Aggregating and Visualizing Microblogs for Event Exploration

System

Create An Event

Define a event

- Specify a few **keywords** for Twitter query
- Give it a **human-readable name**
- [Optional] give it a time window

Start tracking

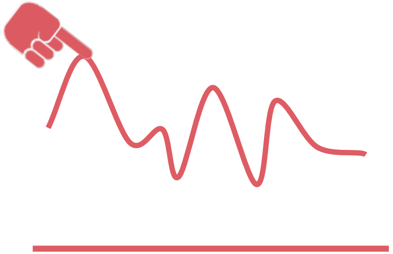
- **Save** the event
- Begin **logging** tweets matching the query

- Only track tweets for a keyword **after** it's enteted
- Possible solution: collect a sample of all tweets, and **historically index** each keyword as users begin tracking them

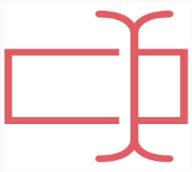
System

Creating Subevents

Pick a peak from the timeline



Give it a human-readable name



- A subevent can be **zoomed into** form another
- e.g. Zoom into a speech from an election

System

Realtime Updating

Refresh at regular intervals

Render real-time tracking possible

Algorithms

Event Dectectation

Preparation

- **Count** related tweets in each interval

Finding peaks and hills

- Calculate the **average tweet rate**
- Find **unusually high** rates
- Find **peaks** and **hills**

Labeling peaks

- **Collect** related tweets in each interval
- Select **frequent terms** as labels

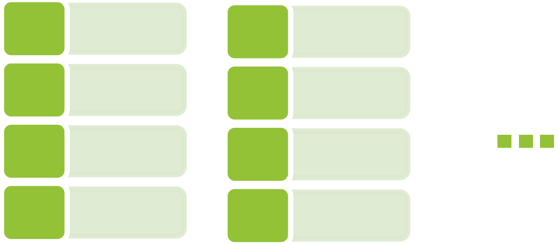
Algorithms

Event Dectectation

Preparation

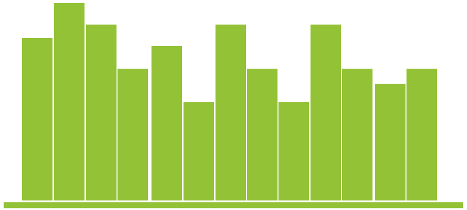
IN

Time-sorted **tweets**
containing given keywords



OUT

List of tweet **counts**



$[C_1, .. C_N]$

* interval size is adjustable

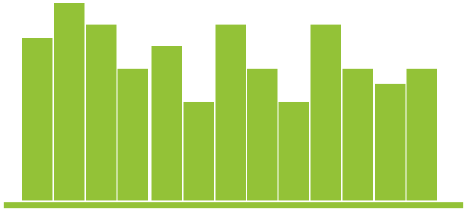
Algorithms

Event Dectectation

Finding peaks and hills

IN

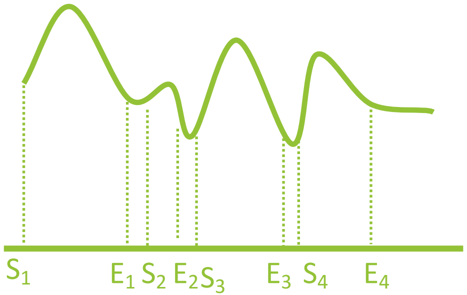
List of tweet **counts**



$[C_1, .. C_N]$

OUT

List of starting and ending points of **windows**



$[(S_1, E_1), ... (S_N, E_N)]$

Event Dectectation

Offline Algorithm

Finding peaks and hills

```
function find_peak_windows(C):
  windows = []
  mean = C1
  meandev = variance(C1, ..., Cp)
```

Counts of tweets

Event Dectectation

Offline Algorithm

Finding peaks and hills

```
function find_peak_windows(C):
  windows = []
  mean = C₁
  meandev = variance(C₁, ..., Cₚ)
```

Starting and ending points of hills

Event Dectectation

Offline Algorithm

Finding peaks and hills

```
function find_peak_windows(C):
  windows = []
  mean = C₁
  meandev = variance(C₁, ..., Cₚ)
```

Mean of tweet rate

Algorithms

Event Dectectation Offline Algorithm

Finding peaks and hills

```
function find_peak_windows(C):
  windows = []
  mean = C₁
  meandev = variance(C₁, ..., Cₚ)
```

Mean deviation of tweet rate

* Initialized to first *p* counts' mean deviation

- Why not standard deviation?
- Mean deviation **doesn't need** historical counts

Algorithms

Event Dectectation

Offline Algorithm

Finding peaks and hills

Iterate through the counts.

```
for i = 2; i < len(C); i++ do
```

$$\text{if } \frac{|C_i - mean|}{meandev} > \tau \text{ and } C_i > C_{i-1} \text{ then}$$

```
    start = i - 1
    while i <len(C) and Ci> Ci-1 do
      (mean, meandev) = update(mean, meandev, Ci)
      i++
    end while
  ...
```

Algorithms

Event Dectectation

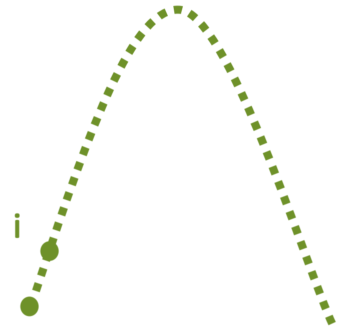
Offline Algorithm

Finding peaks and hills

```
for i = 2; i < len(C); i++ do
   if |Ci−mean| / meandev > τ and Ci > Ci-1 then
     start = i - 1
     while i <len(C) and Ci > Ci-1 do
        (mean, meandev) = update(mean, meandev, Ci)
        i++
     end while
  ...
```

If C_i is unusually high* and C is climbing, mark a starting point of the hill.

* Chauvenet's criterion



Algorithms

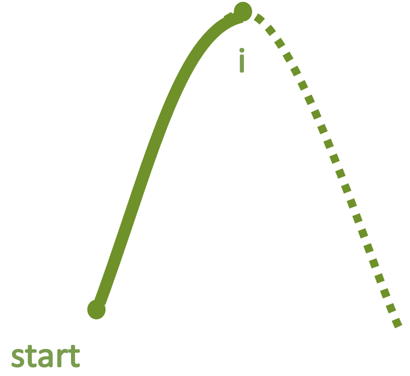
Event Dectectation

Offline Algorithm

Finding peaks and hills

```
for i = 2; i < len(C); i++ do
  if |Ci−mean| / meandev > τ and Ci > Ci-1 then
    start = i − 1
    while i <len(C) and Ci > Ci-1 do
      (mean, meandev) = update(mean, meandev, Ci)
      i++
    end while
 ...
```

Detect the peak (increment i until C_i starts dropping).



Event Dectectation

Offline Algorithm

Finding peaks and hills

```
for i = 2; i < len(C); i++ do
  if |Ci−mean| / meandev > τ and Ci > Ci-1 then
    start = i − 1
    while i <len(C) and Ci > Ci-1 do
      (mean, meandev) = update(mean, meandev, Ci)
      i++
    end while
  ...
```

Whenever i is incremented, update the mean and mean deviation as well.

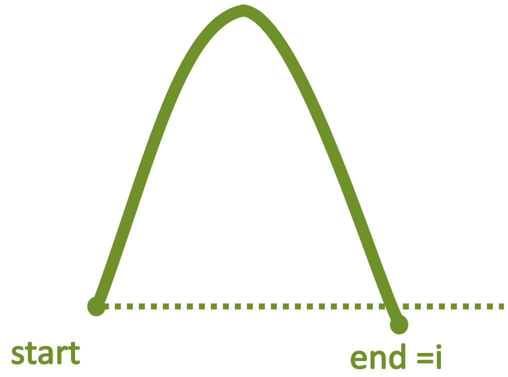
Algorithms

Event Dectectation Offline Algorithm

Finding peaks and hills

```
...
while i < len(C) and Ci> Cstart do
  if |Ci−mean| / meandev > τ and Ci> Ci−1 then
    end = --i
    break
  else
    (mean, meandev) = update(mean, meandev, Ci)
    end = i++
  end if
end while
...
```

Detect the end (increment i until C_i drops under C_{start}).



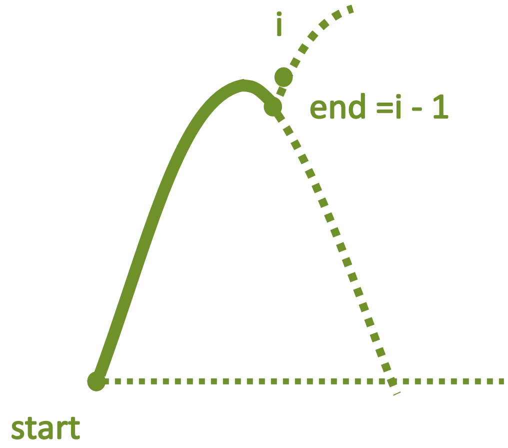
Event Dectectation

Offline Algorithm

Finding peaks and hills

```
...
while i < len(C) and Ci> Cstart do
  if |Ci−mean| / meandev > τ and Ci> Ci-1 then
    end = --i
    break
  else
    (mean, meandev) = update(mean, meandev, Ci)
    end = i++
  end if
end while
...
```

If C_i is unusually high and C starts climbing again, mark it as an end.



Event Dectectation

Offline Algorithm

Finding peaks and hills

```
...
while i < len(C) and Ci > Cstart do
  if |Ci−mean| / meandev > τ and Ci > Ci-1 then
    end = --i
    break
  else
    (mean, meandev) = update(mean, meandev, Ci)
    end = i++
  end if
end while
...
```

Whenever i is incremented, update the mean and mean deviation as well.

Event Dectectation

Offline Algorithm

Finding peaks and hills

```
for i = 2; i < len(C); i++ do
  if |Ci−mean| / meandev > τ and Ci > Ci-1 then
    ...
    windows.append(start, end)
  else
    (mean, meandev) = update(mean, meandev, Ci)
  end if
end for

return windows
```

After the end is dectected, add the newly found hill to `windows`

Algorithms

Event Dectectation

Offline Algorithm

Finding peaks and hills

```
for i = 2; i < len(C); i++ do
  if |Ci−mean| / meandev > τ and Ci > Ci-1 then
    ...
    windows.append(start, end)
  else
    (mean, meandev) = update(mean, meandev, Ci)
  end if
end for

return windows
```

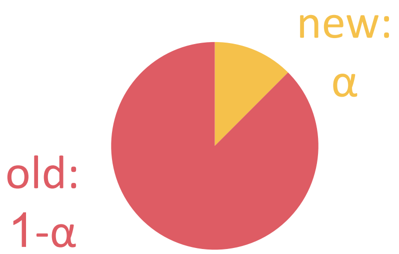
If the rate is not particularly high or it’s dropping, update the mean and mean devivation, then continue searching for peaks.

Event Dectectation

Offline Algorithm

Finding peaks and hills

```
function update(oldmean, oldmeandev, updatevalue):
  diff = |oldmean – updatevalue|
  newmeandev = α*diff + (1–α)*oldmeandev
  newmean = α*updatevalue + (1–α)*oldmean
  return (newmean, newmeandev)
```



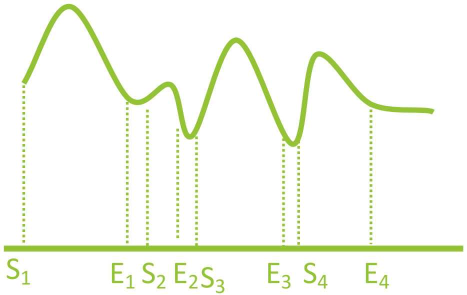
- Weighted exponentially, old data will eventually **lose its influence**. (TCP congestion control)
- Works well with stream.
- $\alpha = 0.125$ is a reasonable value.

Event Dectectation

Labeling peaks

IN

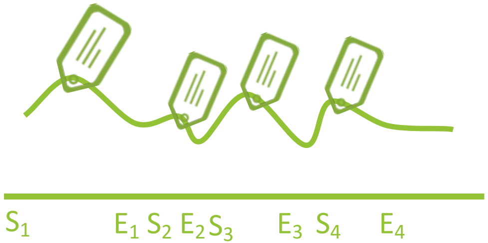
List of starting and ending points of **windows**



$[(S_1,\ E_1),\ ...\ (S_N,\ E_N)]$

OUT

Labels for each windows



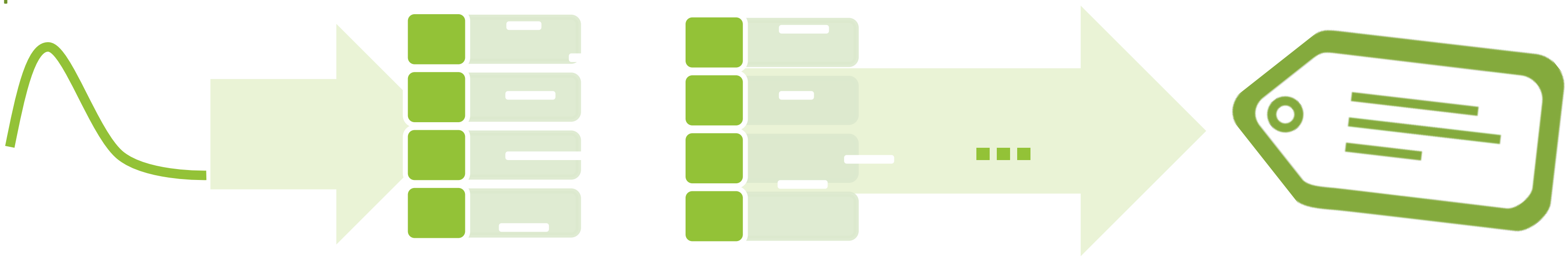
Algorithms

Event Dectectation

Labeling peaks

2. **Select** frequent terms
   in those tweets.

1. **Collect** relevant tweets
   posted in the window.



Algorithms

Event Dectectation

How to Select the frequent terms?

Labeling peaks

1. Tokenize the tweets into unigrams (i.e. context doesn't matter).

> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

2. Rank the unigrams by TF-IDF.

- Lorem 0.0058
- ipsum 0.0034
- dolor 0.0215
- sit 0.0023
- amet, 0.0023
- con 0.0052
- ....

$$\text{TF} = \frac{\text{Frequency of } \textbf{the term} \text{ in W}}{\text{Number of } \textbf{all words} \text{ in W}}$$

$$\text{IDF} = \log_2\left(\frac{\text{Number of } \textbf{all tweets} \text{ in all windows}}{\text{Number of tweets } \textbf{containing the term} + 1}\right)$$

High TF: Frequently mentioned in W

High IDF: Not mentioned much in daily tweets

TF-IDF = TF × IDF

Algorithms

Event Dectectation

How to Select the frequent terms?

Labeling peaks

1. Tokenize the tweets into unigrams (i.e. context doesn't matter).

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam

2. Rank the unigrams by TF-IDF.

Lorem	0.0058
ipsum	0.0034
dolor	0.0215
sit	0.0023
amet,	0.0023
con	0.0052
....	...

- sed
- do
- eiusm
- tempor
- incididunt
- ut
- ...

3. Present the top 5 as labels.

- sed
- do
- eisum
- tempor
- incididunt

Removing Noisy Query Terms

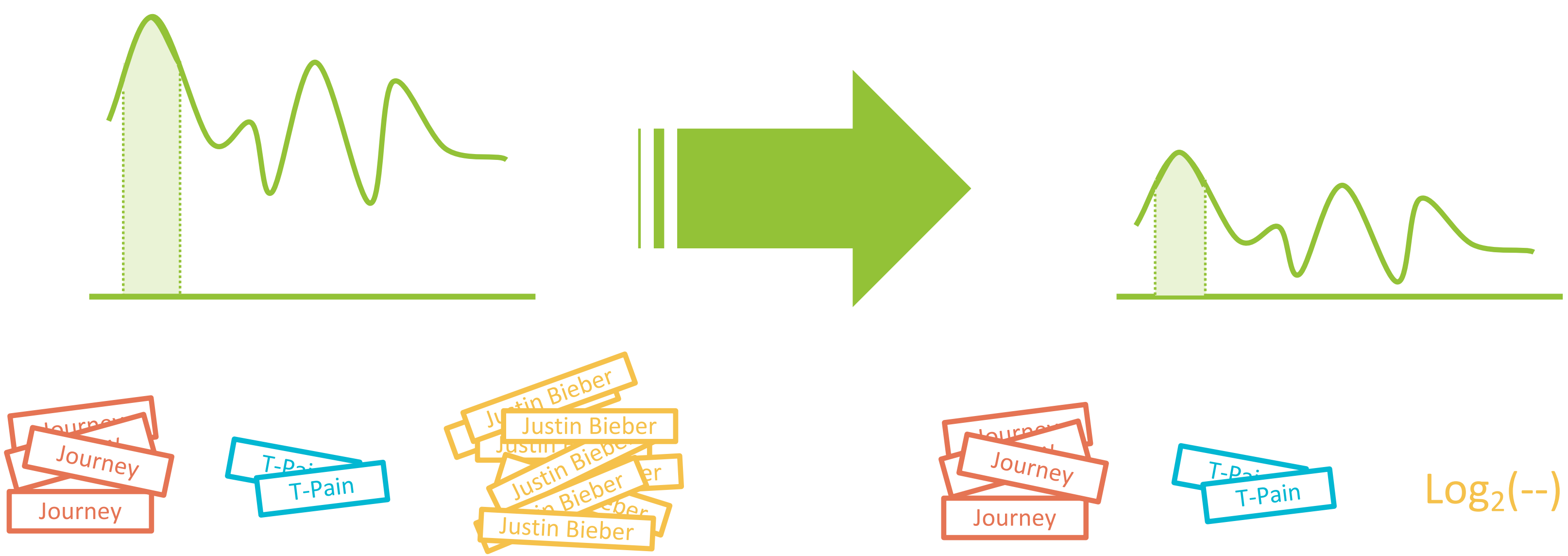
For always-popular terms:

C = global IDF =

$$\log_2\left(\frac{\text{Number of tweets in all windows}}{\text{Number of tweets containing the term} + 1}\right)$$

Removing Noisy Query Terms

BEFORE **AFTER**

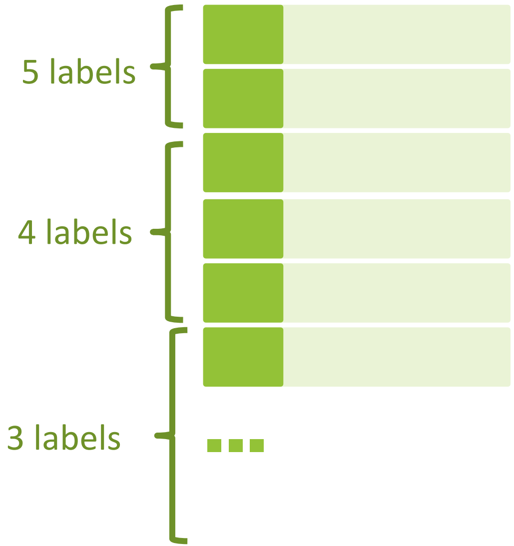


C_i = C(Journey) + C(T-Pain) + C(Justin Bieber)

C_i = C(Journey) + C(T-Pain) + IDF(Justin Bieber)

Identifying Relevant Tweets

- Rank tweets by **number of labels** they contains (e.g. by the sum of TF-IDF)
- Roughly equivalent to **searching** related tweets in W with the **labels as keywords**



Representing Aggregate Sentiment

- Classify tweets into positive and negative classes
- **Naive Bayes classifier** trained on **unigram** features

Algorithms

Representing Aggregate Sentiment

- predicts p_p and p_n for each tweet

p_p : P(belongs to the positive class)

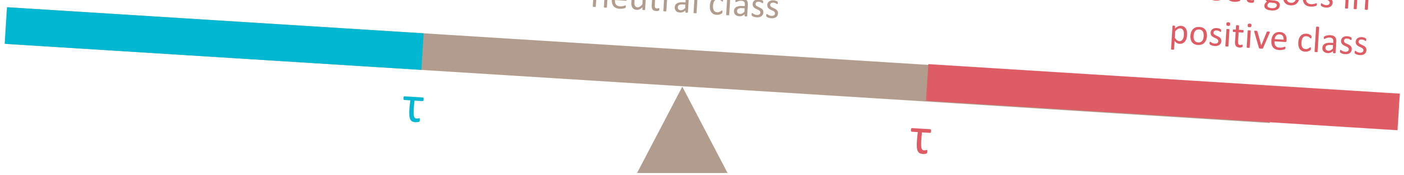
p_n : P(belongs to the negative class)

τ : confidence threshold

$p_n > \tau$:
this tweet goes in negative class

$p_n < \tau$, $p_p < \tau$:
this tweet goes in neutral class

$p_p > \tau$:
this tweet goes in positive class



Representing Aggregate Sentiment

The Naive Bayes Classifier

The probability that it is a positive tweet:

Well done Andros Townsend. Amazing performance!

```
P(positive|“well",        =      P(positive)
           "done",              * P(“well”|positive)
           “andros",            * P(“done"|positive)
           “townsend",          * P(“andros”|positive)
           “amazing",           * P(“townsend”|positive)
           "performance")       * P(“amazing”|positive)
                                * P(“performance”|positive)
```

Algorithms

Representing Aggregate Sentiment

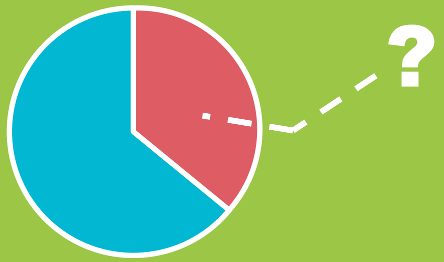
The Naive Bayes Classifier

The probability that it is a positive tweet:

Well done Andros Townsend. Amazing performance!

P(positive)

Proportion of tweets that belongs to the positive class



P(positive)
* P(“well”|positive)
* P(“done"|positive)
* P(“andros”|positive)
* P(“townsend”|positive)
* P(“amazing”|positive)
* P(“performance”|positive)

Representing Aggregate Sentiment

The Naive Bayes Classifier

The probability that it is a positive tweet:

Well done Andros Townsend. Amazing performance!

P(“well”|positive)

Probability that in a positive tweet, the word "Well" shows up

...	welfare	well	west	...

?

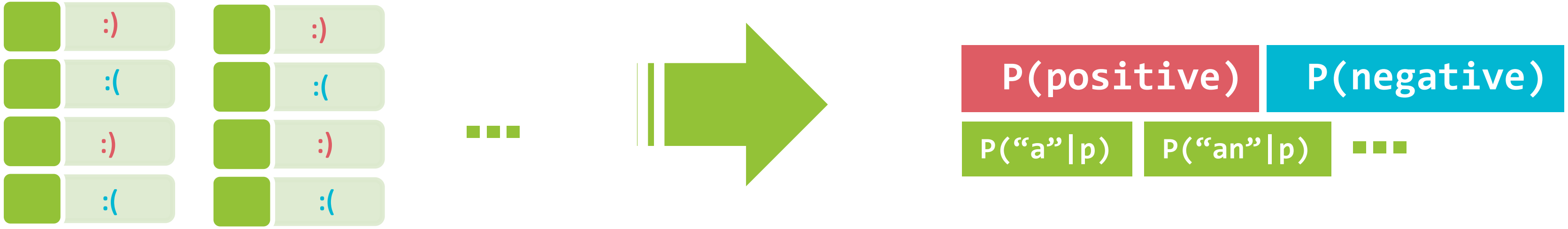
P(positive)
* P(“well”|positive)
* P(“done"|positive)
* P(“andros”|positive)
* P(“townsend”|positive)
* P(“amazing”|positive)
* P(“performance”|positive)

Representing Aggregate Sentiment

- Classify tweets into positive and negative classes
- **Naive Bayes classifier** trained on **unigram** features
- Training sets: use tweets with happy and sad emoticons.

:) :D ^_^

:(T_T Q_Q



Tweets with emoticons

prior probabilities

Algorithms

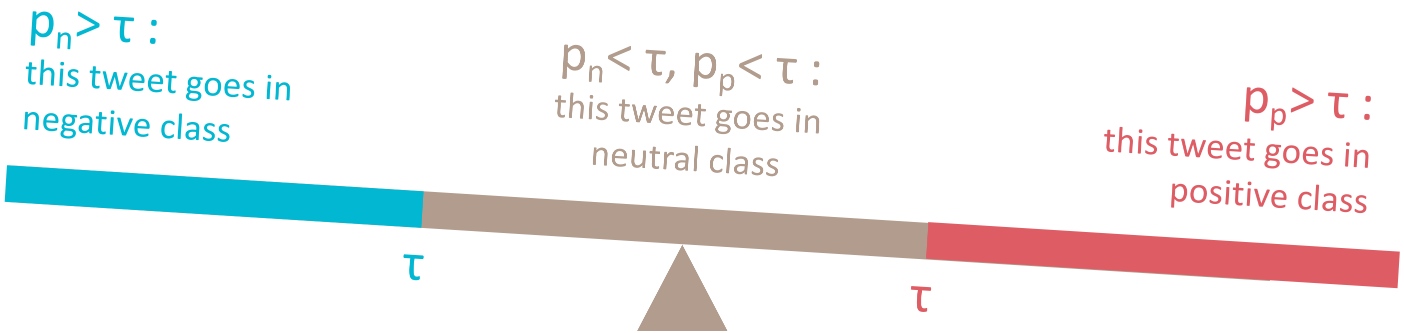
Representing Aggregate Sentiment

- predicts p_p and p_n for each tweet
- After good training, a larger τ can increase precision but also **decrease** recall

p_p : P(belongs to the positive class)

p_n : P(belongs to the negative class)

τ : confidence threshold



Representing Aggregate Sentiment

Problem: **different recall** values at the **same precision**.

i.e. One classifier is conservatively **ignoring** tweets while the other is **accepting** them

recall =

precision =

False Positive	False Negative
True Positive	True Negative

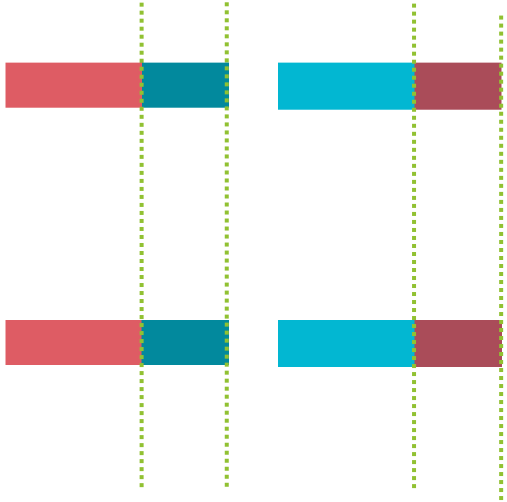
Representing Aggregate Sentiment

Solution: *recall-normalization*

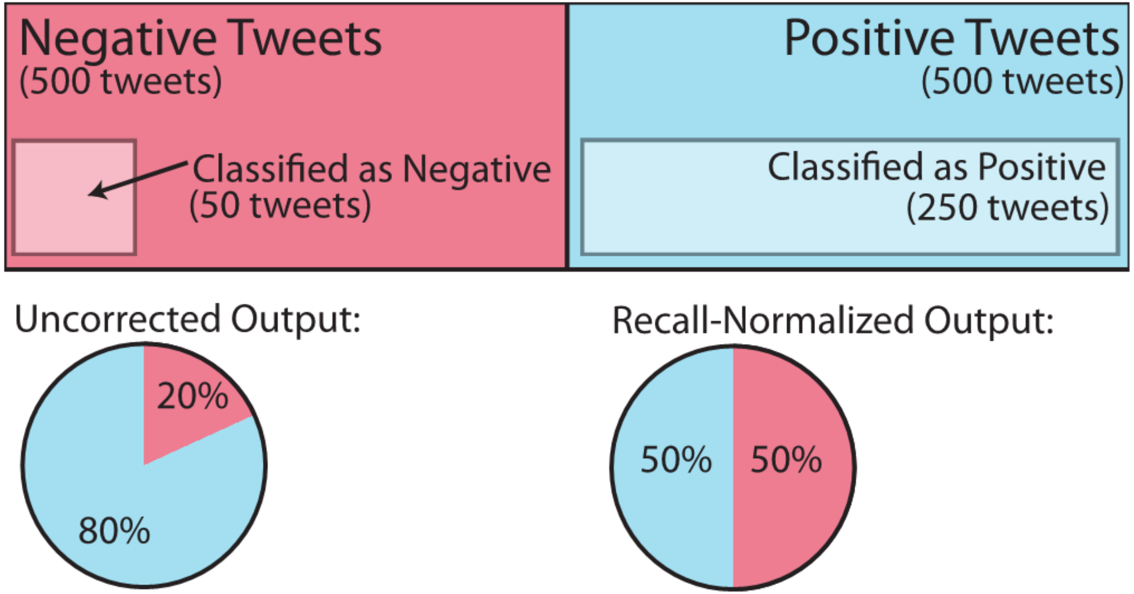
1. **Adjust** classifiers so that they have the same precision on a test dataset

2. **Measure** recalls for the test dataset

3. When using the classifier, **divide** positive/negative tweets counts by the positive/negative recall



Representing Aggregate Sentiment



Implementation

What Does TwitInfo Use

Getting Data & Metadata
Twitter Streaming API

Web framework
Django

Locating & Generating Maps
Google Maps API

Visualizing Data
Google Visualization API

Tweets are indexed
by keywords

Evaluation

Algorithm Evaluation

Test cases

3 soccer games

1 month of earthquakes

Ground truths

- **Game video**
- **Web-based summaries**

- **US Geological Survey**

Algorithm Evaluation

Object of Evaluation

Precision

How many events detected were part of the ground truth set?

Recall

How many events in the ground truth set were detected?

* default threshold cutoff is used

Algorithm Evaluation

Results

Precision

- **Depended** on activity type.
- Fails when there were **minor events** or general discussions **after** the event happened.

Recall

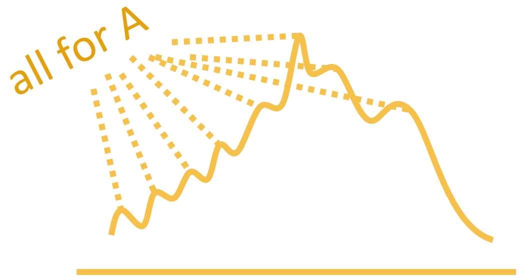
- **High**, all major events are detected.
- Fails when the Twitter volume **didn’t peak** during the event.

Biased by Twitter’s interests

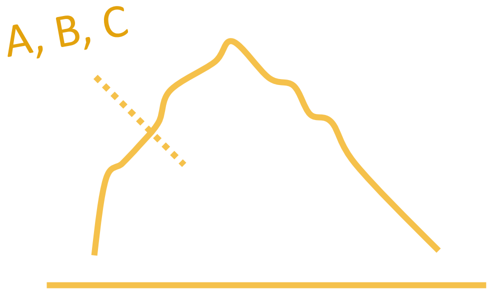
Algorithm Evaluation

Two artifacts

Multiple peaks for one event



Multiple events overlap on one peak



Evaluation

User Interface Evaluation

Subject

12 participants

All can explain Twitter’s mechanics

Evaluation

User Interface Evaluation

Subject

6 had Twitter accounts

10 had visited Twitter before

4 were female

User Interface Evaluation

Method

Directed search tasks

EG: When and where did the earthquake occur, and what was the magnitude?

Time-limited exploration

EG: Find out how Obama got selected as the president in 5 minutes.

Semi-structured interview

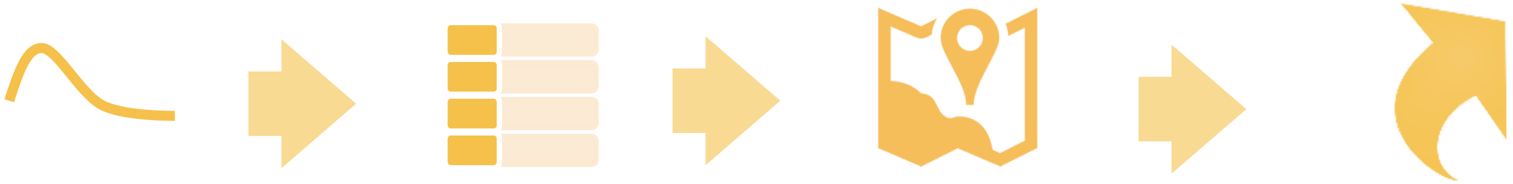
EG: What are the best and worst parts of the TwitInfo interface?

User Interface Evaluation

Results

- It can give users a **quick, high-level** understanding of the event.
- Might be a little “**shallow**”

Common Usage Patterns



Picked the largest peak

Read the tweets

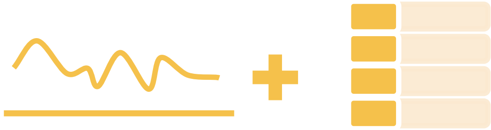
Drilled in on the map

Followed links

Evaluation

User Interface Evaluation

Common Usage Patterns

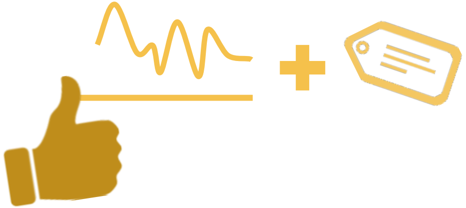


Free Tasks

Time-limited Tasks

Details

The Timeline Focused User Activity



Mapping Requires Aggregation

- Sth. like a heatmap?
- Bounds and zoom levels of the map to act as a filter

Evaluation

User Interface Evaluation

Users Do Not Trust Sentiment Analysis

The sentiment classifiers
worked correctly.

Human sentiments are
more subtle than just **“positive”**
and “negative”.

Sentiments **in tweets**
might not be
sentiments **about the topic**

What Alternatives Did Users Suggest?

Online articles,
Google News

Traditional media

Hard to decide what to
pay attention to.

User Interface Evaluation

A Journalist's Perspective

Timeline & labels

Useful for **backgrounding** on a longer-running topic.

Add stories from **traditional** sources of news.

Helpful to decide which **substories** to explore further.

Topic-based **drill-down** interface along the lines.

Evaluation

User Interface Evaluation

A Journalist's Perspective

Map

Helpful to find **on-the-ground eyewitnesses** to follow up with.

Sentiment

Skeptical of the **quality** and **accuracy** of the algorithm as well as **the sample population**

Discussion & Conclusion

Achievements

Real-time

High level

Opens The Door To...

Interfaces for social computing systems

Event-based **Notification**

Large Text Corpus **Summarization**

Smarter **Trend** Detection

Limitations

First-use

How someone might use it longitudinally?

Low external validity

Can it be used for real investigation?

Sentiment Aggregation

Needs Improvement

Twitter As A Source

- Somtimes too **shallow**
- Only for **quick** reactions & information

THANKS