Twitter's Visual Pulse

Jonathon S. Hare
jsh2@ecs.soton.ac.uk

Sina Samangooei
ss@ecs.soton.ac.uk

David P. Dupplaw
dpd@ecs.soton.ac.uk

Paul H. Lewis
phl@ecs.soton.ac.uk

Electronics and Computer Science, University of Southampton, United Kingdom

ABSTRACT

Millions of images are tweeted every day, yet very little research has looked at the non-textual aspect of social media communication. In this work we have developed a system to analyse streams of image data. In particular we explore trends in similar, related, evolving or even duplicated visual artefacts in the mass of tweeted image data — in short, we explore the visual pulse of Twitter.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Stream analysis; near-duplicate image detection; trend analysis; Twitter analysis

1. INTRODUCTION

There is currently a massive amount of research being performed by the natural-language understanding and data-mining communities on trying to understand and harness the mass of information being tweeted across the world. The aims of this research are diverse, from trying to understand opinions about current events, to detecting disasters as they happen, to predicting future stock prices.

Virtually all the current research looking at Twitter concentrates on the short textual messages that compose a Tweet and on the social network of Twitter users. Millions of images are tweeted every day. However, there has been very little research looking at non-textual aspects of social media communication. Recently, we've been exploring some different aspects of this mass of tweeted image data.

This paper describes a demonstration system we have built for investigating trends in streams of images in Twitter. Specifically, we've investigated and designed a system for detecting near-duplicates in live streams of images (where we *forget* after a period of time) and coupled this to the live Twitter sample stream. We detect duplicates using a graph-based approach in which Locality Sensitive Hashing is applied to local features to efficiently determine feature matches between images. The system also incorporates a modular approach to actually extracting images from Tweets. Currently we have modules for a number of the most common image hosting sites used by Twitter.

ICMR'13, April 16–20, 2013, Dallas, Texas, USA.
ACM 978-1-4503-2033-7/13/04.

2. TECHNICAL DESCRIPTION

The system is built from three main components: 1) a tool called PicSlurper, that is responsible for reading the Tweets in the Twitter stream and parsing them to extract any images; 2) a component for detecting near duplicates within the most recent images; and 3) a visualisation, showing currently trending images as well as historical trends. The system is able to process the Twitter sample stream (>100,000 images per day) in real-time on a standard PC. The system built on top of OpenIMAJ [3] and is available as part of the OpenIMAJ codebase[1].

2.1 PicSlurper

Tweets do not themselves contain images; rather they contain links to images hosted by a variety of different providers. This means that to extract images from Tweets, we need to be able to extract the links from the Tweet and resolve the image content. Even this is not simple, as each image hosting provider uses a different technique for providing the image; usually the image is embedded in an html page.

PicSlurper is a tool we have developed to extract the images from a stream of Tweets (either read from a file, or in real-time from the live Tweet data provided by the Twitter API). Internally, PicSlurper has a set of consumer modules that deal with specific hosting sites. When PicSlurper extracts a URL from a Tweet, it asks each module if it is able to extract an image from the URL. If a module is able to extract the image, then the module is used to download the image, and the image and the respective Tweet metadata are emitted for further processing. If no module is able to deal with the link(s) in the Tweet (or if there are no links), then the Tweet is ignored and nothing is emitted. Currently, we have modules for a number of the most popular image hosting services used with Twitter: Facebook, imgur, Instagram, ow.ly, Tmblr, Twiple, TwitPic, yfrog, and of course Twitter's own image hosting service. Using PicSlurper with the Twitter sample stream[2] we are able to extract over 100,000 images per day; the actual number of tweets processed is far higher than this though (it varies a lot, but between 30 & 80 tweets per second is normal).

2.2 Streaming Duplicate Detection

Our streaming duplicate detection is heavily inspired by the techniques proposed in [2], however it does have a number of differences. Most notably, in our technique, we are

[1] `http://www.openimaj.org`
[2] `https://dev.twitter.com/docs/api/1.1/get/statuses/sample`

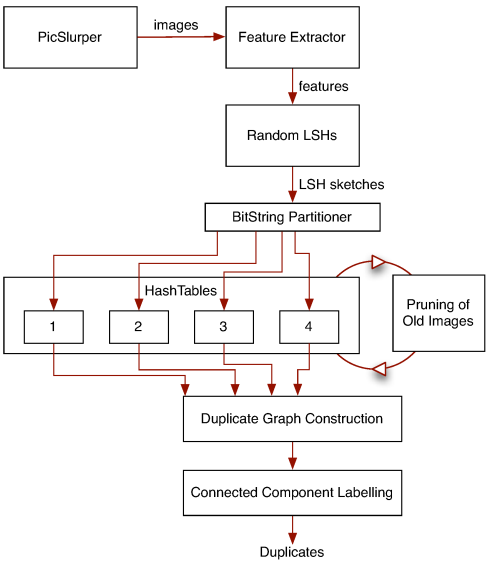

Figure 1: Flowchart illustrating the streaming duplicate detection algorithm.

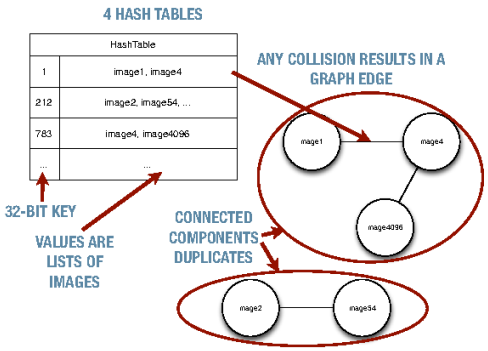

Figure 2: Illustration of the how the hash tables define the duplicates graph.

Figure 3: The trending image visualisation.

only interested in images within a time-window, so we have to continually remove old images as new ones arrive. In the approach, two images are defined as near-duplicates if they share a certain number of (local) features. To detect all near-duplicates we build an undirected weighted graph where the vertices are images and the edge weights represent the number of matching features between the images. Once the graph is constructed, edges with low-weight are pruned, and connected-component analysis is performed to extract all the sets of near-duplicates. The largest sets of duplicates are said to be trending and are emitted to the visualisation component of the system.

To efficiently assess whether features match, Locality Sensitive Hashing is used to create sketches (compact binary strings) from the features. The sketches are produced such that the Hamming distance between sketches approximates the Euclidean distance between the features [1]. As in [2], we choose our sketches to be 128 bits in length, and set the minimum Hamming distance for two sketches/features to be classed as matching at 3 bits. Rather than explicitly compute Hamming distances between all features, a more efficient (approximate) scheme is used: The 128-bit sketches are partitioned into 4 32-bit strings and represented as 32-bit integers. For a pair of matching sketches there could be at most 3 different bits, so at least one of the pairs of integers from the sketches must be the same. By using the four integers from each sketch as keys in four hash tables and storing the images containing the features as the values in the tables, the graph construction becomes trivial. As we are interested in temporal detection of duplicates, we regularly prune *old* images from the hash tables so that their respective contents only cover images from a fixed window into the past. The overall process we use is illustrated in Figure 1. Figure 2 illustrates the duplicate detection from the hash tables through construction of a graph.

The specifics of the image analysis and feature extraction up to the point of sketch construction are as follows: 1) images are resized to 150px on the longest side; 2) SIFT features detected at peaks in a difference-of-Gaussian pyramid [4] are extracted; 3) Low-entropy features are removed [2]; and finally, 4) log-scaling is applied to make the feature values more uniform [2].

2.3 Visualisation

Currently we have the system set up so that the PicSlurper & duplicate detection modules produce information about the trending images (and the Tweets in which they occur) in a file in JSON format. A visualisation written in HTML and JavaScript continually polls this file and displays the currently trending images. The visualisation is depicted in Figure 3. A video describing the system and showing the visualisation is online at `http://youtu.be/CBk5nDd6CLU`.

3. ACKNOWLEDGMENTS

The described work was funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreements 270239 (ARCOMEM), and 287863 (TrendMiner).

4. REFERENCES

[1] W. Dong, M. Charikar, and K. Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *SIGIR'08*, pages 123–130. ACM, 2008.

[2] W. Dong, Z. Wang, M. Charikar, and K. Li. High-confidence near-duplicate image detection. In *ACM ICMR'12*, pages 1:1–1:8. ACM, 2012.

[3] J. S. Hare, S. Samangooei, and D. P. Dupplaw. OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images. In *Proceedings of ACM Multimedia 2011*, MM '11, pages 691–694. ACM, 2011.

[4] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, January 2004.