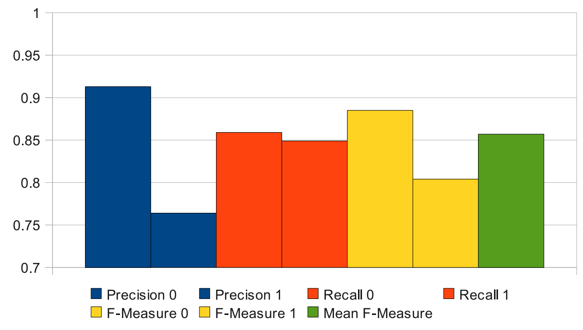
Lastly, having a large number of messages allows us to create one more feature, *rush_period*. This feature is based on the observation that users of social media websites often discuss about a show during the time it is on air. We keep a running count of the number of times each show was mentioned in every 10 minute interval. When classifying a new message we check how many mentions of the show there were in the previous window of 10 minutes. 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.

6. EVALUATION

6.1 Evaluation of Initial Classifier

We conducted a 10-fold cross validation of the Initial Classifier on the DT dataset. We ran our experiments with *Rotation Forest (RF)* [10], which is a classifier ensemble method. Among the classifiers we tested, RF achieved the best overall precision and recall. It uses Principal Component Analysis to achieve greater accuracy and diversity by rotating the feature axes. The underlying classifier we used was *J48*, a variant of the *C4.5* [9] available in the Weka machine earning software [2]. To save space, we will refer to labels *"Yes"* and *"No"* as 1 and 0 respectively. The results are shown in Figure 1. Along the X axis we displayed the precision, recall and F-Measure of the two labels. Note that in this case by recall we mean the recall of the RF classifier we are using, not the recall of the overall system. We also plotted the combined F-Measure of the two labels. The precision and F-measure of label *"Yes"* are 0.76 and 0.8, respectively.

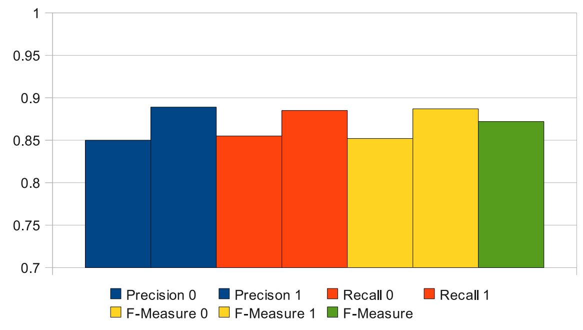
Figure 1: Initial Classifier - 10 fold cross validation on DT



6.2 Evaluation of Improved Classifier

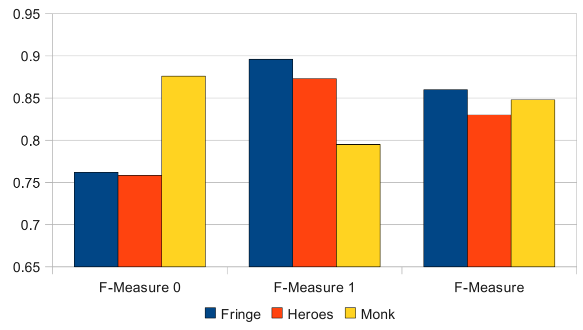
Next, we evaluated the Improved Classifier. We first ran the same evaluation as for the Initial Classifier. Figure 2 shows the results of the 10-fold cross validation on the DT dataset. We can easily see that both precision and recall have improved significantly for label *Yes*. Precision has increased from 0.76 to 0.89, while the F-measure has increased from 0.80 to 0.89.

Figure 2: Improved Classifier - 10 fold cross validation on DT



Previously we argued that one major advantage of this classifier is that it generalizes to television programs it has not been directly trained on. To test this claim, we ran an experiment by training on two of the shows, and testing on the third one, The results are in Figure 3. Averaging the result over the three possible combinations yields a precision of 0.84 and an F-measure of 0.85 for label *Yes*.

Figure 3: Improved Classifier - leave one show out on DT



7. PREVIOUS WORK

Social networks in general and microblogging websites such as Twitter in particular have attracted much interest from the academic community in the last few years [4, 5, 6]. Social TV projects have used audio [8], video [3], and text chat [12] 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 [7]. Unfortunately such attemps uses the naive method of simply searching for the title of the TV show. To the best of our knowledge our work is the first to filter and display only the messages relevant to the show currently playing on the screen.

8. SUMMARY

We presented a bootstrapping approach for training a classifier which can filter messages for given TV shows. First we trained an initial classifer from a small set of annotated data and domain knowledge. Second, we used the obtained initial classifier to label a large dataset of unlabeled data. Third, we automatically derived a broader feature set from the large data set which was automatically annotated by the Initial Classifer. These expanded features are used to construct the second classifier. Experiments showed that the second classifier achieved significantly higher performance, and it could successfully label messages about television programs which were not in the original training data.

9. REFERENCES

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

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

[3] C. Huijnen, W. IJsselsteijn, P. Markopoulos, and B. de Ruyter. Social presence and group attraction: exploring the effects of awareness systems in the home. *Cognition, Technology & Work*, 6(1):41–44, 2004.

[4] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65. ACM, 2007.

[5] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks*, pages 19–24. ACM, 2008.

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

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

[8] L. Oehlberg, N. Ducheneaut, J. Thornton, R. Moore, and E. Nickell. Social TV: Designing for distributed, sociable television viewing. In *Proc. EuroITV*, volume 2006, pages 25–26, 2006.

[9] J. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.

[10] J. Rodriguez, L. Kuncheva, and C. Alonso. Rotation forest: A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1619–1630, 2006.

[11] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast—but is it good? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics, 2008.

[12] J. Weisz, S. Kiesler, H. Zhang, Y. Ren, R. Kraut, and J. Konstan. Watching together: integrating text chat with video. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, page 886. ACM, 2007.