Discovery of Trending Videos on Twitter

Victor Spiridon - 4184599

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, the Netherlands. April 24, 2013
v.spiridon@student.tudelft.nl

Abstract. Microblogging platforms such as Twitter have witnessed a constant growth in popularity over the last few years. These short messages come by the millions of billions every week, and contain loads of information: from personal statements, to product reviews, advertisements and news. This high rate of information can be exploited for social, political, financial and marketing studies. We choose to use this high flux of information for discovering trending YouTube videos. A daily list of trending videos can represent the starting point for a *news platform* with respect to videos posted on YouTube. Our experiments are based on a corpora of 3.96 million tweets collected over a period of 112 hours. Furthermore, we propose a technique for identifying videos which represent spam on Twitter.

Keywords: Twitter; YouTube; trend; data mining; pig; mapreduce

1 Introduction

Twitter[1] is one of the fastest and constantly growing social networks of the Internet and recently it became one of the top websites in the US. It consists of a micro-blogging platform on which users can post messages (tweets) containing at most 140 characters. In March 2011, it would take only 1 week for a billion tweets to be posted [2]. During the 112 hours period (between April 18, 2013, 20:23:28 CET and April 23, 2013, 12:08:27 CET) a number of 1,685,747 million tweets were posted. Most of the users are English speaking and live in the US.

YouTube[2] is the biggest website for freely hosting video clips. Depending on the popularity of a specific video and the country of the visitor, advertisements can be included before the actual video. In order to increase the efficiency of the advertisements, quickly identifying the trending videos can be very useful, as more viewers are reached in the same amount of time. Usually a trending video is viewed by a group of viewers with common preferences. Therefore, by studying these commonalities, the best advertisement can be selected for the respective group, increasing the chances that a viewer clicks on the advertisement.

We choose to use Twitter as a data input for identifying the video trends. Twitter and YouTube are among the biggest websites on the World Wide Web,

[1] http://www.twitter.com
[2] http://www.youtube.com

see Figure. 1, therefore it is to be expected to find many references of YouTube links in tweets.

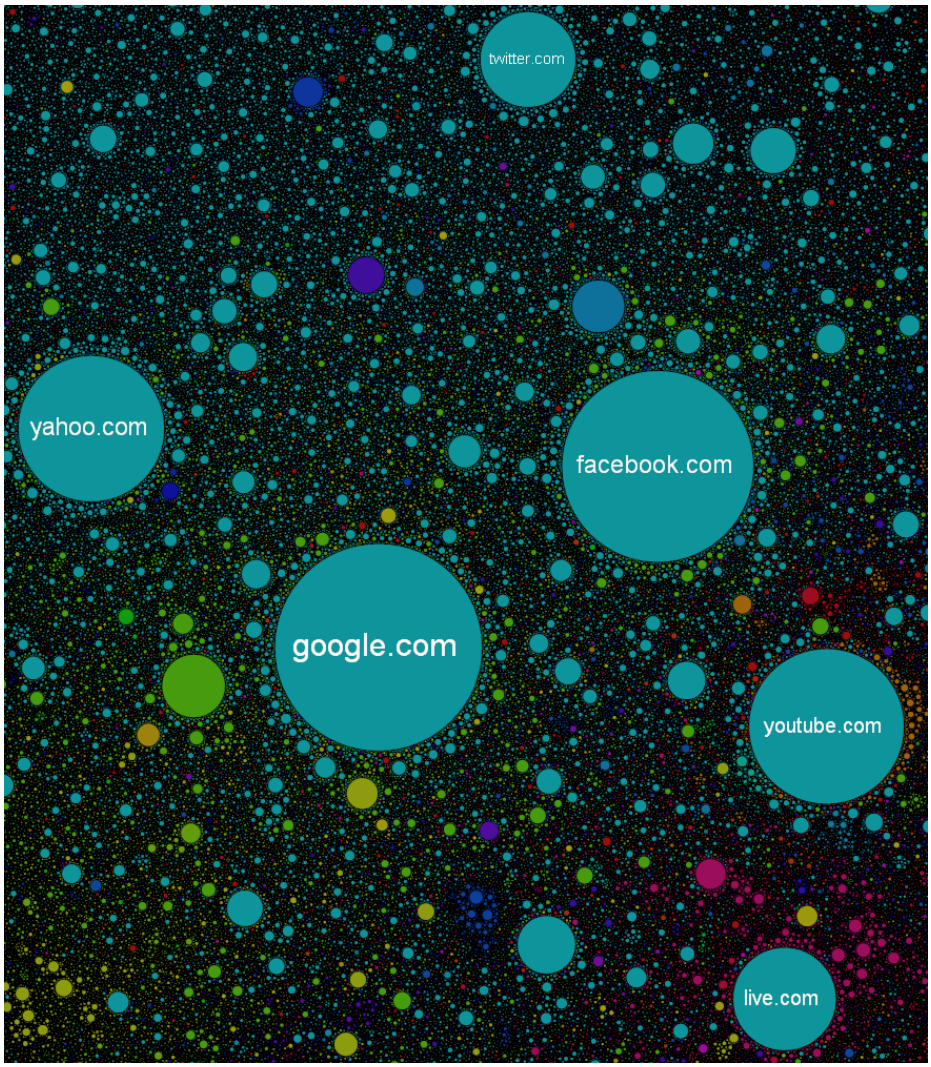


Fig. 1: Central part of the Internet Map. Size describes the number of visitors. [1]

In section 2 we are going to present the general work-flow of identifying video trends, followed by section 3 describing the results, section 4 concerned with filtering the spamming videos and finally the conclusion and future work.

2 General Work-flow

In this section the main steps of gathering and processing the tweets are described.

2.1 Gathering the tweets - Twitter API

For collecting the tweets the Twitter search Web API is used. The limit of the API for an user is 180 pages per 15 minutes and each page can contain at most 100 tweets. An answer to a query can contain multiple pages. The search query string used was "youtube filter:links" and the tweet's language was set to English. In order to ensure that we are not going to reach the limit of the API, two dev users were registered which were used in a round-robin fashion to query to API.

In addition to the text, each tweet comes with information such as the posting user and its ID, and the tweet's ID. Each tweet has an unique ID and the values of this parameter grow over time, so using this ID we were able to calculate the number of tweets posted on Twitter over our collection period. The API sends a lot of tweets ordered from the present moment to the past. Therefore answers to subsequent queries can contain only partly new tweets and a lot of old tweets. One solution to tackle this is to send queries rare enough such that enough tweets had been posted in the mean time. However, such a solution possibly misses a significant number of tweets. A better solution is to send queries more often, but in this case we have to ensure that we don't store duplicate tweets. In order to tackle this, we can use the the option to set the "sinceID" parameter to the query, such that the answer does not contain tweets with an ID smaller (older) than a specified ID. Unfortunately, we had found that this functionality is not always working. Therefore, we chose to implement this functionality ourselves, that is, request new pages of a certain answer, until we encounter the maximum ID of the last query's answer. In this way we ensure that we don't miss tweets, and that we don't store duplicates. Furthermore, we were pleased to observe that the conservative nature of this approach also prevents us from being limited. Therefore we are able to issue a query every 20 seconds. Usually, we received from 100 to 1000 tweets per answer.

The tweets were dumped in a new text file every 15 minutes, together with the Tweet ID, the user's name and ID and the post date and the retweet count.

2.2 Filtering the links

Having the tweets' text we want to extract the links to the videos from the text. Finding the links is somewhat made easier by the fact that each link comes shortened to `http://t.co/` links by the Twitter API. Furthermore, there is a one to one match from a short link to a shortened link, therefore we were sure that we are not counting the number of appearances of `t.co/` links but the number of appearances of videos. To accomplish this task we use `Pig-0.10.0`, a

high-level platform for creating `MapReduce` programs used with `Hadoop`. In order to extract the link we can use the following naive algorithm:

1. For each tweet: create a tuple (user, text).
2. For each text: tokenize the text using a set of delimiters.
3. Keep only tokens representing links.
4. Save tokens.

However, there are two reasons why this algorithm will not work:

- Most of the tweets contain Unicode characters:
- The link may not be separated from other words.

See the following example of a tweet: `\u5f3e\u4e38\u30c4\u30a2\u30fc\u88cf\u5c71\u3060\u3057\u30fc\nhttp:\/\/t.co\/TXcqTw4P`

The two reasons make the tokenization very hard as an exhaustive set of delimiters has to be identified. To overcome this, we are going to use the following algorithm:

1. For each tweet: create a tuple (user, text).
2. For each text: extract the links from the text.
3. Keep only not null links.
4. Save links.

This algorithm poses a different challenge as it cannot be implemented using only native `Pig` functions. Therefore we are going to define an UDF (User Defined Function) in Java. For more details about the code see Appendix 1.

2.3 Counting the links

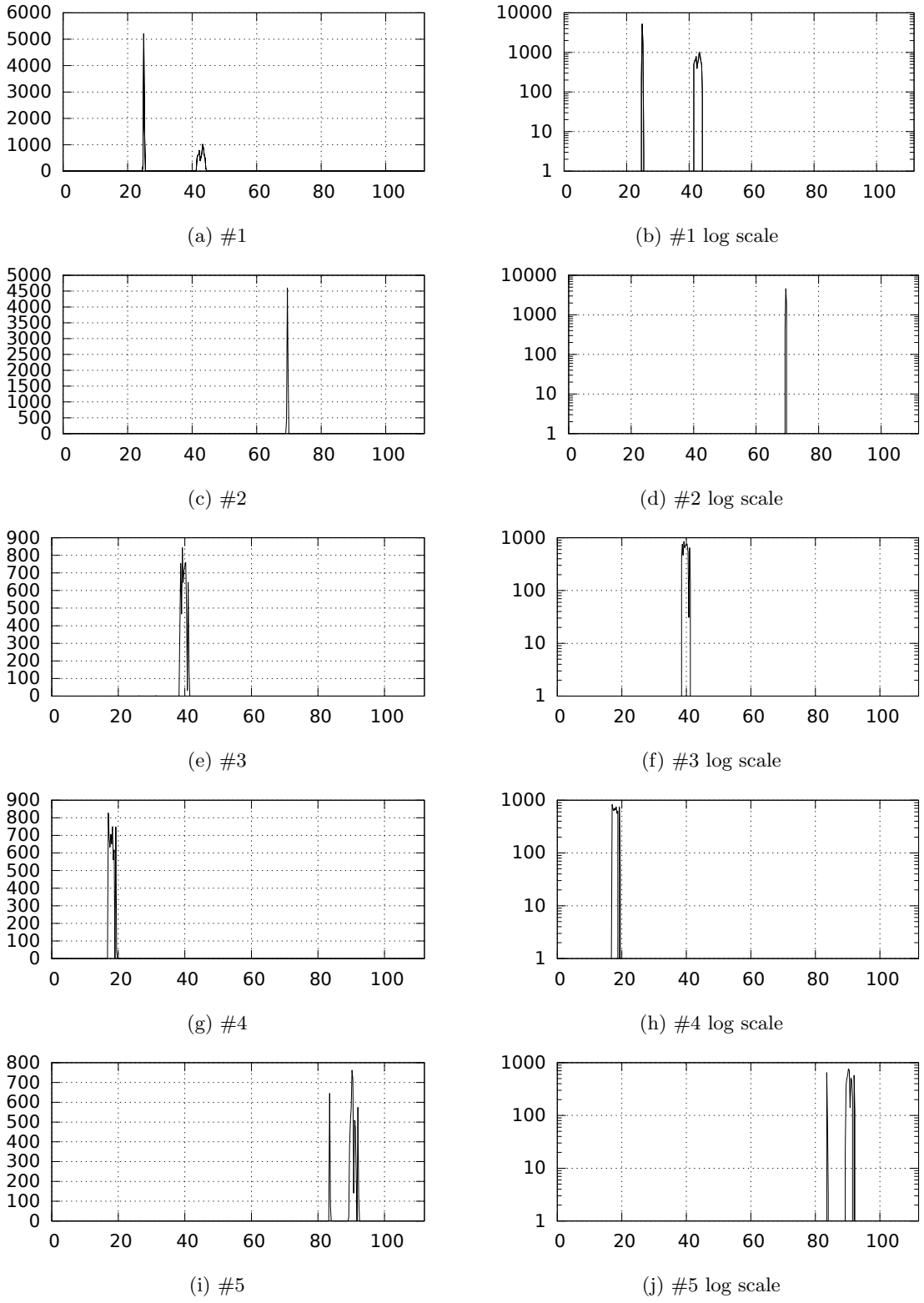
The approach for finding the most frequently mentioned videos is to count the number of appearances of every link, in the corpus. Then the links are displayed, ordered by the number of appearances. The `Pig Latin` code can be found in Appendix 2. Information about the videos is shown in Table 1.

3 Results

In order to further study the number of times the above selected videos are mentioned we also counted the number of hits every 15 minutes. The results are plotted in Figures 2, 3, 4, 5. On the left column of graphs the results are plotted on a linear scale in order to show the height of the spikes. On the right column of graphs the results are plotted on a logarithmic scale in order to better show the low values. The x-axis represents the time in hours. The y-axis represents the number of hits per 15 minutes.

ID	YouTube ID	Hits	Classification	YouTube user - clip name	YouTube views	Published on
#1	Xyy_nge5VTg	14222	spam, long movie	Izuniy - Injustice Gods Among Us The Movie 2013 - All Cutscenes / Cinematics - HD	178,113	Apr 16, 2013
#2	Se5b7e8a8JQ	6765	spam, music	Mathew Schulz - Matthew Schultz feat. Jim Jones - Money or Me	1,263,990	Mar 5, 2013
#3	MO5RSTVfcSI	6683	spam, music	Al-FatirTV - Al-Fatir - ODB [Prod. Jaynez][Official Music Video]	347	Apr 19, 2013
#4	VxnW2HiJ8JE	6221	spam, short movie	Tom Kim - Miami Shuffle **OFFICIALLY AWESOME**	138,387	Apr 17, 2013
#5	08IU0fGK1O0	5710	spam, commercial	TheCashmoneygoldmine - How to use Cash Money Goldmine	4,402	Apr 17, 2013
#6	3Z_u7fwPNLc	4873	spam, music	HotfireProduction - Hotfire-GoodTimes Feat Elliven (official Music Video) @ilovehotfire	1,012,377	Apr 7, 2013
#7	FCdrILPKS8Y	4727	spam, commercial	fenvirantiviral - Cold Sore Blisters, Shingles and Genital Herpes Sufferers Sing Praises!	16,216	Dec 26, 2012
#8	uEkFx_M36FU	3234	spam, commercial	TheWandomWill - Herobrine's Mansion - Episode 1 - Owned by Zombies	23	Apr 21, 2013
#9	c3mDWe2-fA4	3112	spam, commercial	onlinecamerashop - Promotion new website with a lot of great features!	672	Apr 17, 2013
#10	APdyHlz5K7M	2611	spam, music	BulletRealHipHop - Keep it Moving ft Jadakiss Official	843	Apr 12, 2013
#11	1OEron4rXfk	6678	music	LanaDelRey - *SUMMER WINE*	1,068,987	Apr 18, 2013
#12	OmliBRZUxnk	5068	music	JustinBieberVEVO - Justin Bieber - All Around The World (Official) ft. Ludacris	8,884,437	Apr 12, 2013
#13	DGIgXP9SvB8	5216	music	williamVEVO - will.i.am - #thatPOWER ft. Justin Bieber	6,020,545	Apr 19, 2013
#14	y9uSyICrtow	4975	music	30SecondsToMarsVEVO - Thirty Seconds To Mars - Up In The Air	2,188,237	Apr 19, 2013
#15	p8uYcCeafCU	4336	music	CodySimpsonMusic - Cody Simpson - Pretty Brown Eyes [Audio]	261,720	Apr 18, 2013
#16	oU40LNclQEA	3462	commercial	Nickelodeon - Sam & Cat Coming to Nickelodeon in June!	154,354	Apr 18, 2013
#17	ASO_zypdnsQ	3598	music	officialpsy - PSY - GENTLEMAN M/V	208,550,486	Apr 13, 2013
#18	EGDlVvIGTZk	3427	music	CodySimpsonMusic - Cody Simpson - Pretty Brown Eyes [Official Video]	84,742	Apr 22, 2013
#19	7etjda4JCRY	2373	clip	BIGtimeRUSH - A Special Message from Big Time Rush	51,963	Apr 18, 2013
#20	e8yZCAs3Asw	2111	clip	bandits9910 - Big papi swearing at red sox game	215,926	Apr 20, 2013

Table 1: Table with details about the videos plotted.
"ID" specifies the tweets id as we will reffer to it in this report.
"YouTube ID" is the address the video can be accessed at: http://youtu.be/<YouTube ID>.
"Hits" represents the number of tweets collected that contain the link to the video.
"Classification" contains keywords manually defined in order to describe the clip.
"YouTube user - clip name" specifies the uploaded and the clips name as shown on its YouTube page.
"YouTube views" specifies the number of views of that respective video at April 23, 2013.
"Published on" represents the date the video clip was published.



(a) #1 (b) #1 log scale

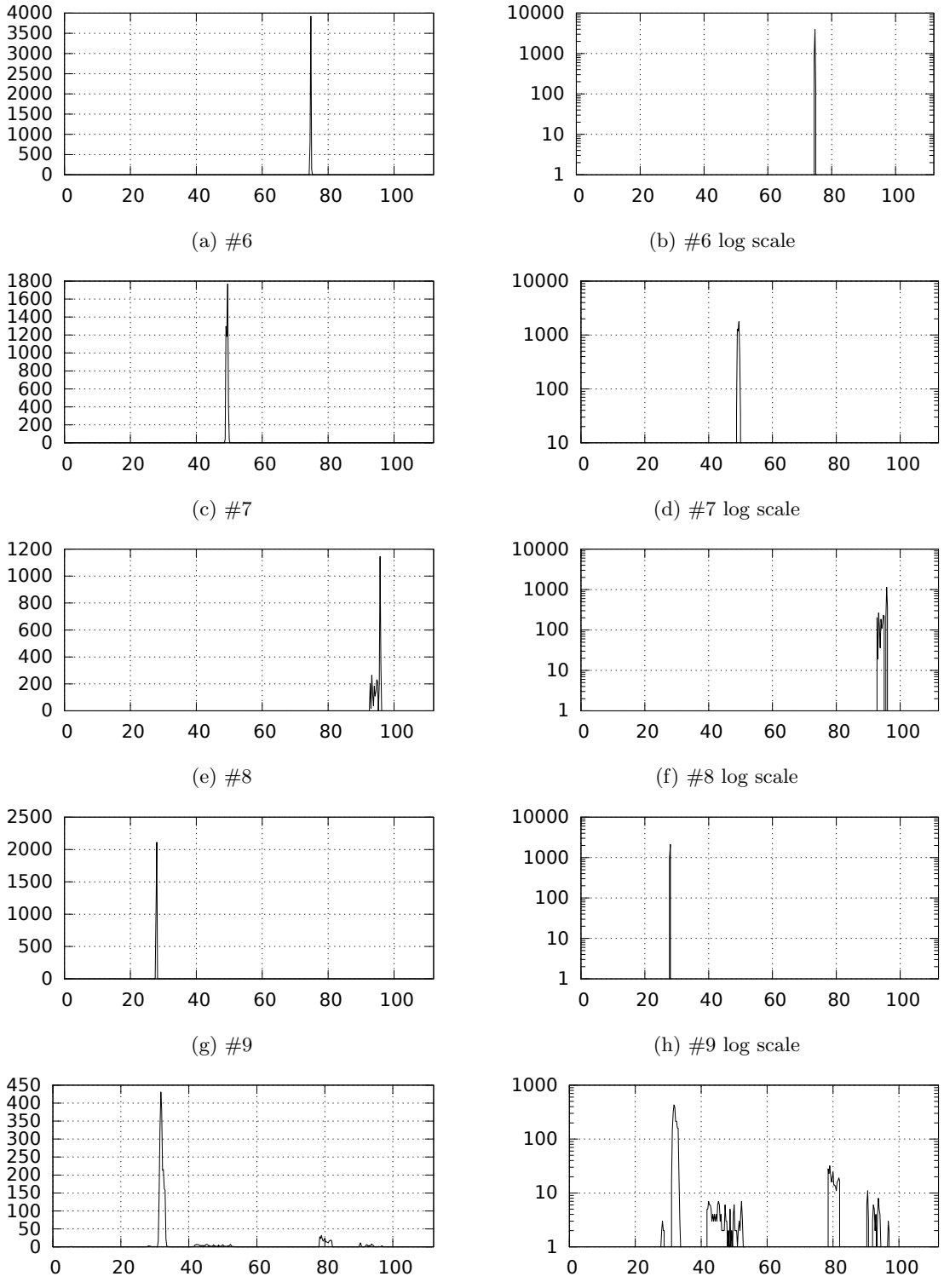
(c) #2 (d) #2 log scale

(e) #3 (f) #3 log scale

(g) #4 (h) #4 log scale

(i) #5 (j) #5 log scale

Fig. 2: Graphs for spamming tweets #1 - #5



(a) #6

(b) #6 log scale

(c) #7

(d) #7 log scale

(e) #8

(f) #8 log scale

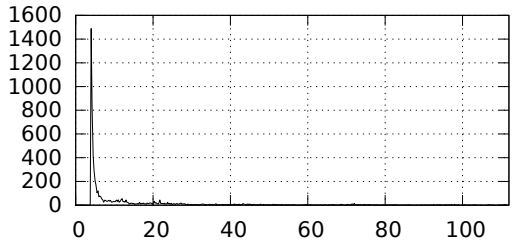
(g) #9

(h) #9 log scale

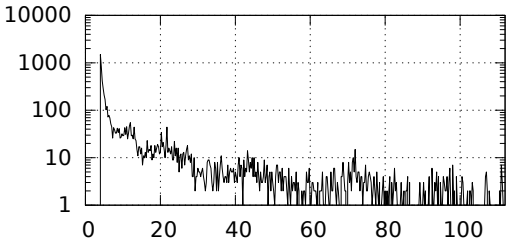
(i) #10

(j) #10 log scale

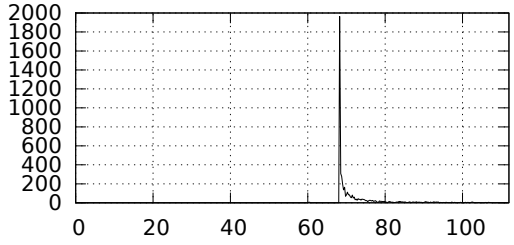
Fig. 3: Graphs for spamming tweets #6 - #10



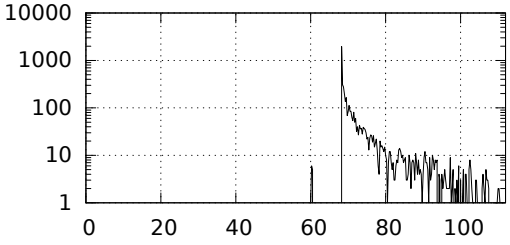
(a) #11



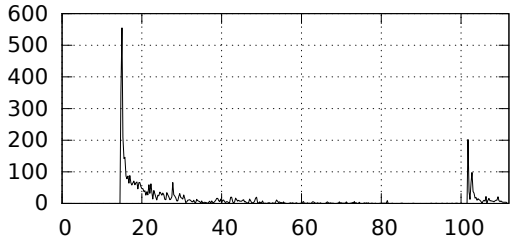
(b) #11 log scale



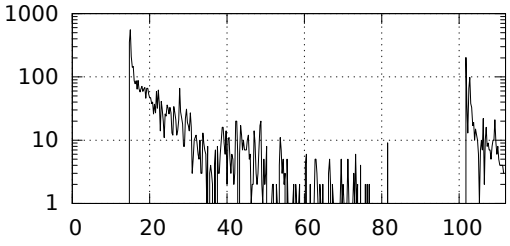
(c) #12



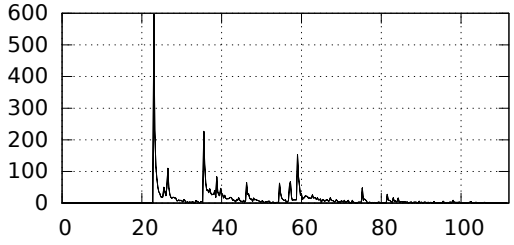
(d) #12 log scale



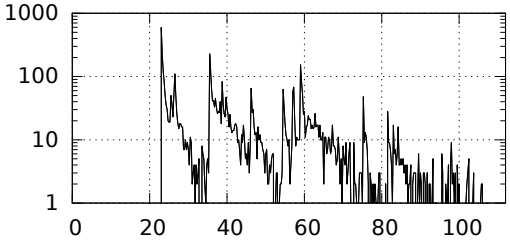
(e) #13



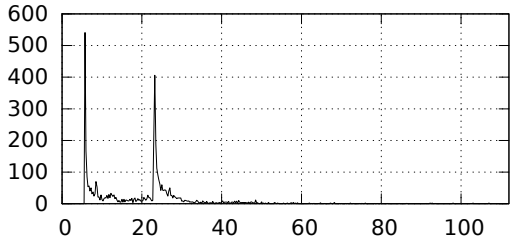
(f) #13 log scale



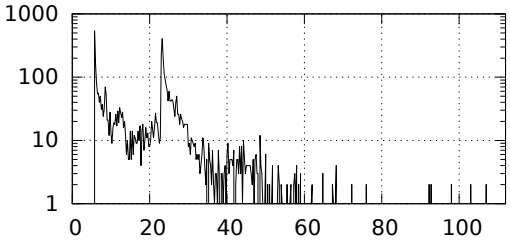
(g) #14



(h) #14 log scale

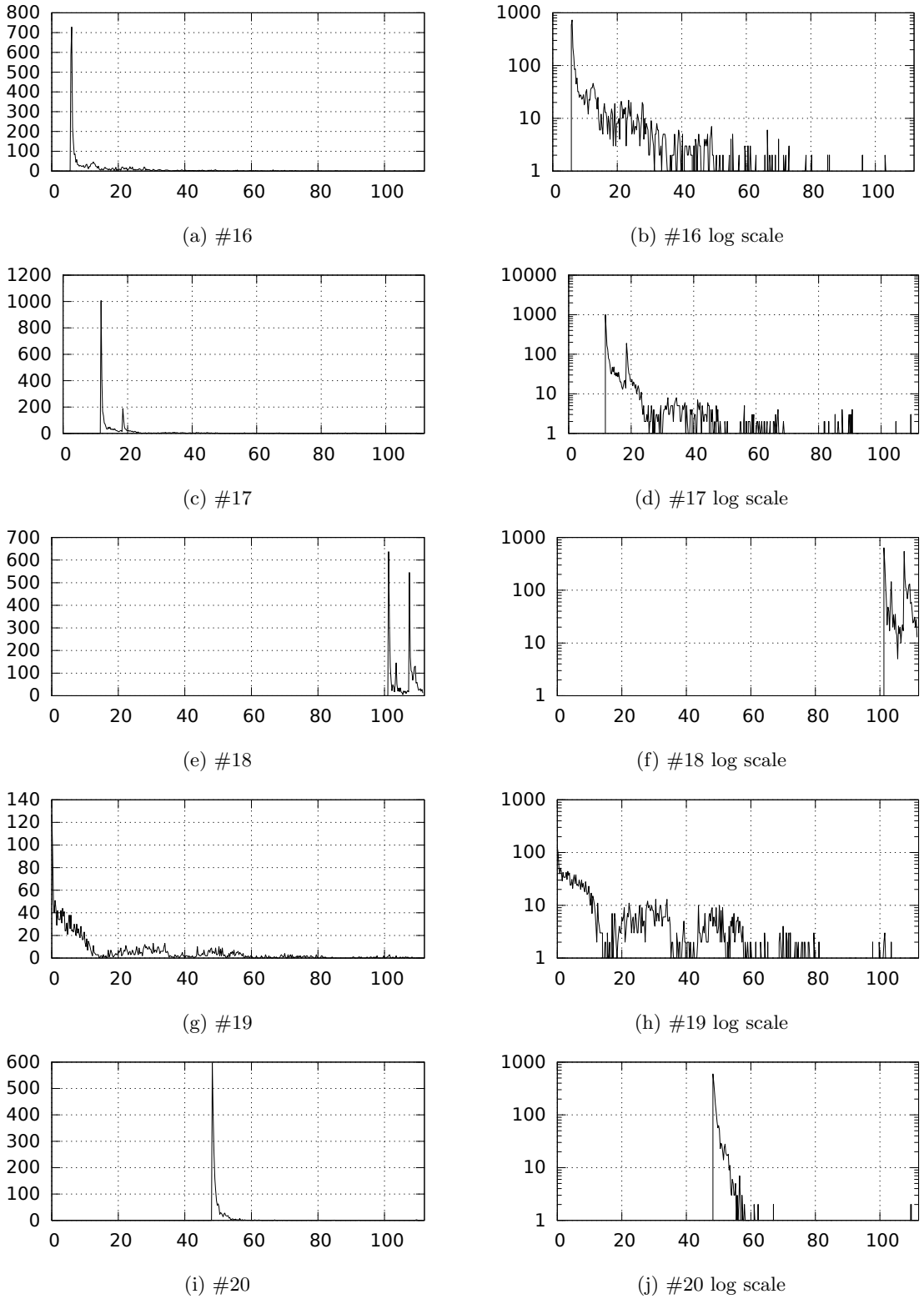


(i) #15



(j) #15 log scale

Fig. 4: Graphs for trending tweets #11 - #15



(a) #16 (b) #16 log scale

(c) #17 (d) #17 log scale

(e) #18 (f) #18 log scale

(g) #19 (h) #19 log scale

(i) #20 (j) #20 log scale

Fig. 5: Graphs for trending tweets #16 - #20

4 Finding Trending Videos

We might be tempted to consider that the most frequently mentioned videos are the trending ones. However, by inspecting the results we observed that almost half of the most popular tweets represent some sort of advertisement. These videos cannot be trending because even if they have a lot of hits on Twitter they barely have any views on YouTube. Let's take the case of the video #8 with 23 views and 3234 hits. Looking at the tweets containing this video we observe that all of them are retweets:

`"RT @TheWandomWill: I added a video to a @YouTube playlist http://t.co/ZXAT4omclk Lets Play Herobrines Mansion - Episode 1 - Owned by Zombies".`

Taking a careful look at the user names which retweeted we observe that most of the names seem like they are generated by an automatic tool: `Shara_kxah`, `Vannessa_geip`, `Dalia_4927`, `Alease_6288`, `Eric_9254`, `Kami07787`, `Trinhizspp`. By inspecting the tweets of video #11 we again observe the same retweet pattern:

`"RT @LanaDelRey: new video http://t.co/pF8at3SXog".`

However, the user names which retweeted look like names of real persons: `pauliinapop`, `JordaneDenboer`, `NateBettye`, `KarinaPavon`, `MallorieRuiz`. This discrepancy between the user names can be used as a first indicator for a identifying spamming videos. Furthermore, when we attempted to visit the suspect Twitter accounts, in all cases, we were welcomed by an "account blocked" announcement. Searching our corpus we observed that the same user names are used for different spamming videos. For example the ones mentioned above are used for #5 and #8. Also important to note that the each user name is used only once per video.

Looking at the graphs for videos #1 – #10 we see that most of them feature a high hit rate over a period of 15 minutes to several hours and then the mentioning rate suddenly goes to zero flat. This behavior is very counter-intuitive for a video people are talking about. Looking at the #11 – #20 videos, we observe a pattern which matches our intuition. At the beginning there is a spike of retweets of the original message. The wave of retweets are probably issued by the followers of the user who posted the original tweet, and then their followers and so on. The number of hits over time slowly decreases, but it never quite hits zero. Videos such as #13, #14, #15, feature more of these shapes overlapped, one for every original tweet of the respective artist. Each new one is smaller than the previous ones because over time the novelty of the video decreases. The shape of the graph of the number of hits over time can be used as a second indicator for identifying spamming videos.

The two indicators presented can be used to create a classifier system which can separate the trending videos from the spamming videos out of a set of videos which have high hit rates.

5 Conclusions and Future Work

During the work for this report we successfully managed to use the Twitter API for collecting tweets containing links to YouTube videos. Furthermore, we managed to do this in a manner which avoids being limited by the Twitter API and which avoids the collection of duplicate tweets or the losing of tweets (with respect to what the API is sending as an answer). In order to process this corpus we had got acquainted with `Pig` and developed `Pig latin` program in order to generate `Hadoop MapReduce` programs.

On the basis of the findings presented in the previous section we can formulate a hypothesis about the phenomena of video spamming on Twitter:

> There are spammers which can create a significant number of new Twitter accounts, every day, probably by exploiting vulnerabilities in the new account process. For them the cost of creating a new account is low enough, because these accounts often get blocked by Twitter. The spammers use these accounts for the purpose of advertising, probably selling this functionality as a service to other parties.
> However, in most of the cases the spam is useless because the links never get clicked, and furthermore, the tweets never get retweeted by real users. The small number of video views can be explained by the fact that nobody follows the spammers' Twitter accounts, therefore almost nobody sees the respective tweets. We can assume, that their targets are tools that aggregate tweets such as the one presented in this report.

The development of an automated classifier which can separate trending from spamming videos seems a very good direction for future work. In addition, the tweet gathering engine can be used for developing a *"trending videos" news website*. A parallel research direction can be represented by further using the metadata available for the YouTube videos in order to identify not just trending videos but trending topics.

References

1. The Internet Map, Ruslan Enikeev, `http://internet-map.net/` August 2012
2. @twitter: *#numbers*, `http://blog.twitter.com/2011/03/numbers.html` 2011

6 Appendix 1

The Pig script for extracting links:

```
REGISTER udfs.jar;

-- Load tuples of form (user, text) from the 'tweets' file
tweets = LOAD 'output.txt' AS (user: chararray, text: chararray);

-- Extract the t.co links
tokens = FOREACH tweets GENERATE udfs.SPLIT(text);

-- Keep only not null tokens
links = FILTER tokens BY $0 MATCHES '.+';

-- store the links in links.out
STORE links INTO 'links.out';
```

User defined Pig function: SPLIT:

```
package udfs;

import java.io.IOException;
import java.util.Locale;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.pig.EvalFunc;
import org.apache.pig.PigException;
import org.apache.pig.backend.executionengine.ExecException;
import org.apache.pig.data.DataType;
import org.apache.pig.data.Tuple;
import org.apache.pig.impl.logicalLayer.schema.Schema;

public class SPLIT extends EvalFunc<String> {

       private static final Pattern pattern = Pattern.compile(
                            "https?:\\/\\/t\\.co\\/[a-z0-9]+",
                            Pattern.CASE_INSENSITIVE);

    @Override
    public String exec(Tuple input) throws IOException {
        try {
            if (input==null || input.size() == 0)
                return null;

            Object text = input.get(0);
            if (text == null)
```

```
            return null;

        if (!(text instanceof String)) {
            int errCode = 2114;
            String msg = "Expected input to be chararray, but" +
                          " got " + text.getClass().getName();

            throw new ExecException(msg, errCode, PigException.BUG);
        }

        Matcher m = pattern.matcher((CharSequence) text);
        String outputString = "";
        boolean found = false;

        while (m.find()) {
            if (!found){
                    outputString +=
                        m.group().toLowerCase(Locale.ENGLISH);
                    found = true;
            }
            else
                    outputString += "\n" +
                        m.group().toLowerCase(Locale.ENGLISH);
        }

            return outputString.replace("\\", "");
        } catch (ExecException ee) {
            throw ee;
        }
    }

    @SuppressWarnings("deprecation")
    @Override
    public Schema outputSchema(Schema input) {
        Schema.FieldSchema bagFs = new Schema.FieldSchema("links",
            DataType.CHARARRAY);
        return new Schema(bagFs);
    }
}
```

7 Appendix 2

The Pig script for counting link appereances:

```
-- Load links from the 'links.out' file
input_links = LOAD 'links.out' AS (link: chararray);

-- create a group for each word
link_groups = GROUP input_links BY link;

-- count the entries in each group
link_count = FOREACH link_groups GENERATE COUNT(input_links) AS count,
    group AS link;

-- order the records by count
ordered_link_count = ORDER link_count BY count DESC;

STORE ordered_link_count INTO 'links_count.out';
```