Katy Perry and Trend Detection using

Matt Kirk – Wetpaint

JRuby

`whoami`

Former finance quant turned Ruby hacker

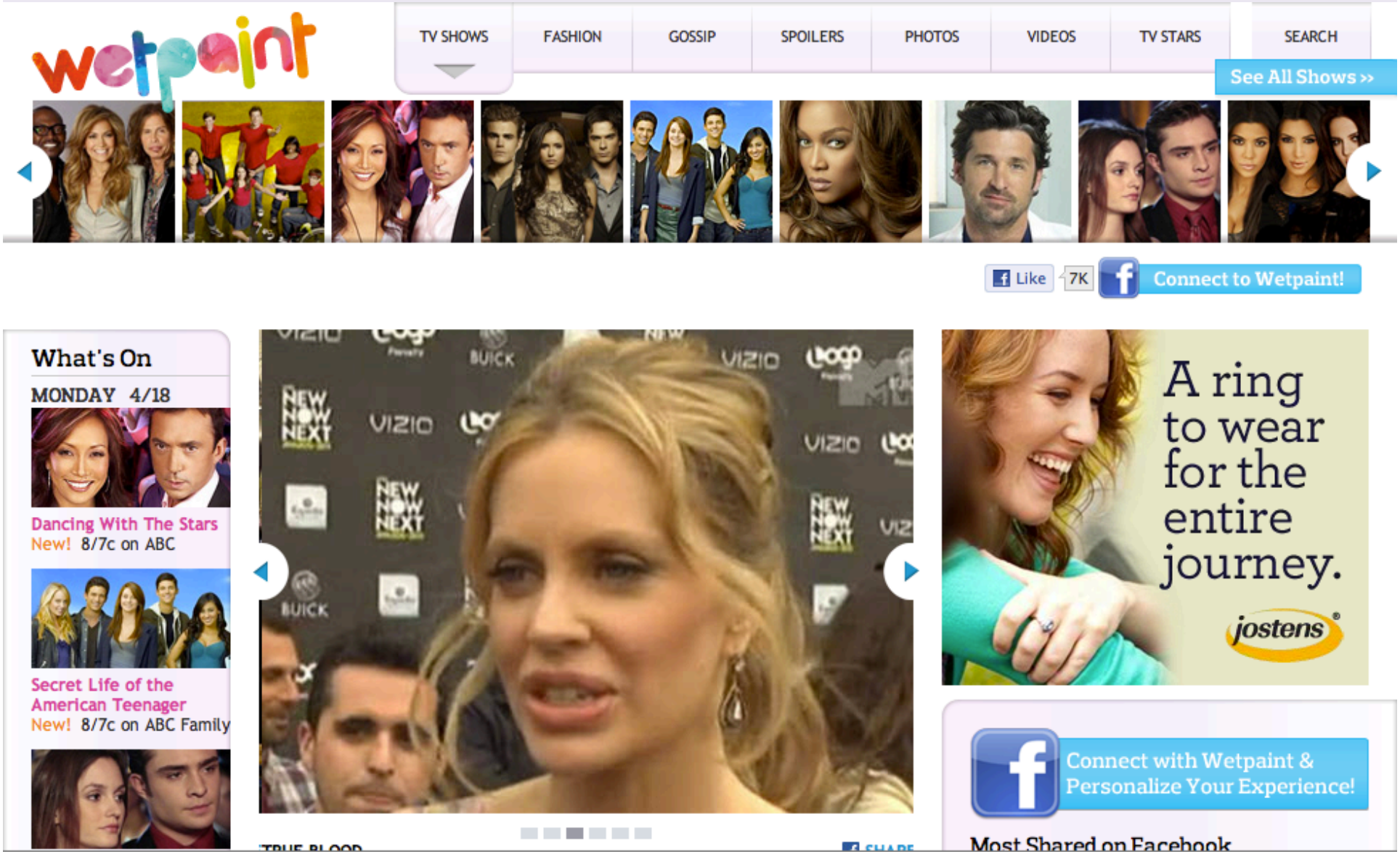
twitter: @mjkirk

website: matthewkirk.com

I work for wetpaint.com

Who the hell is Wetpaint?

wetpaint.com



WTF am I doing here at RedDirt??!!!

6 months ago we started building an engine

To find relevant news

That Utilizes

- Statistical processing
- Natural Language Processing
- Hardcore mathematics

And Magic

Ruby doesn’t really have tools for this sort of thing

Except for magic ☺

Java Packages of help

- Apache commons math
- Stanford CoreNLP
- OpenNLP
- Weka
- LingPipe
- Mahout
- etc, etc

Who wants to write Java all day long though?

Java + JRuby = Awesome

So what about Katy Perry….

JRuby helped us find Katy Perry

In the context of Glee

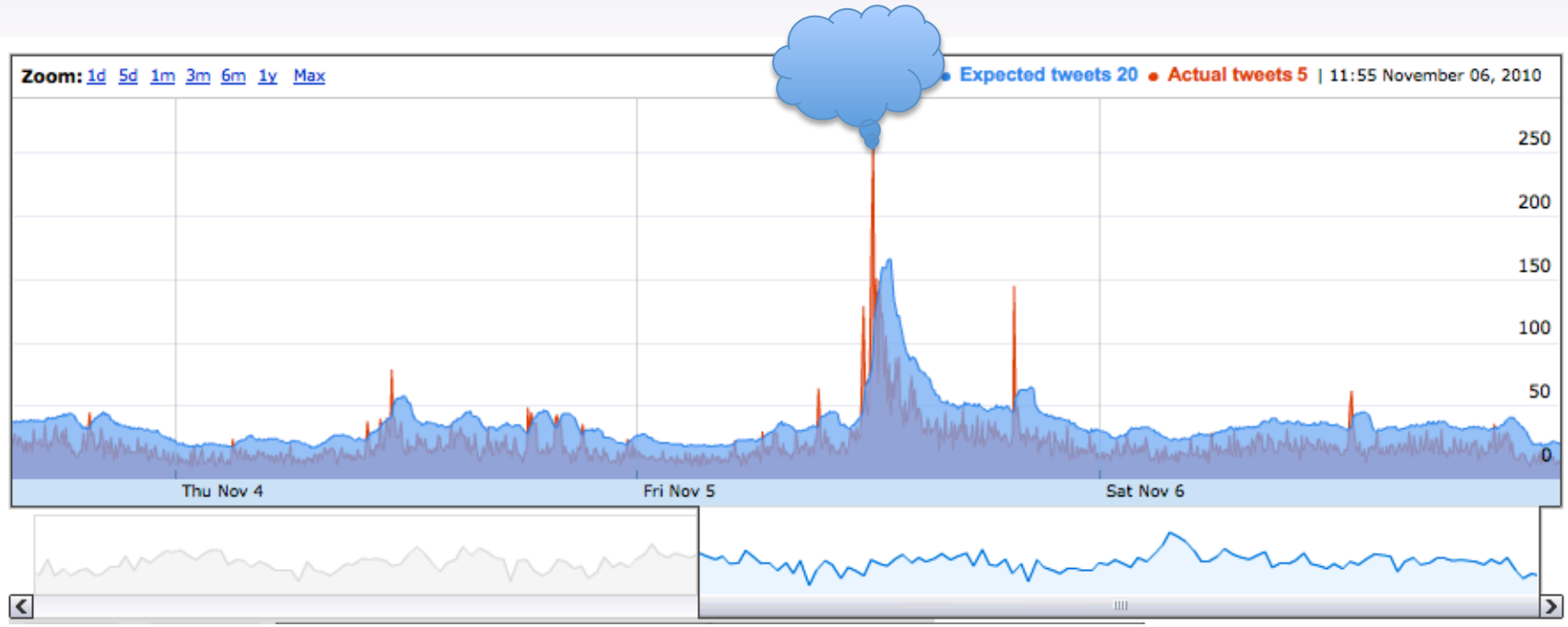
How we went about finding Katy Perry

1. Detect activity around shows on Twitter, Facebook etc.
2. Extract attributes about that activity
3. Cluster everything together to reduce clutter

Back in November she tweeted

Oh...My...Gosh... this just brought a sweet tear to my eye! Teenage Dream on GLEE makes my heart go WEEEEE! http://t.co/8SAFkGl

Which fed into a re-tweet frenzy



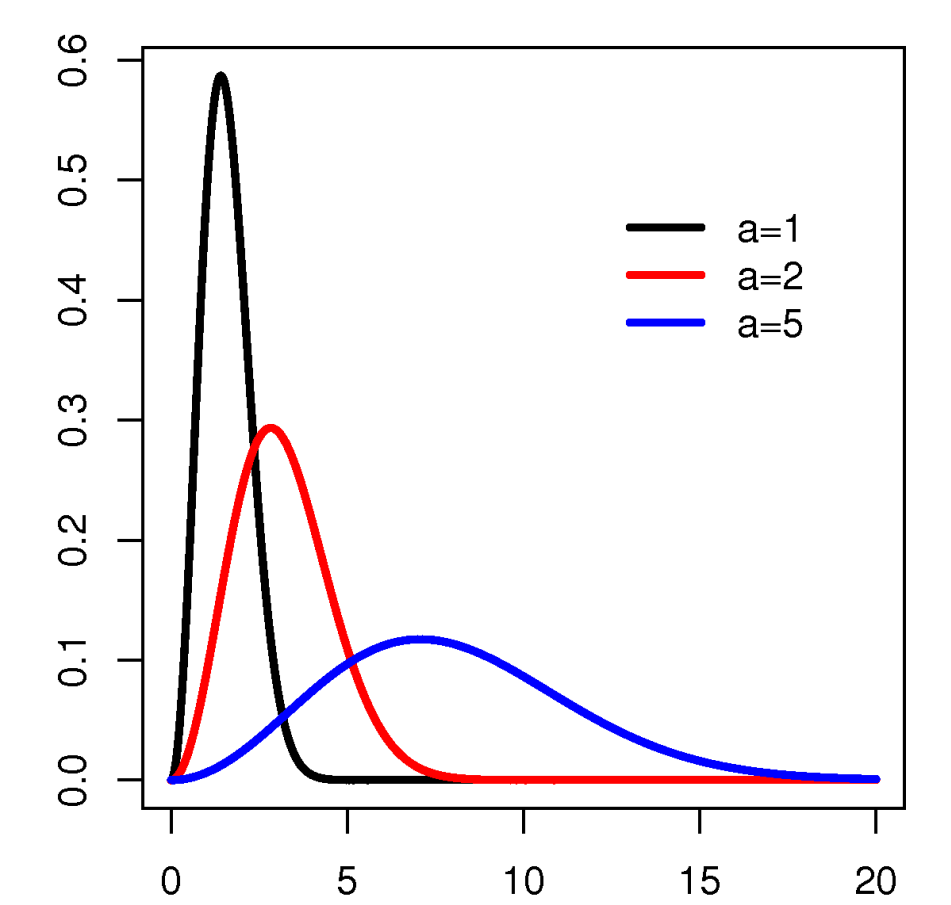
Obviously that’s an outlier

But how would we find that?

- Fit the Poisson distribution to the last day and figure out the percentile of the current data point.

- If it's greater than say 95% there's something weird going on

Poisson Distribution



Let’s use the Apache Commons Math Package!!!

```
require 'java'; require'math.jar'
Poisson =
  org.apache.commons.math.distribut
  ion.PoissonDistributionImpl

mean = 20 # tweets per five minutes

fishy = Poisson.new(mean)

fishy.cumulative_probability(30) #
  => 98.6%
```

Coding a Poisson distribution in ruby wouldn't be as much fun

Ok so we know there's something there. What is it?

Let's assume we don't know it was Katy Perry

Extract some attributes

- Using a tool like the Stanford CoreNLP

- Extract
  - n-gram phrases
  - words
  - urls

Probably get attributes like

```
n_grams = ["sweet tear", "teenage dream"]

words = ["oh", "gosh", "just", "brought",
  "sweet", "tear", "eye", "teenage",
  "dream", "glee", "heart", "go", "weeeee"]

urls = ["http://t.co/8SAFkGl"]
```

Stanford Core NLP

```
require 'java'; require 'nlp.jar'
include_class
  "edu.stanford.nlp.ie.machinereading.domain
  s.ace.reader.RobustTokenizer"
# Seriously wtf guys...

RT = RobustTokenizer
```

Stanford Core NLP

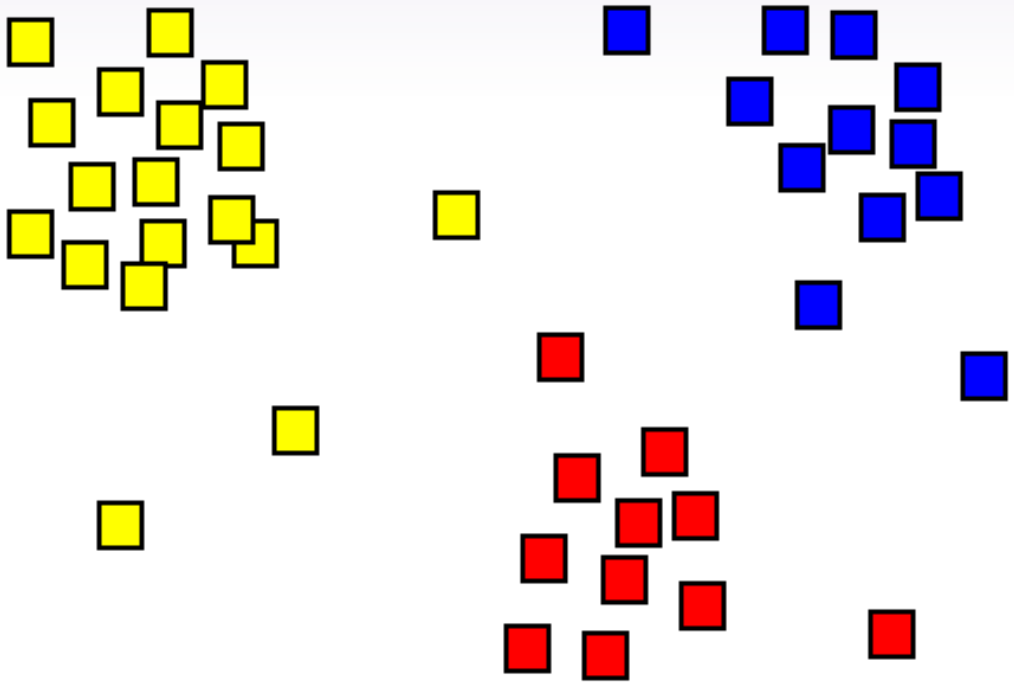
```
tok = RT.new(katy_perry_tweet)

tokens = tok.tokenize.map(&:to_s)
urls = tokens.select do |t|
 RT.is_url(t)
end

words = tokens.uniq - urls - punctuation -
  stopwords
```

We have a bag full of words and urls. Now what?

Cluster it!

- Little more of a difficult problem. In our case we wrote our own package.

- Apache Commons math and Weka both have k-means clustering in them



Quickest solution is to use Apache Commons Math

Build a Point Class First

```
class Point
 attr_reader :attrs
 include
  org.apache.commons.math.stat.clustering.Clust
  erable
 def initialize(attrs = [])
  @attrs = attrs
 end
 def distanceFrom(point)
 ((point.attrs | @attrs) - (point.attrs &
  @attrs)).length
 end
```

Build a Point Class First

```
def centroidOf(points)
    u = Point.new(points.map
     (&:attributes).flatten.uniq)
    guess = points.first
    points.each do |point|
     if u.distanceFrom(point) < u.distanceFrom
     (best_guess)
      guess = point
     end
    end
    best_guess
 end
end
```

Feed it into Apache Commons

```
require 'java'; require 'math.jar'
include_class
  "org.apache.commons.math.stat.clustering.K
  MeansPlusPlusClusterer"
clusterer = KMeansPlusPlusClusterer
  (java.util.Random.new)
num_clusters = ? # Depends...
max_iter = -1 # no max
clusterer.cluster(collection_of_points,
  num_clusters, max_iter)
```

Now that we've clustered, our peeps can find Katy Perry

And now we can have a party

Conclusion

- Detect
- Extract
- Cluster

Conclusion

- Java + JRuby = Killer combo for fun development of statistics apps.

THANKS!!!

If you want to work with Females 18-34. We're Hiring!!!

http://bit.ly/wpdevs