TV Program Detection in Tweets

Paolo Cremonesi
Politecnico di Milano, DEI
P.zza Leonardo da Vinci, 32
Milan, Italy
paolo.cremonesi@polimi.it

Roberto Pagano
Politecnico di Milano, DEI
P.zza Leonardo da Vinci, 32
Milan, Italy
pagano@elet.polimi.it

Stefano Pasquali
Politecnico di Milano, DEI
P.zza Leonardo da Vinci, 32
Milan, Italy
stefano.pasquali@mail.polimi.it

Roberto Turrin
Moviri S.p.A., R&D
Via Schiaffino, 11
Milan, Italy
roberto.turrin@moviri.com

ABSTRACT

Posting comments on social networks using second screen devices (e.g., tablets) while watching TV is becoming very common. The simplicity of microblogs makes Twitter among the preferred social services used by the TV audience to share messages about TV shows and movies. Thus, users' comments about TV shows are considered a valuable indicator of the TV audience preferences. However, eliciting preferences from a tweet requires to understand if the tweet refers to a specific TV program, a task particularly challenging due to the nature of tweets - e.g., the limited length and the massive use of slangs and abbreviations.

In this paper, we present a solution to identify whether a tweet posted by a user refers to one among a set of known TV programs. In such process, referred to as *item detection*, we assume the system is given a set of items (e.g., the TV shows or movies) together with some features (e.g., the title of the TV show). We tested the solution on a dataset composed by approximately 32000 tweets, where the optimal configuration reached a precision of about 92% with a recall equals to about 65%.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language parsing and understanding, Text analysis*; I.5.2 [**Computing Methodologies**]: Pattern Recognition—*Classifier design and evaluation*

General Terms

Algorithms

EuroITV'13, June 24–26, 2013, Como, Italy.

Keywords

Item detection, Twitter, SVM, ROC, implicit elicitation, recommender systems

1. INTRODUCTION

Sharing TV viewing experience through a second screen is getting very common [10, 16, 17]. An increasing number of users watching television desire to be part of a more social experience, either reading other viewers' messages or sharing their own opinions, e.g., commenting an episode of the TV show currently on air. Simplicity makes microblogs among the preferred social services to post short messages while watching TV[8], so that Twitter[1] - the most known and used microblog - has been recently mentioned as a possible measure of television audience [11, 19] and as a tool for preference elicitation in recommender systems [9].

In order to elicit user preferences from a microblog comment, we need to recognize if the user is writing about specific TV shows or movies. However, automatically detecting the subject of tweets - i.e. comments posted on Twitter - is a challenging task. In fact, the 140-character limit of tweets strongly affects the way people write on Twitter. Minimal contextualization, use of slang, grammatical exceptions, acronyms, abbreviations, and tiny URLs are characteristics common in tweets, making them very noisy and difficult to be analyzed. As an example, the tweet

> *Anyone going to @cstn this year? @theperfectfoil will be presenting BLJ in the Imaginarium tent on Thurs evening, July 5th*

refers to a talk that would be given by Steve Taylor ('The Perfect Foil') about the movie *Blue Like Jazz* ('BLJ') during the Cornerstone music festival in Illinois ('cstn') on Thursday ('Thurs').

In this work we propose a solution to analyze the content of tweets and to identify whether they refer to one (or more) known items - e.g., TV programs or movies. Given a set of known items (e.g., a Video on Demand catalog) and the tweet of a user, the proposed solution tries to identify if the user is commenting about some of the items. In the following we will refer to this process as *item detection.*

Item detection is a variant of topic detection, a text classification task. Topic detection tries to assign a text to one

[1] `http://twitter.com`

or more known categories; similarly, item detection tries to assign a tweet to one or more known items. However, item detection differs from standard topic detection mainly because of:

(i) **The number of classes**. Classes correspond to categories - e.g., the genres of movies - in topic detection and to items - e.g., the movies of a Video on Demand provider - in item detection. Thus, while the number of categories in topic detection is small, the number of items in typical TV applications can exceed some thousands elements.

(ii) **The variability of classes**. In typical item detection problems, items are constantly added to/removed from the catalog. This prevents the use of traditional classifiers, used in topic detection tasks, that are designed to work when the set of possible classes is defined a priori and static. In fact, such classification algorithms must be initially trained with a set of classified samples (e.g., tweets) representing all possible classes. Once trained, the classifiers can be used to predict the classes of unclassified tweets. Unfortunately, traditional classifiers require to be re-trained when some classes (i.e., items) are added or removed, being not suitable for applications where classes frequently change.

In order to overcome these two problems we opted to use a battery of one-class supervised classifiers [7]. A one-class classifier is a classification algorithm that is trained using only the objects of a given class c; once trained, it tries to distinguish class-c objects from all other possible objects. In our application domain, each one-class classifier is trained using only tweets referring to a single item. We implemented a battery of classifiers composed by a one-class classifier for each item in the catalog. When a new item (i.e., class) is added to the catalog, we simply need to train an additional one-class classifier, while no updates are required for the remaining one-class classifiers previously trained. Given the tweet of a user, we detect the items it refers to by interrogating all the one-class classifiers forming the battery; hence, the tweet is assigned to the items related to the classifiers that positively classify the tweet.

We tested the classification performance of our solution on a dataset composed by approximately 32000 tweets on 40 movies, reaching a precision of 92% and a recall of 65%.

Finally, we implemented a set of additional experiments in order to analyze the impact on the performance of (i) the number of tweets per item, (ii) the number of items in the dataset, and (iii) the presence of out-of-scope tweets, i.e., tweets not related to any item in the catalog or even not related to movies at all. The results demonstrate a good robustness of the item detection system with respect to all the considered variables.

The rest of the paper is organized as follows. In Section 2 we discuss related and supporting work. In Section 3 we formalize the problem and design the architecture of the item detection problem, together with a description of the three stages of the developed system. Section 4 describes methodology, metrics, and dataset used. In Section 5 we discuss more in detail about the third stage of our system, analyzing the main parameters that it requires. The main findings of our research are discussed in Section 6, where we also explore the hypothesis that a tweet refers to only one item. Finally, Section 7 draws some conclusions and presents some directions for future work.

2. RELATED WORK

Item detection shares several characteristics with topic detection, a traditional classification task that assigns a text document to one or more topics on the basis of its content. The analysis of the topic of a text is usually performed by means of statistical and mathematical models, assuming that there is a strong correlation between the terms (and their frequencies) used within the text and the arguments under discussion.

Blei et al. [3] describe *Latent Dirichlet Allocation* (LDA), an unsupervised approach based on hierarchical Bayesian models and the Expectation Maximization algorithm (EM) to classify text documents into predefined topics. Following this work, several other approaches based on LDA (e.g., [1, 2]) have been proposed. Makkonen et al. [18] describe *Topic Detection and Tracking* (TDT), a technique that analyzes a stream of news articles, taking into consideration its temporal nature, to extract topics or specific events. Griffiths et al. [13] use short-range syntactic correlations and long-range semantic dependencies in a Monte Carlo Markov Chain Model to classify documents into topics.

Support Vector Machines (SVM) have shown to yield good generalization performance on several classification problems, including text categorization [14, 15, 26, 27].

In the case of short texts, topic detection results more complex. In a very short text like a microblog post, in which the user message must be compacted within few sentences, finding statistically-relevant correlations among words is uncommon. With reference to Twitter, a tweet typically contains retweets, mentions, and hashtags. Retweets - abbreviated as 'RT' - are used to cite a tweet of an other user; mentions - defined by the prefix '@' - are used to address a tweet to a specific Twitter account; hashtags are free labels used to mark tweets.

Ramage et al. [22] try to identify the topics of a set of tweets using Labeled-LDA, a supervised learning model which extends LDA. Labeled-LDA has been applied to Twitter comments, where hashtags, emoticons, and generic social signals (e.g., the presence of questions or mentions) have been used as labels. Cataldi et al. [4] detect the emerging topics within a flow of short messages posted by the community of users. They take into account not only the term frequency but also the social network relationships (based on the well-known Page Rank algorithm). Finally, they generate a topic graph that connects emerging terms with semantically-related words, in order to form sets of emerging topics.

Part-of-speech (PoS) tagging [5] has been applied to microblogs by Gimpel et al. in [12], leading to an accuracy of 90% on about 1800 tweets manually annotated.

Wakamiya et al. [25] analyze Twitter messages to evaluate TV audience. Tweets are mapped onto TV shows - assuming that each tweet can be at most mapped onto a TV program - on the basis of the data available from the EPG (Electronic Programming Guide). The matching criteria takes into consideration word-based similarity in the Japanese language between tweet terms and the TV program title. Our work extends the work of Wakamiya et al. in the following directions:

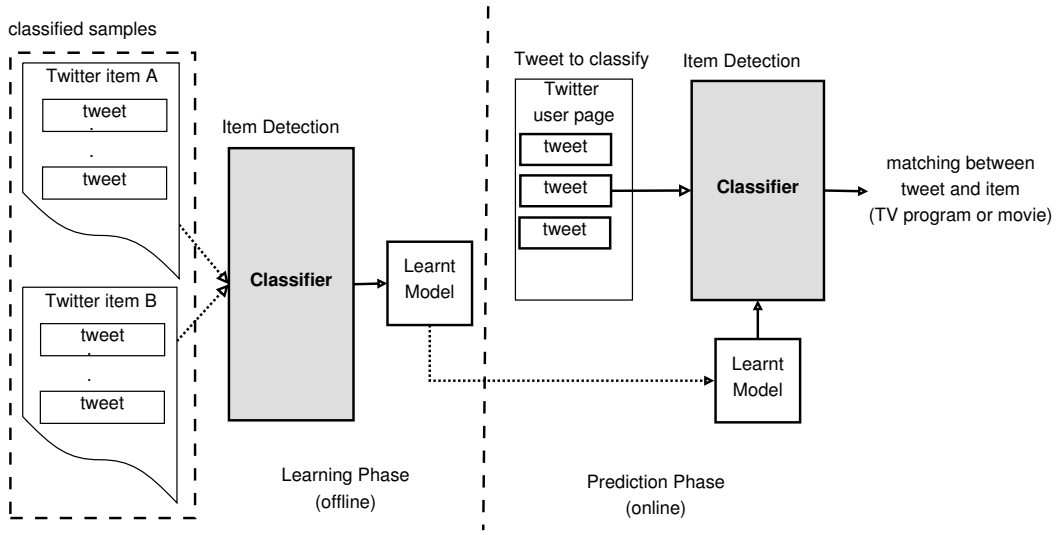

Figure 1: Item detection framework: learning and classification phases

- language: our classifier has been designed and tested to work with English tweets
- scalability: the structure of the system described by Wakamiya et al. does not allow to run the classifier in real-time applications; our classifier is intrinsically parallel and scales linearly with the number of available cores
- multiple classes: our classifier can detect if a tweet refers to more than one item
- our classifier is able to directly classify tweets that do not contain any word from item title or description

3. ITEM DETECTION

We define the *item detection* problem as a classification problem whose purpose is to detect whether a tweet refers to one of the items of a given catalog.

Classification systems are machine learning algorithms capable of assigning an item to a class. In our settings, the classes are the items in catalog, i.e. movies or TV shows. Typically, the classification task can be decomposed into two parts: *learning* and *prediction*. During the *learning* phase, as detailed on the left-hand side of Figure 1, the classifier takes a set of classified samples (i.e., tweets whose class is known) as input and computes a classification model - e.g., a set of parameters - optimized to accurately classify the input samples. Learning is usually performed off-line by a batch process. Once the classification model has been computed, the classifier can predict the class of any unclassified tweet, as shown on the right-hand side of Figure 1. Prediction can be usually performed on-line by a real-time process.

Detecting the item discussed in a tweet can be difficult, even for human. As an example, we reported in Table 1 three different tweets. Tweet 1 is very easy to classify and it clearly refers to the movie 'Harry Potter'. However, there can be ambiguous tweets, as tweet 2. In this case the tweet is comparing two movies: a correct classifier should assign it to two different classes. Some tweets, as tweet 3, require additional knowledge, not included in the message. Tweet 3 refers to the movie 'Blue Like Jazz' being Steve Taylor the movie's director.

We can note from the previous examples that a tweet can refer to one or more known items and can express either a positive, negative, or neutral opinion (the polarity). In this work we will neglect the tweet polarity and we will focus on the item detection task.

We designed the item detection system as a battery of one-class classifiers. A one-class classifier is a particular classifier that is trained with samples (tweets) belonging to a single class. The classifier, once trained, will be able to recognize whether an unknown tweet belongs or not to that class. Thus, the output of a one-class classifier can be either positive (the tweet is 'classified') or negative (the tweet is 'unclassified'). The choice of using a battery of one-class classifiers offers different advantages over a single multi-class classifier [24], such as:

1. **Scalability**. This solution easily scales with the number of items (i.e., TV programs or movies). In fact, since each classifier is independent from the others, the system can be straightforwardly parallelized in separated processes running on multiple cores.
2. **Incremental training**. When new items are added to the catalog, we simply need to train a new one-class classifier, with no impact on the rest of the system. Similarly, the removal of an item simply implies excluding the associated classifier.
3. **Multiple item classification**. Each classifier returns its own decision value, allowing a tweet to be assigned to multiple classes. For example, tweet 2 (Table 1) should be classified by two one-class classifiers, associated to the movies 'Twilight Breaking Dawn' and 'The Amazing Spiderman'.

Let $\mathcal{I}$ be the set of items available in the catalog and $|\mathcal{I}|$ its cardinality. Given a tweet t, the one-class classifier associated to item $i \in \mathcal{I}$ estimates the score $c_i \in \mathbb{R}$, which represents the degree of membership of such tweet to item i (i.e., how much the tweet refers to the item). The whole

ID	Tweet content	Item	Note
1	*I must be the one to kill #HarryPotter.*	Harry Potter	trivial
2	*#BreakingDawn Part 2 trailer ahead of The Amazing Spiderman! cheering: from the (female) crowd!*	Twilight: Breaking Dawn The Amazing Spiderman	ambiguous
3	*Special Q&A with Director Steve Taylor tomorrow night following the 7:40 show at Regal Cinema 8!*	Blue Like Jazz	difficult

Table 1: Sample tweets.

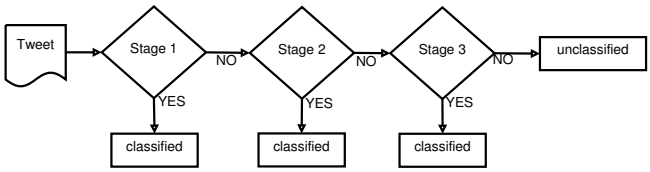

Figure 2: Architecture of a one-class classifier.

system is composed by a battery of $|\mathcal{I}|$ one-class classifiers. Its classification function $c(t)$ can be written as:

$$c(t) : t \rightarrow \mathbb{R} \times |\mathcal{I}| \quad (1)$$

Given the classification function (1), we can detect which items are commented in a certain tweet t.

We have decided to implement each one-class classifier as a pipeline composed by three stages, as detailed in Figure 2. An unknown tweet is initially processed by stage 1, that can either assign the tweet to its associated class or it can discard the tweet. Only tweets not classified at stage 1 are processed by stage 2. Similarly, only tweets not classified by stage 2 are processed by the last stage. Tweets not even classified by stage 3 remain not classified by this specific classifier. We used two keyword matching algorithms in stages 1 and 2: these algorithms are based on regular expressions. As for stage 3, we implemented a more advanced approach based on the SVM algorithm.

In the next sections we will describe the three stages in detail. Without loss of generality, we will assume the catalog of items is formed by movies, each one described by the movie title.

3.1 Stage 1: Exact Match

This stage deploys a simple keyword matching on the basis of the movie title, denoted as *exact match*. Given the item-i classifier, this algorithm classifies the tweet as belonging to class i if the text contains exactly the title of movie i.

3.2 Stage 2: Free match

Stage 2 of the one-class classifiers is composed of a keyword matching algorithm, denoted as *free match*. Differently from the keyword matching implemented in stage 1, stage 2 (i) searches for single words of the title within the tweet and (ii) searches for words of the title contained in other words in the tweet.

For instance, this stage is able to find a matching between the tweet

> *the future is back to the present in the past*

and the movie 'Back to the Future'. Still, this approach allows us to match the movie title also with hashtags and mentions. For instance, the tweet

> *Happy Birthday Elizabeth Reaser! #BreakingDawn*

is not recognized by stage 1, but it is associated by stage 2 to the movie 'Breaking Dawn' because it contains the hashtag '#BreakingDawn'. Similarly, the tweet

> *@abduction Just got home and it was awesome!!!!*

is matched by stage 2 because it contains the mention '@abduction', but not by stage 1. Moreover stage 2 is able to detect some cases of misspelling, as in the tweet

> *@OnlyHumour Sara had a life in the city, loopers are obviously not a secret in the club scene.*

in which the movie title 'Looper' is found in the word 'loopers'.

Stage 2 computes a matching score as the percentage of words in the item title that have been found in the tweet. The tweet is classified by stage 2 if the matching score is above a fixed threshold. With a low threshold (approaching 0%) we accept also tweets only partially matching with the terms in the movie title; contrarily, with a high threshold (approaching 100%) we accept only tweets strongly matching with the terms in movie titles.

3.3 Stage 3: SVM-based

Stage 1 and 2 search for the title of the related movie within the tweet. Thus, tweets reaching stage 3 are those not containing the title of the movie. For this reason, the classification task tackled by stage 3 is very difficult and requires a classification solution more advanced than the trivial keyword matching implemented in the previous stages.

We opted to implement in stage 3 a classification algorithm based on SVMs. In particular, we used the one-class SVMs derived by Cortes and Vapnik in [7]. In the following we will refer to this approach simply as SVM.

Differently from the previous two stages, the SVM algorithm requires also a learning phase to tune its parameters for a specific class. Learning is performed using a 5-fold cross validation [7] on a set of classified tweets. The output of the SVM algorithm is a real number, referred to as *decision value*. Only tweets with a decision value over a fixed threshold are classified. Training the SVM requires representing the tweet content: the following section describes how to represent tweet content as vectors of features.

3.3.1 Feature Extraction

According to most text processing works [21], we represented tweets as vectors of features. In such feature vector space, similar items - i.e., items that share common features - are closer than dissimilar items.

Before proceeding with feature extraction some text preprocessing must be performed because tweets are free text

and, partially due to the 140-character limit, they often include slangs, abbreviations, and acronyms. Furthermore, flexed forms of verbs and conjugations of adjectives and nouns should be normalized in order to be attributed to the same feature.

We implemented common text processing tasks such as tokenization, stop word removal, expansion of contracted forms, punctuation cleaning, and term stemming [21]. To better understand the next sections, we detail here two of the techniques we have implemented: tokenization and stop word removal. Tokenization splits text into tokens, i.e., individual entities such as terms. In addition, each language has words used much more than the others (e.g., articles). Such words - denoted as stop words - should be removed since they do not add any information useful to classify items, but they rather add noise.

Once the text has been preprocessed we can build a vector of features. Each possible value of a feature corresponds to a dimension of the vector. Each dimension has a numeric weight which is related to its importance in representing the tweet content. We extracted the following four types of features:

Unigrams correspond to the single terms. Each distinct unigram is represented by a dimension in the feature space. Unigrams formed by stop words are discarded.

Bigrams are pairs of adjacent terms. Each unique bigram corresponds to a dimension in the feature space. Note that only bigrams formed by two stop words have been discarded. For instance, this allows to extract bigrams such as 'The Matrix' ('the' is a stop word) while discarding 'I will' (both 'I' and 'will' are stop words).

Hashtags represent the tags given by a user to a tweet. However, due to the lack of constraints and common criteria, users can freely use hashtags, either reusing existing tags or defining new ones. Any different hashtag is represented with a dimension in the feature space.

Title is not linked to specific terms, but it refers to the degree of matching between the tweet content and the item title. Each possible title - related to an item - is represented by a dimension in the feature space.

In our test dataset composed by 32000 tweets - described in Section 4.1 - we have approximately 5400 unigrams, 14000 bigrams, 700 hashtags, and 40 titles for a total amount of 20000 features. Note that we discarded features occurring just in a single tweet.

3.3.2 Feature Normalization

Features extracted from a tweet must be normalized [6] in order to differentiate their importance in representing the tweet content. This task can be achieved in different ways. We evaluated the use of (i) binary weighting and (ii) the TF-IDF schema.

In the case of *binary weighting*, a feature can be either 1 (if present) or 0 (if absent), without differentiating among terms with different significance.

Classic text processing [21] uses the TF-IDF weighting schema to discriminate the different features [23]. According to this model, the TF (term frequency) component takes into account the repetitions of a term inside a document, while the IDF term (inverse document frequency) considers how common a term is in the whole collection of documents.

In our normalization schema we ignored the TF term, assuming the absence of repeated terms because of the limited length of tweets. Given a tweet t, the IDF term for its feature w is given by

$$\text{idf}(w, T) = \log \frac{|\mathcal{T}|}{|\{t \in \mathcal{T} : w \in t\}|} \quad (2)$$

where $|\mathcal{T}|$ is the cardinality of the set of sample tweets and the denominator is the number of documents in which feature w occurs. After having calculated this value, groups of features of the same type (unigrams, bigrams, etc, ...) are normalized by dividing for the maximum value of the group, so that the value for each feature is in the range [0,1].

We excluded the features of type title from TF-IDF normalization.

Property	Value
Items	40
Tweets	31694
Avg tweets per item	792.4
Avg terms per tweet	15.3
Avg stop words per tweet	5.2
Tweets with URLs	46.2%
Tweets with hashtags	40.4%
Tweets with exactly 1 hashtag	30.0%
Tweets with exactly 2 hashtags	7.6%
Tweets with at least 3 hashtags	2.8%

Table 2: Dataset statistics

4. EXPERIMENT SETTINGS

In this section we describe the testing methodology and the metrics used to tune and validate our solution, together with the dataset of tweets collected from Twitter.

4.1 Dataset

For the purpose of automatically testing the performance of our classifier, tweets are required to be classified. However, recent Twitter policies have limited the availability of public datasets of tweets. Thus, we collected a dataset of classified tweets of movies[2]. In order to automatically classify the tweets retrieved, we assumed that, given the official Twitter page of a movie[3] (or a similar page, such as the movie fan page), all tweets posted on such page refer to that movie. On the basis of this assumption, confirmed by a manual analysis on some Twitter pages, we identified 40 Twitter pages concerning 40 recent movies. Hence, using the Twitter RESTful Web Services we collected 800 tweets for each item, among the most recent, for a total of 32000 tweets. The dataset has been collected in September 2012 and is composed by English tweets. Table 2 reports its main statistical properties.

4.2 Testing Methodology

[2]The dataset is available for download at `http://home.dei.polimi.it/cremones/recsys/Microblog_Item_Detection.zip`

[3]E.g., the official Twitter page of the movie Harry Potter is related to the account '@HarryPotterFilm' hosted at `http://twitter.com/HarryPotterFilm`

The performance of the classification system have been measured using a standard hold-out dataset partitioning.

For each one-class classifier, we randomly selected 200 tweets to form the *test set*. The class (i.e., the item) of tweets in the test set is assumed to be unknown to the classifier and is used to verify if the predicted class corresponds to the actual class. Part of the remaining tweets has been used to form the *training set*, used by stage 3 as sample tweets for the learning phase.

Taken a generic class-c classifier, its performance has been measured on the test set computing common classification accuracy metrics: recall, precision, and f-measure [21]. *Recall* is defined as the percentage of class-c tweets that have been classified as class c. *Precision* is defined as the percentage of tweets classified as class c that are actually class-c tweets. Finally, *F-measure* is defined as the harmonic mean between precision and recall. In our study, in order to give more importance to precision with respect to recall, we have used a special implementation of F-measure, referred to as $F_{0.5}$-measure:

$$F_{0.5} = \frac{\left(1 + 0.5^2\right)(P \cdot R)}{0.5^2 \cdot P + R} \quad (3)$$

where precision (denoted by P) weights twice as much as recall (denoted by R).

The global performance of the battery of one-class classifiers has been computed as the macro-average of the single metrics, i.e., (arithmetically) averaging the performance indicators of each single one-class classifier. An ideal classifier should have recall, precision, and f-measure equal to 1. We graphically represented the performance indicators using precision versus recall plots.

5. STAGE 3: PARAMETER VALIDATION

In this section we focus on stage 3, the only component which requires a learning phase. Furthermore, stage 3 operates on the non-trivial tweets that the first two stages were not able to classify.

Different combinations of features can lead to different performance of the SVM classifier; in Section 5.1 we experiment several variants of feature vectors in order to find the optimal settings. Finally, in Section 5.2 we will study the impact of different factors on the classification performance.

5.1 Feature analysis

In this section we look for the optimal combination of features used to classify tweets with the SVM algorithm.

Initially, we considered one type of features at a time. Hence, we trained the SVM using such set of features and compared the classification performance both in the case of binary and TF-IDF schema normalizations. The TD-IDF normalization has been used only for unigrams (uni), bigrams (bi), and hashtags (HT).

As for the features of type 'title', we tested two weighting schemas, percentage (%) and binary weighting (bin). Percentage weighting is the matching score computed by stage 2, while binary weighting assigns 1 if the matching score is at least 75%, 0 otherwise. The choice of this threshold derives from the results of the experiments performed on stage 1 and 2. Table 3 reports the recall and precision values.

In these experiments we selected the configurations leading to the highest precision, regardless of the recall. Unigrams, bigrams, and hashtags have been proven to reach the best performance using the TF-IDF schema, which outperforms the binary weighting - in terms of precision - of about 10-13%. As for the title, we selected the binary weighting, being the one with the best precision and recall. It is worth noting that the precision obtained using only bigrams is slightly higher than the one computed by using only unigrams, while the recall is lower; in fact, a bigram is more significant than an unigram, but much less frequent.

Feature	Weight	P	R
uni	idf	0.636	0.122
	bin	0.505	0.110
bi	idf	0.671	0.037
	bin	0.576	0.050
HT	idf	**0.807**	0.176
	bin	0.704	0.166
title	%	0.228	0.086
	bin	0.501	**0.779**

Table 3: Feature comparison. Metrics are reported in correspondence of the maximum precision point. The table reports the binary (bin) and the TD-IDF normalization (idf) for the features: 'unigrams' (uni), 'bigrams' (bi), and 'hashtags' (HT). As for the feature 'title' the table reports the binary (bin) and the percentage (%) weighting schema.

Feature	P	R	$\mathbf{F_{0.5}}$
HT	0.777	0.169	0.452
HT+T	**0.789**	0.300	0.595
HT+T+Bi	0.695	0.403	**0.607**
HT+T+Bi+Uni	0.464	**0.628**	0.490

Table 4: Comparison of the best feature combinations. Metrics are reported in correspondence of the maximum precision point.

Finally, note that all recalls are very low. For instance, the use of hashtags (with idf weighting schema) led to the highest recall (as well as precision), which, however, is only 18%. For such reason, in these further experiments we took into account also the other features, focusing on the combination leading to the highest $F_{0.5}$-measure, so to improve recall while maintaining high values of precision.

Therefore, we considered pairs of features, by combining hashtags with either unigrams, bigrams, or title. The combination hashtags+title provided the best $F_{0.5}$-measure, with a precision equals to 80% and a recall equals to 60%. Once established that the best two-feature combination is given by hashtags and title, we explored the addition of a third feature, either unigrams or bigrams. The experiments showed that the best three-feature combination is hashtags+title+bigrams, with an $F_{0.5}$-measure equals to 0.607, thanks to an increment of precision (88%) that compensates a significant decrease of recall (down to 12%).

We finally compared the best single feature, the best two-feature combination, and the best three-feature combination with the use of all four kinds of features. The results are reported in Table 4. The best combination of features - in terms of $F_{0.5}$-measure - is given by hashtags+title+bigrams. Therefore, in the following we will use this combination of features for stage 3.

# tweets per item	P	R
100	**0.730**	0.216
200	0.741	0.314
400	0.695	0.403
600	0.690	**0.475**

Table 5: Sensitivity analysis of the SVM algorithm (stage 3): impact of the number of tweets per item. Metrics are reported in correspondence of the maximum precision point.

# items	P	R
5	**0.956**	**0.541**
10	0.769	0.488
20	0.733	0.432
40	0.695	0.403

Table 6: Sensitivity analysis of the SVM algorithm (stage 3): impact of the number of items. Metrics are reported in correspondence of the maximum precision point.

5.2 Sensitivity analysis

Several factors can impact the performance of the SVM algorithm. In the next sections we answer to the following three questions: (i) "how many tweets are required to obtain a good classification quality?", (ii) "does performance degrade by adding more items?", and (iii) "how are out-of-scope tweets classified?".

Number of tweets per item.

In all previous experiments we used a fixed training set. However, the *number of tweets per item* can affect the SVM algorithm performance. Indeed, the larger the sample of classified tweets, the higher the confidence of the classifier.

We fixed the test set as composed by 200 tweets per item and we used part of the remaining tweets to train stage 3. We varied the number of training tweets per item in the range $[100, 600]$. The results are reported in Table 5 and show that performance increases with the number of tweets up to 400. Increasing the number of tweets per item from 400 to 600 only slightly affects the performance.

Number of items.

This set of experiments evaluates how the number of items in the catalog can affect the quality of stage 3.

In fact, the higher the number of items the harder the classification task. We tested the SVM algorithm by varying the catalog size in the range $[5, 40]$. Table 6 confirms that the performance decreases with the number of available items. However, we can observe that the decrease between 20 and 40 items is minimal, proving a good robustness of the classifier.

Out-of-scope tweets.

In the rest of the paper we have tested the classifiers in an uncontaminated environment, where we predicted the classes only of tweets associated to the items in the catalog. However, if we take into consideration the tweets posted by a typical Twitter user, only a small part of his/her comments is likely to refer to one of the movies in the catalog.

# out-of-scope tweets	P	R
0	**0.695**	**0.403**
2000	0.674	0.400
4000	0.652	0.395
8000	0.638	0.398

Table 7: Sensitivity analysis of the SVM algorithm (stage 3): impact of out-of-scope tweets. Metrics are reported in correspondence of the maximum precision point.

Thus, in this section we verify how out-of-scope tweets (i.e., tweets not referring to any known item) are classified. We collected a set of further 8000 tweets retrieved from 10 different general-purpose, popular Twitter pages (e.g., NHL, IBM, etc.). These tweets have been added to the test set, which now contains 16000 tweets, 8000 regarding the 40 known items and 8000 regarding *out-of-scope* subjects.

Basically, the experiments showed that the presence of out-of-scope tweets does not impact the performance of the SVM algorithm, whose recall, precision, and fallout resulted almost identical to the previous tests, as reported in Table 7.

6. RESULTS

In this section we present and discuss the main results of the proposed item detection system. For this set of experiments we trained stage 3 with a training set composed by about 400 tweets per item using the three types of features derived in Section 5.1: hashtags, title, and bigrams. Figure 3 reports the precision vs recall comparing the quality of the stage, the quality of stage 1 and stage 2, with the quality of the overall pipeline composed by the three stages.

Stage 1 (solid diamond) obtains a high precision (96%), against a recall of only 11%. This result confirms that tweets containing the exact title of a movie effectively refer to that movie. On the other hand, the low recall suggests that only a few tweets contain the exact movie title. In fact, the title might not have been used at all, or it might have been contracted, misspelled, or mangled.

The dashed line shows the performance of stages 1+2. The line has been obtained by varying the acceptance threshold of stage 2. The optimal value (circle point) is in correspondence of a threshold equals to 75%: recall is 58% and precision 96%. Stage 2 allows to increase recall while maintaining the same precision. The power of stage 2 is given by its capability of searching for the title also within other terms (e.g., hashtags, mentions, abbreviations)

The solid line in Figure 3 shows the performance of the whole pipeline composed by the three stages. Stage 2 threshold has been set to 75%, while the threshold on the decision value of stage 3 has been varied. We note that stage 3 does not significantly affect precision, but it improves recall.

6.1 Aggregation policies

The proposed item detection system allows to assign a tweet to multiple items. This happens when multiple one-class classifiers classify the tweet.

However, the dataset we have collected contains only tweets related to one single item (see Section 4.1). In order to check the correctness of the results obtained in the previous sections, we run a set of experiments where we forced the system to return one class at most. In the following, we verify

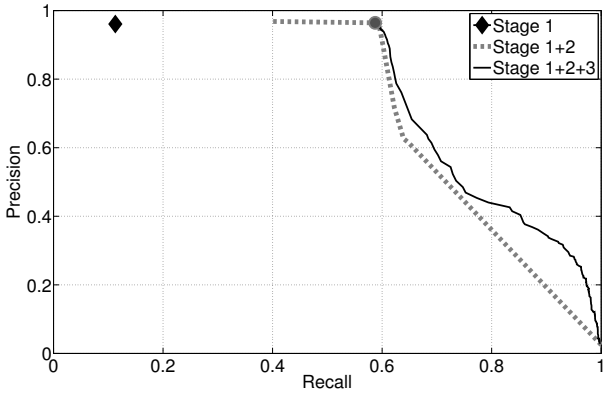

Figure 3: System performance of the battery of one-class classifier in term of precision vs recall

Aggregation policy	P	R	$F_{0.5}$
No aggregation	**0.950**	0.595	**0.849**
Max precision	0.922	0.598	**0.832**
Max decision value	0.917	**0.654**	**0.849**

Table 8: Comparison between the two aggregation policies, with respect to the baseline approach where no aggregation is applied. Metrics are reported in correspondence of the maximum precision point.

that the presented performance does not depend on the fact that the item detection system allows a multiple-item assignment, while the tweets we classified referred to a single item. We tested two possible aggregation strategies: (i) max decision value and (ii) max precision policies.

Given a tweet and a subset of items it has been classified into, the *max decision value policy* assigns the tweet to the class with the highest decision value resulting from stage 3.

Alternatively, the *max precision policy* uses a subset of the input sample tweets to estimate the precision of each single one-class classifier. Therefore, in the case of ambiguous assignment, it prefers the classifier with the higher estimated precision. Thus, for each one-class classifier, we used the 400 tweets of the training set to train the SVM algorithm, and an additional set of 200 tweets (whose class is known) - referred to as *validation set* - to optimize the SVM parameters. Parameters are set so to maximize the precision computed on the validation set. Afterwards, we classify the tweets in the test set (composed, again, by 200 tweets whose class is assumed to be unknown) using the parameters computed on the validation set. In the case multiple classifiers classify a tweet, the tweet is assigned to the class corresponding to the classifier with the highest precision on the validation set.

Table 8 reports the outcome of these experiments and confirms that the performance reported in the previous tests were not affected by the multiple-assignment hypothesis. In fact, neither maximum precision nor maximum decision value policy bring a significant change - in term of $F_{0.5}$-measure - with respect to not applying any aggregation strategy: in particular, both aggregation strategies bring to a decrease of precision, compensated by a slight increase of recall.

7. CONCLUSIONS

In this work we defined and evaluated a three-stage item detection system. The system is based on a battery of one-class classifiers for the matching between tweets and TV programs.

Empirical tests on 32000 tweets have proven the ability of the system to accurately classify tweets. The scalability of the system allows for an extensive usage in VoD and Linear TV platforms, both as mechanism for preference elicitation and recommendation and as audience meter.

Interesting for future work would be the inclusion of temporal information (e.g., matching the time message was posted with the TV program scheduling in Electronic Programming Guide) and the analysis of tweets as conversations (e.g., a sequence of tweets with questions/answers). This additional information could be used, for example, to disambiguate two movies with the same title but released in two different years. Furthermore, item detection associated with opinion mining (e.g., [20]) will allow to infer the polarity (e.g., positive or negative) of user preferences.

8. REFERENCES

[1] D. Andrzejewski and X. Zhu. Latent dirichlet allocation with topic-in-set knowledge. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, SemiSupLearn '09, pages 43–48, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[2] D. Andrzejewski, X. Zhu, and M. Craven. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 25–32, New York, NY, USA, 2009. ACM.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[4] M. Cataldi, L. Di Caro, and C. Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, pages 4:1–4:10, New York, NY, USA, 2010. ACM.

[5] E. Charniak, C. Hendrickson, N. Jacobson, and M. Perkowitz. Equations for part-of-speech tagging. In *In Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789, 1993.

[6] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1185–1194, New York, NY, USA, 2010. ACM.

[7] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sept. 1995.

[8] C. Courtois and E. D'heer. Second screen applications and tablet users: constellation, awareness, experience, and interest. In *Proceedings of the 10th European conference on Interactive tv and video*, EuroiTV '12, pages 153–156, New York, NY, USA, 2012. ACM.

[9] P. Cremonesi, F. Garzottto, and R. Turrin. User effort vs. accuracy in rating-based elicitation. In *Proceedings*

of the sixth ACM conference on Recommender systems, RecSys '12, pages 27–34, New York, NY, USA, 2012. ACM.

[10] E. D'heer, C. Courtois, and S. Paulussen. Everyday life in (front of) the screen: the consumption of multiple screen technologies in the living room context. In *Proceedings of the 10th European conference on Interactive tv and video*, EuroiTV '12, pages 195–198, New York, NY, USA, 2012. ACM.

[11] M. Doughty, D. Rowland, and S. Lawson. Who is on your sofa?: Tv audience communities and second screening social networks. In *Proceedings of the 10th European conference on Interactive tv and video*, EuroiTV '12, pages 79–86, New York, NY, USA, 2012. ACM.

[12] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 42–47, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[13] T. L. Griffiths, M. Steyvers, D. M. Blei, and J. B. Tenenbaum. Integrating topics and syntax. In *In Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press, 2005.

[14] B. Han and T. Baldwin. Lexical normalisation of short text messages: makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 368–378, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[15] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998. Springer-Verlag.

[16] M. Lochrie and P. Coulton. Mobile phones as second screen for tv, enabling inter-audience interaction. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, ACE '11, pages 73:1–73:2, New York, NY, USA, 2011. ACM.

[17] M. Lochrie and P. Coulton. Sharing the viewing experience through second screens. In *Proceedings of the 10th European conference on Interactive tv and video*, EuroiTV '12, pages 199–202, New York, NY, USA, 2012. ACM.

[18] J. Makkonen, H. Ahonen-Myka, and M. Salmenkivi. Simple semantics in topic detection and tracking. *Inf. Retr.*, 7(3-4):347–368, Sept. 2004.

[19] Nielsen. Nielsen and twitter establish social tv rating. http://www.nielsen.com/us/en/insights/press-room/2012/nielsen-and-twitter-establish-social-tv-rating.html, 2012.

[20] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, Jan. 2008.

[21] M. F. Porter. Readings in information retrieval, 1997.

[22] D. Ramage, S. Dumais, and D. Liebling. Characterizing microblogs with topic models. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*. AAAI, 2010.

[23] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, Aug. 1988.

[24] D. Tax. *One-class classification - Concept-learning in the absence of counter-examples*. PhD thesis, TU Delft, 2001.

[25] S. Wakamiya, R. Lee, and K. Sumiya. Towards better tv viewing rates: exploiting crowd's media life logs over twitter for tv rating. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '11, pages 39:1–39:10, New York, NY, USA, 2011. ACM.

[26] M.-C. Yang, J.-T. Lee, and H.-C. Rim. Using link analysis to discover interesting messages spread across twitter. In *Workshop Proceedings of TextGraphs-7: Graph-based Methods for Natural Language Processing*, pages 15–19, Jeju, Republic of Korea, July 2012. Association for Computational Linguistics.

[27] J. T. yau Kwok. Automated text categorization using support vector machine. In *In Proceedings of the International Conference on Neural Information Processing (ICONIP*, pages 347–351, 1998.