Mining user activity data in social media services

Alceu Ferraz Costa
Supervisor: Prof. Dr. Agma Juci Machado Traina
Co-supervisor: Prof. Dr. Christos Faloutsos

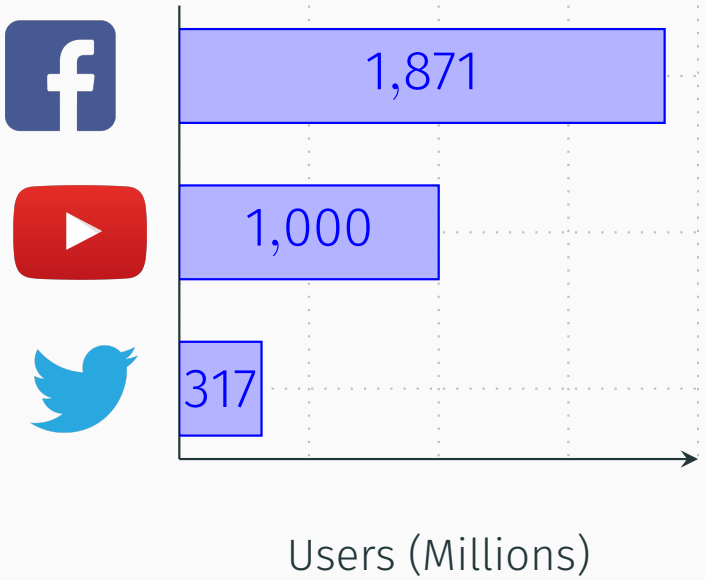
PhD Thesis Defense, ICMC – USP, May 2017

Outline

Introduction

Social media services:

- Creation and exchange of of user-generated content
- **Highly popular**
- Impact: news consumption, advertisement, communication...



Social media popularity $\rightarrow$ large volume of data on how users behave

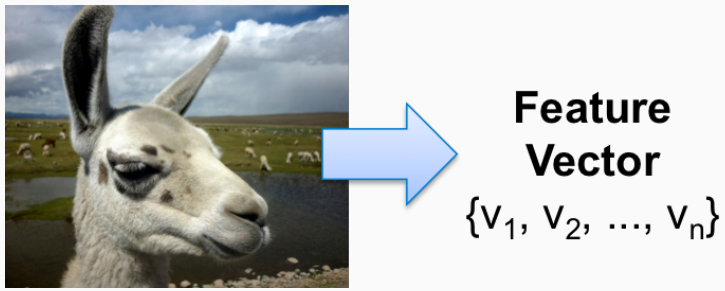
- Many potential applications (e.g. design of highly accurate fraud detection methods)

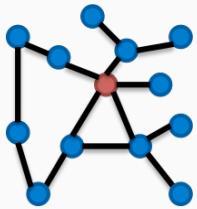
Introduction

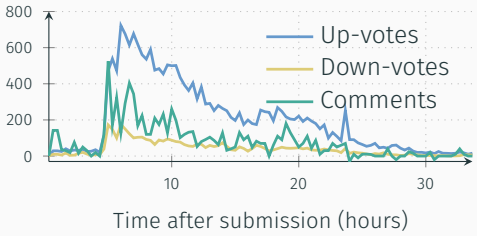
Research goal: How can we extract useful knowledge from social media data?

Challenges: social media data is...

1. **Complex:** objects are represented by a large number of attributes
2. **Diverse:** can be represented by images, time-series, graphs, textual information







Introduction

Thesis

Data from social media services obey patterns that can be explored to design specialized methods for mining the raw data and improve the accuracy of prediction, anomaly detection and forecasting tasks when compared to traditional non-specialized data mining methods.

Introduction

Research problems:

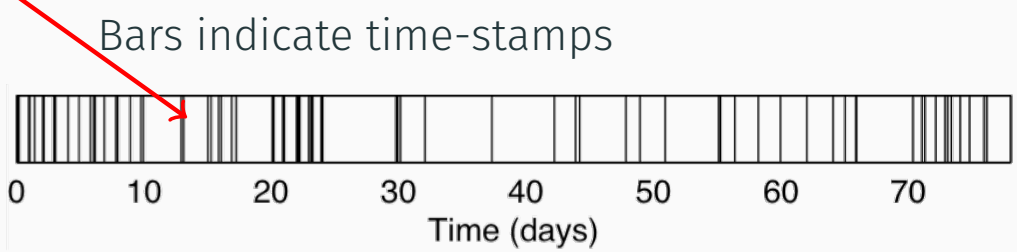
1. Temporal analysis users' actions
   - What patterns can be discovered from the time-stamps of users' activities?
   - Is it possible to use this data to detect anomalous behavior?
2. Modeling how the volume of users' interactions evolves over time
   - If a picture receive 2,000 "likes", how many comments is it expected to receive?
3. Automatically annotating social media images
   - Can we search for social images even when textual metadata is missing?

Outline

Act-M Model: Problem Statement

Users generate sequences of time-stamps when they use a social media Web site:

Sequence of tweets time-stamps:



What can we learn from time-stamps?

- Are there common patterns?
- Can we tell if a user is a bot or a human?

Act-M Model: Pattern Mining – Datasets

Reddit

- 21,198 users

Twitter

- 6,790 users

Stack-Overflow

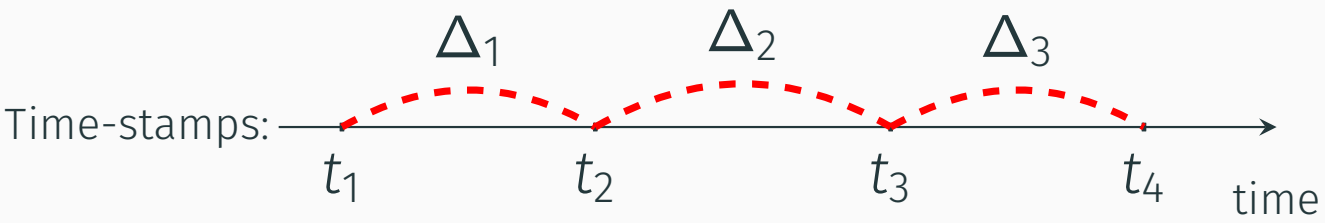
- 8,755 users

Hacker-News

- 1,368 users

For each user we have:

- Sequence of postings time-stamps: $T = (t_1, t_2, t_3, \cdots)$
- Inter-arrival times (IAT) of postings: $\boldsymbol{\Delta} = (\Delta_1, \Delta_2, \Delta_3, \cdots)$

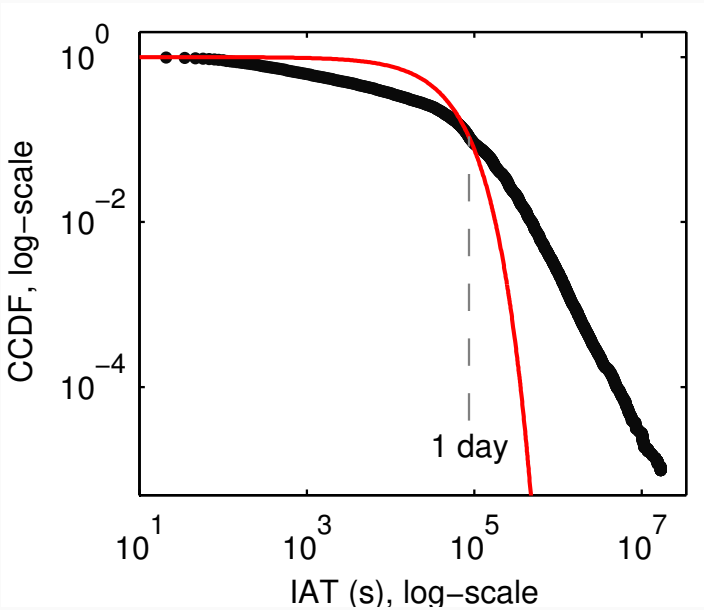


Act-M Model: Pattern Mining – Datasets

Pattern 1: The distribution of IAT is heavy-tailed

Users can be inactive for long periods of time before making new postings

IAT Complementary Cumulative Distribution Function (CCDF) (log-log axis)



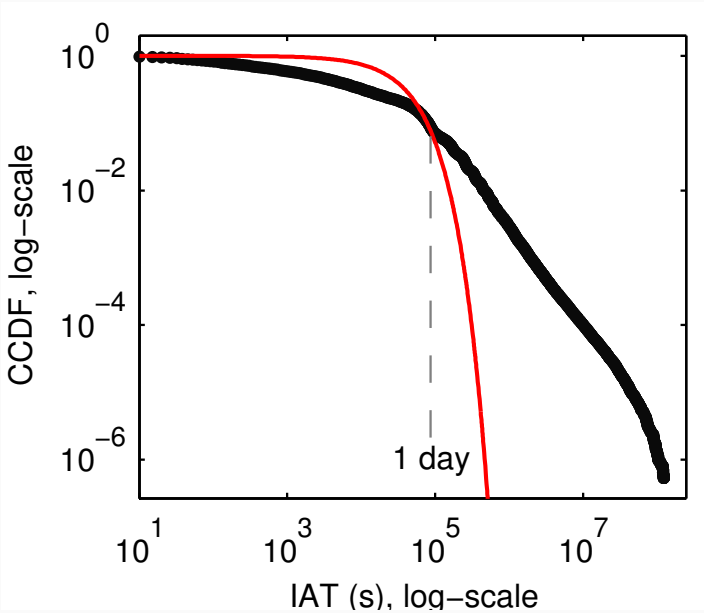
- Reddit Dataset

Act-M Model: Pattern Mining – Datasets

Pattern 1: The distribution of IAT is heavy-tailed

Users can be inactive for long periods of time before making new postings

IAT Complementary Cumulative Distribution Function (CCDF) (log-log axis)



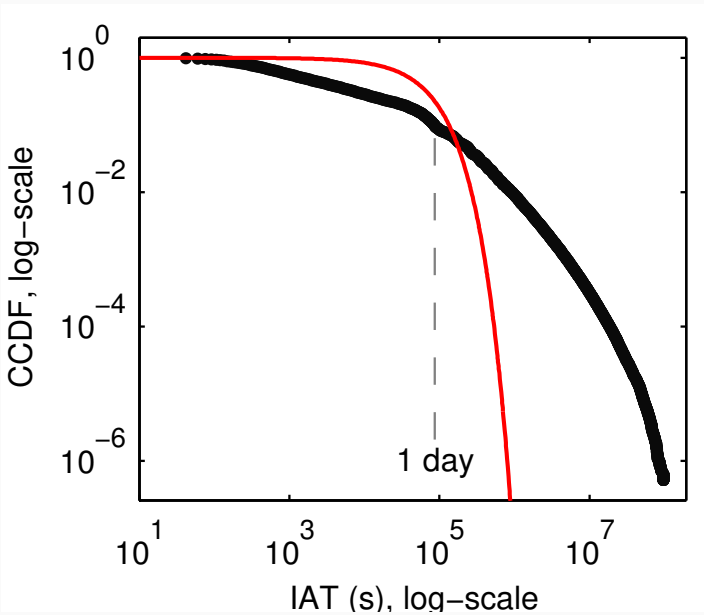
- Reddit Dataset
- Twitter Dataset

Act-M Model: Pattern Mining – Datasets

Pattern 1: The distribution of IAT is heavy-tailed

Users can be inactive for long periods of time before making new postings

IAT Complementary Cumulative Distribution Function (CCDF) (log-log axis)



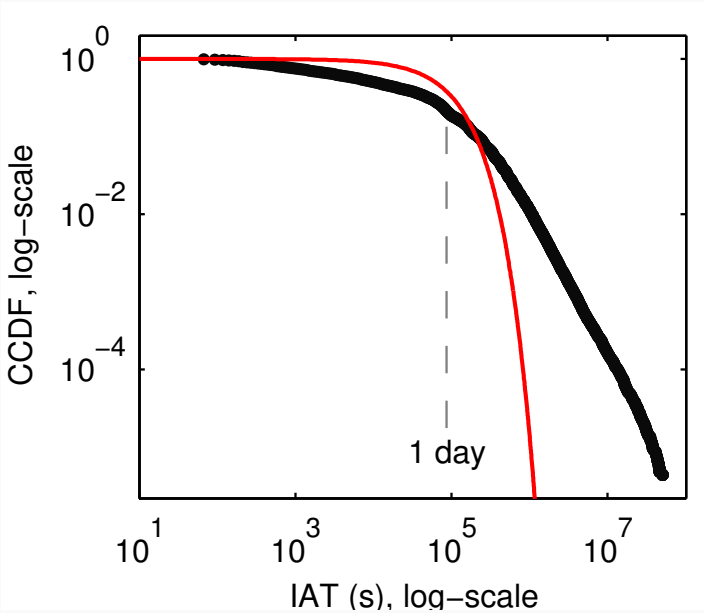
- Reddit Dataset
- Twitter Dataset
- Stack-Overflow Dataset

Act-M Model: Pattern Mining – Datasets

Pattern 1: The distribution of IAT is heavy-tailed

Users can be inactive for long periods of time before making new postings

IAT Complementary Cumulative Distribution Function (CCDF) (log-log axis)

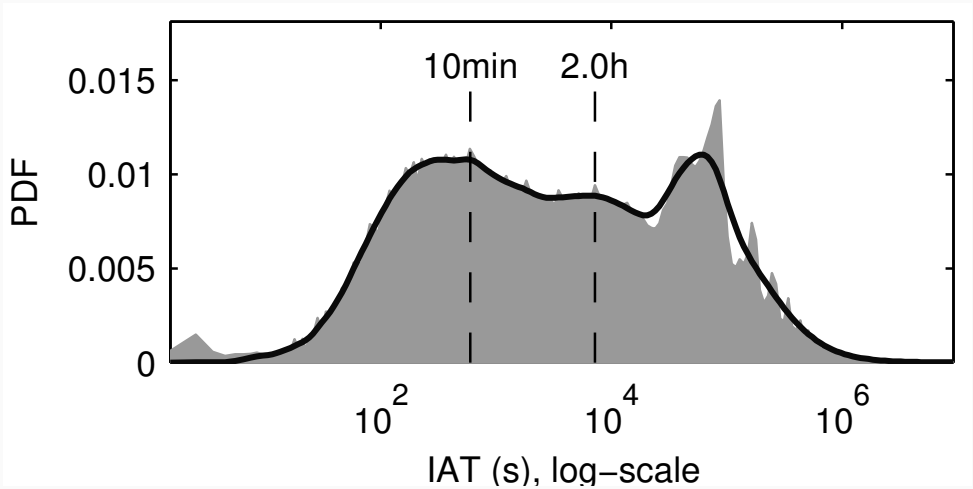


- Reddit Dataset
- Twitter Dataset
- Stack-Overflow Dataset
- Hacker-News Dataset

Pattern 2: Bimodal IAT distribution

Users have highly active sessions and resting periods

Log-binned histogram and KDE of postings' IAT

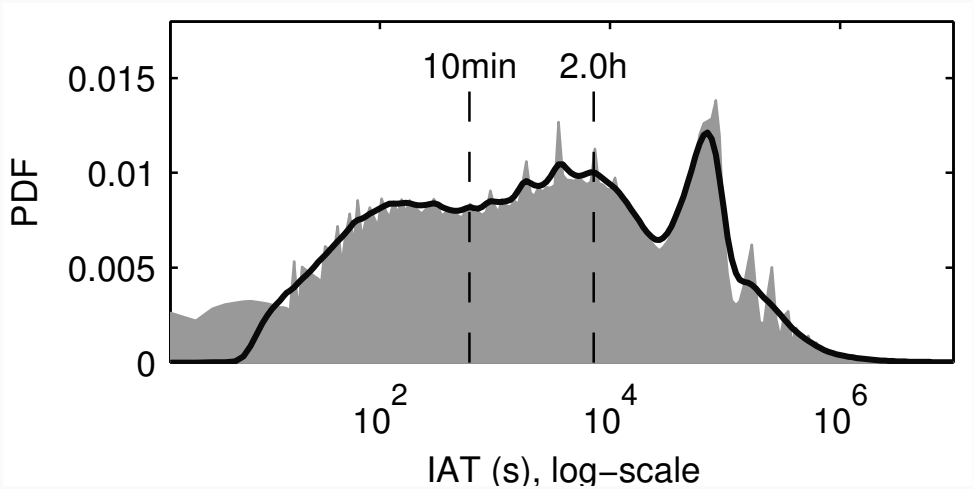


- Reddit

Pattern 2: Bimodal IAT distribution

Users have highly active sessions and resting periods

Log-binned histogram and KDE of postings' IAT

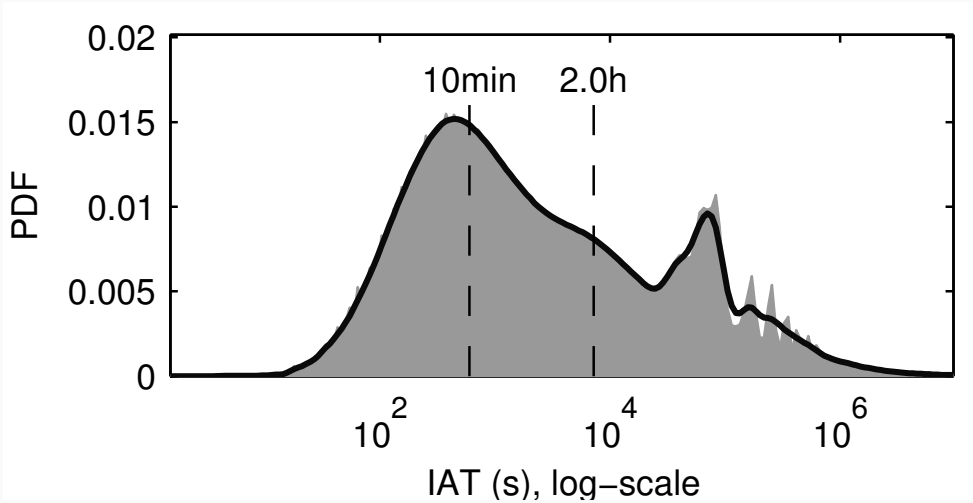


- Reddit
- Twitter

Pattern 2: Bimodal IAT distribution

Users have highly active sessions and resting periods

Log-binned histogram and KDE of postings' IAT

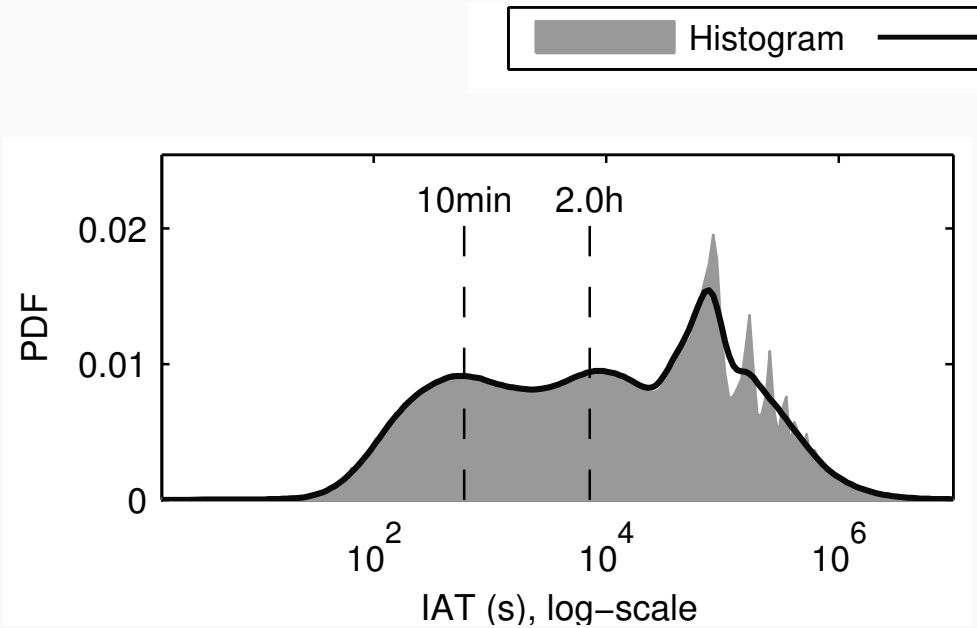


- Reddit
- Twitter
- Stack-Overflow

Pattern 2: Bimodal IAT distribution

Users have highly active sessions and resting periods

Log-binned histogram and KDE of postings' IAT



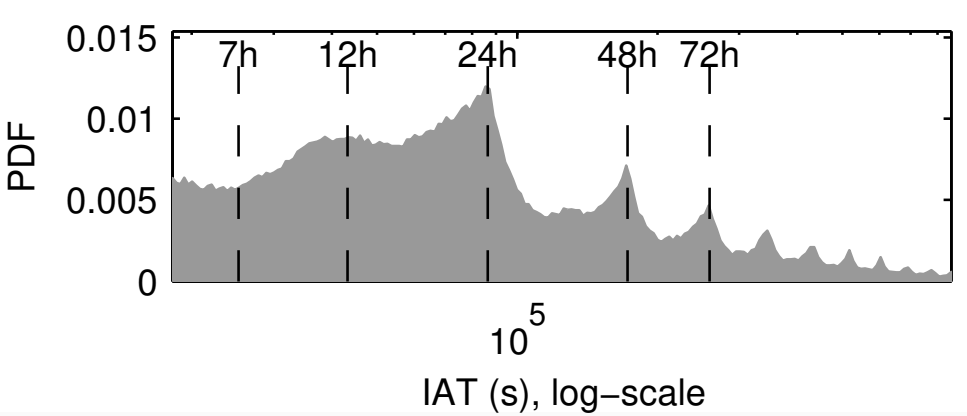
- Reddit
- Twitter
- Stack-Overflow
- Hacker-News

Act-M Model: Pattern Mining – Datasets

Pattern 3: Periodic spikes in the IAT distribution

Users' daily sleeping intervals create periodic peaks in the log-binned histogram of IAT

Log-binned histogram of postings' IAT (zoom)



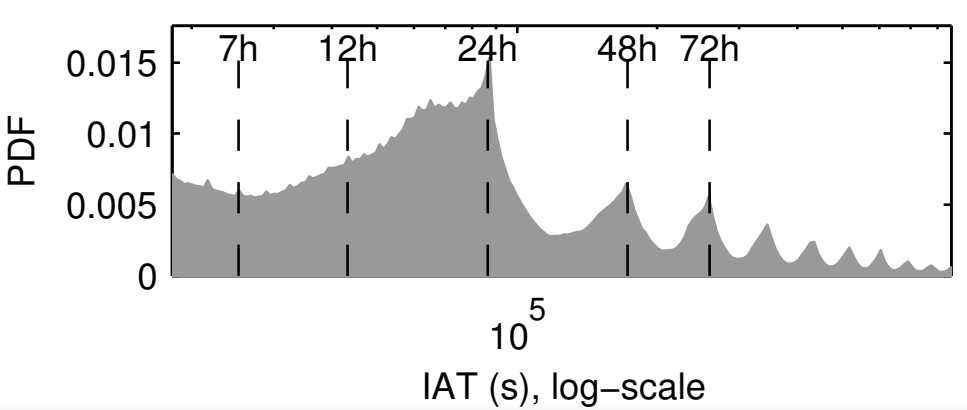
- Reddit

Act-M Model: Pattern Mining – Datasets

Pattern 3: Periodic spikes in the IAT distribution

Users' daily sleeping intervals create periodic peaks in the log-binned histogram of IAT

Log-binned histogram of postings' IAT (zoom)



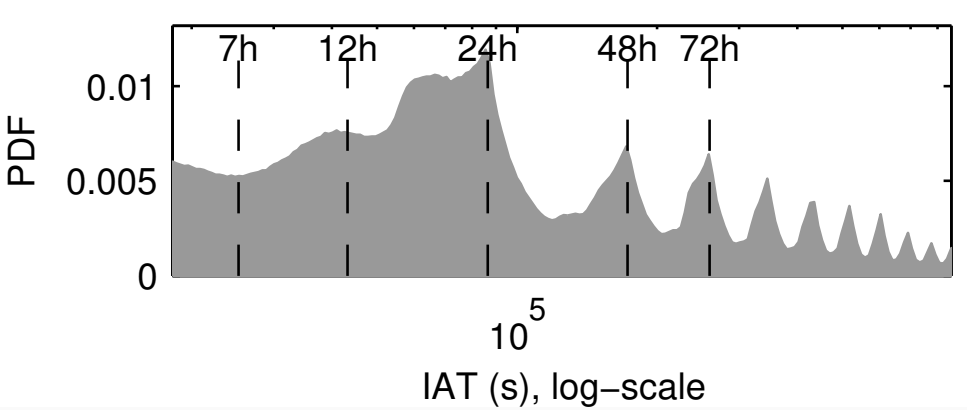
- Reddit
- Twitter

Act-M Model: Pattern Mining – Datasets

Pattern 3: Periodic spikes in the IAT distribution

Users' daily sleeping intervals create periodic peaks in the log-binned histogram of IAT

Log-binned histogram of postings' IAT (zoom)



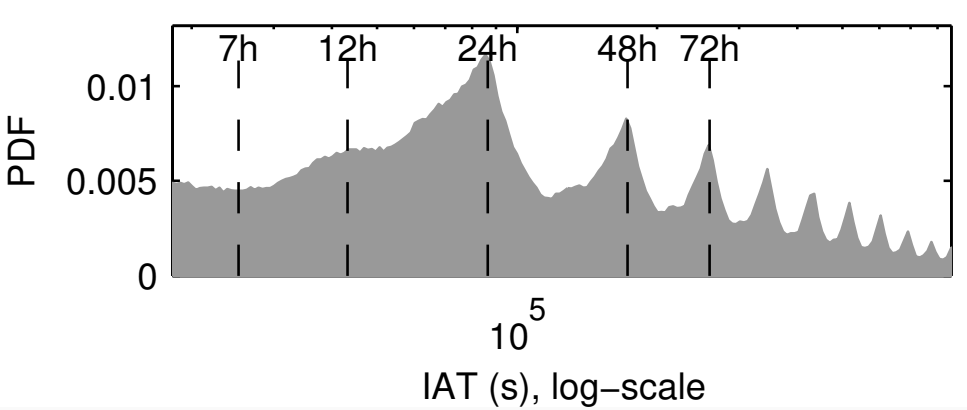
- Reddit
- Twitter
- Stack-Overflow

Act-M Model: Pattern Mining – Datasets

Pattern 3: Periodic spikes in the IAT distribution

Users' daily sleeping intervals create periodic peaks in the log-binned histogram of IAT

Log-binned histogram of postings' IAT (zoom)

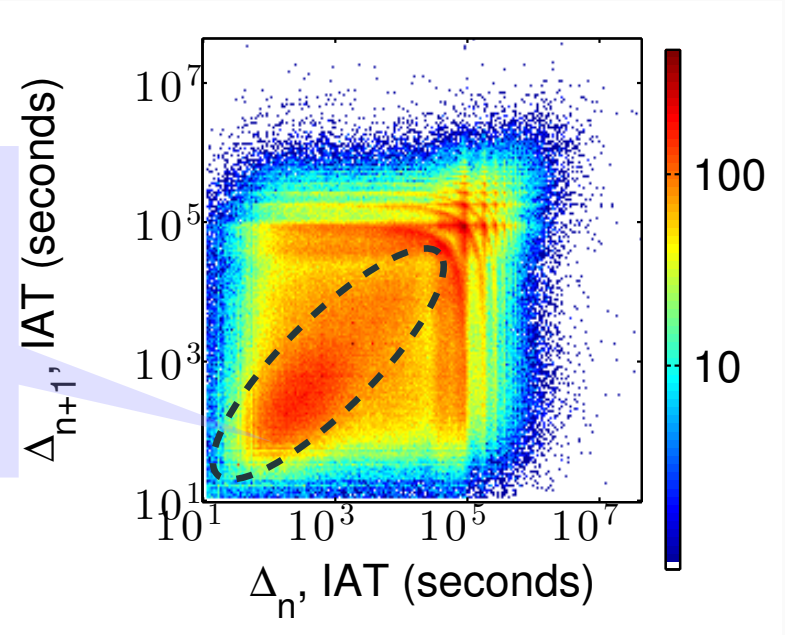


- Reddit
- Twitter
- Stack-Overflow
- Hacker-News

Pattern 4: Consecutive IAT are correlated

Long/short IAT are likely to be followed by long/short IAT

Joint distribution of consecutive IAT (heat-map)



Concentration of pairs in the diagonal: positive correlation

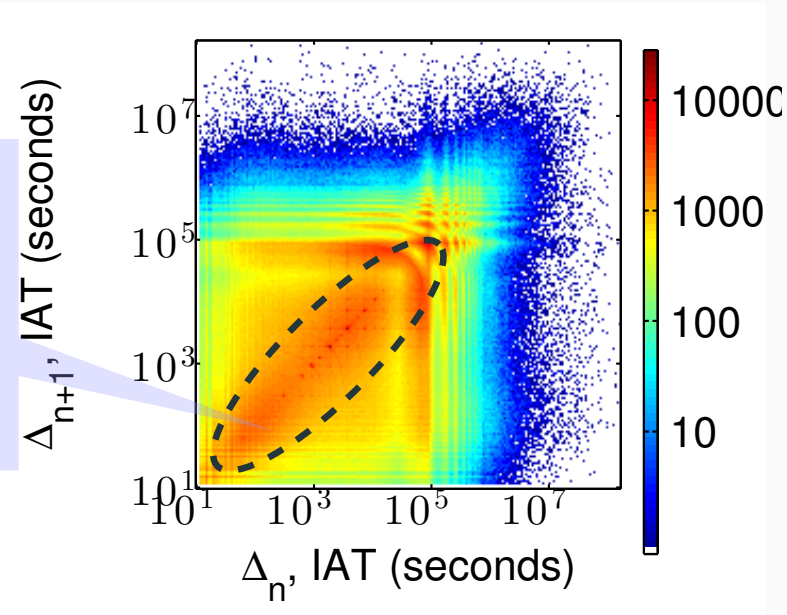
- Reddit

Act-M Model: Pattern Mining – Datasets

Pattern 4: Consecutive IAT are correlated

Long/short IAT are likely to be followed by long/short IAT

Joint distribution of consecutive IAT (heat-map)



Concentration of pairs in the diagonal: positive correlation

- Reddit
- Twitter

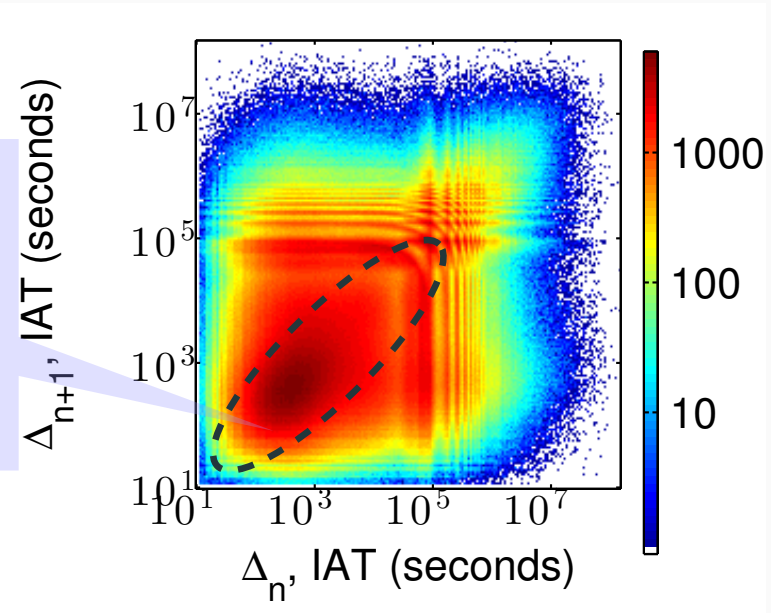
Act-M Model: Pattern Mining – Datasets

Pattern 4: Consecutive IAT are correlated

Long/short IAT are likely to be followed by long/short IAT

Joint distribution of consecutive IAT (heat-map)

Concentration of pairs in the diagonal: positive correlation



- Reddit
- Twitter
- Stack-Overflow

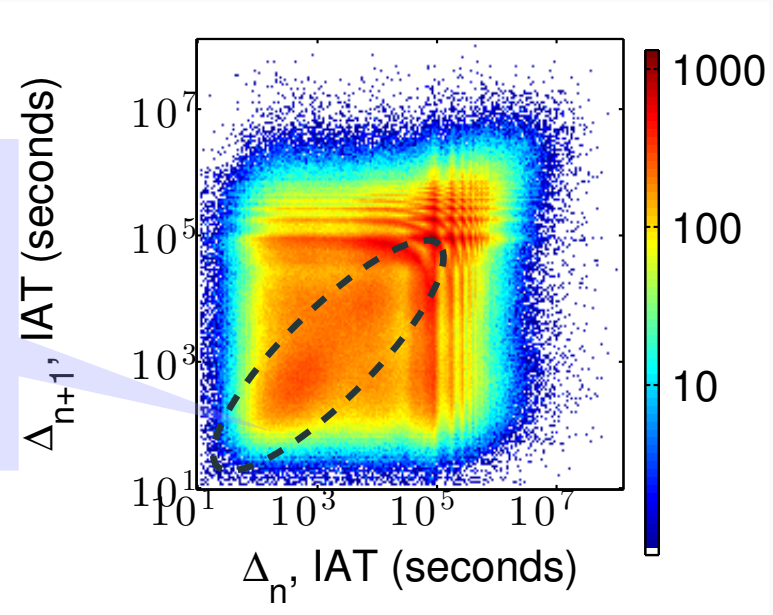
Act-M Model: Pattern Mining – Datasets

Pattern 4: Consecutive IAT are correlated

Long/short IAT are likely to be followed by long/short IAT

Joint distribution of consecutive IAT (heat-map)

Concentration of pairs in the diagonal: positive correlation



- Reddit
- Twitter
- Stack-Overflow
- Hacker-News

Act-M (Activity Model)

Can we generate synthetic time-stamps that match real data patterns?

Pattern	Poisson	Power-Law Barabási, 2005	CNPP Malmgren, 2009	SFP Vaz de Melo, 2013	**Act-M** Proposed
Heavy Tails		✓		✓	✓
Bimodality			✓		✓
Periodic Spikes					✓
IAT Correlation				✓	✓

Goal

Design a model that matches all the activity patterns.

Act-M Model: SCorr Process

Act-M Model: SCorr | Markov Chain | Clock

Base model: Self-Correlated Process (SCorr) process

- **Definition**: A stochastic process is a SCorr process with base rate λ_o and correlation ρ if:

$$\delta_1 \sim \text{Exp}\left(\frac{1}{\lambda_o}\right), \delta_i \sim \text{Exp}\left(\rho \cdot \delta_{i-1} + \frac{1}{\lambda_o}\right)$$

$X \sim \text{Exp}(1/\lambda)$: Exponential random variable with rate λ

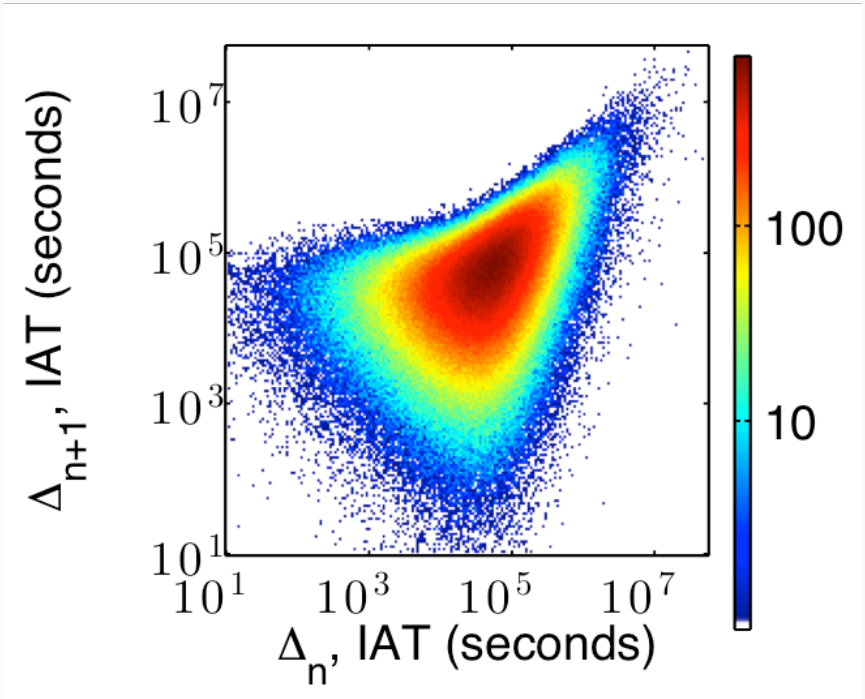
Properties:

- **Correlated IAT:** The i-th IAT δ_i depends on the previous IAT δ_{i-1}
- **Correlation Strength:** ρ controls correlation strength. If $\rho = 0$, SCorr reduces to an exponential distribution

Act-M Model: SCorr Process

Act-M Model: SCorr | Markov Chain | Clock

Consecutive IAT Distribution



$\lambda = 20\text{h}$, $\rho = 0.7$

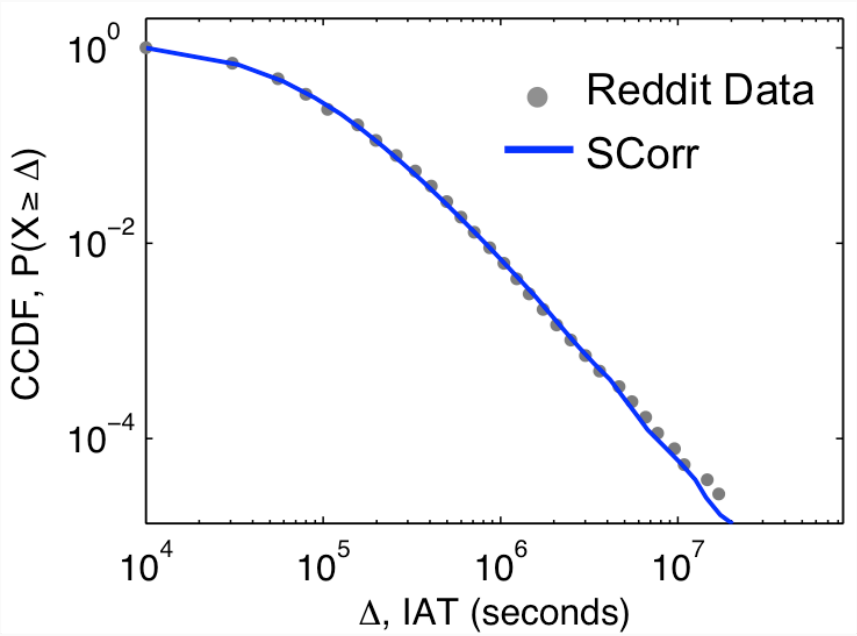
Patterns:

- Correlated IAT

Act-M Model: SCorr Process

Act-M Model: SCorr | Markov Chain | Clock

IAT CCDF



$\lambda = 20\text{h}$, $\rho = 0.7$

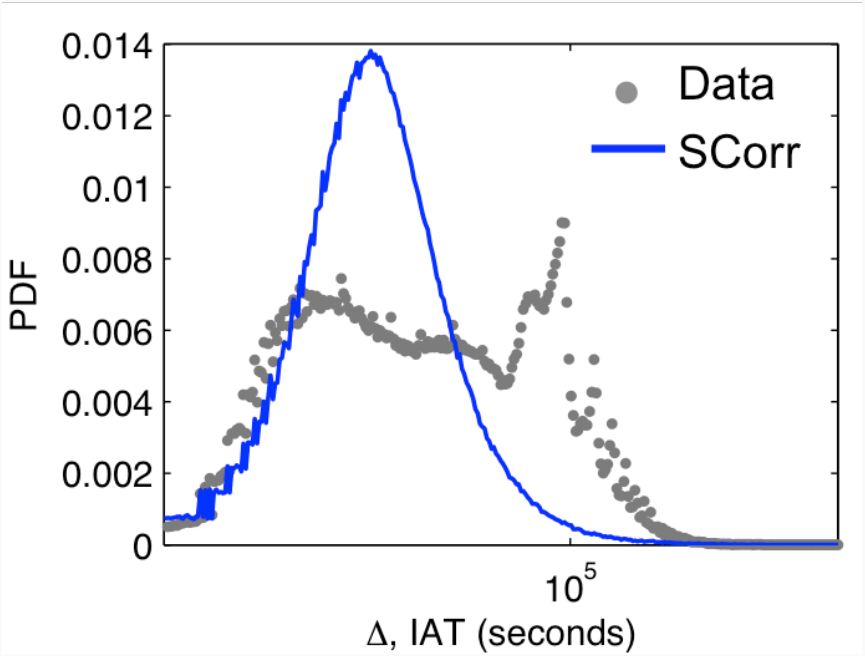
Patterns:

- Correlated IAT
- Heavy-tail

Act-M Model: SCorr Process

Act-M Model: SCorr | Markov Chain | Clock

IAT Log-binned Histogram



$\lambda = 20\text{min}$, $\rho = 1.0$

Patterns:

- Correlated IAT
- Heavy-tail
- ~~Bimodality~~
- ~~Periodic Spikes~~

Act-M Model: Markov Chain

Act-M Model: SCorr | Markov Chain | Clock

Active State:

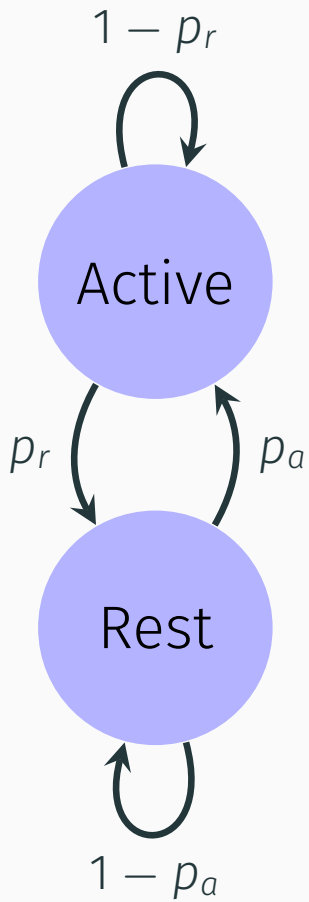
1. Wait $\delta_i \sim \text{SCorr}(\lambda_A, \rho_A)$
2. Make a post
3. Transition

Rest State:

1. Wait $\delta_i \sim \text{SCorr}(\lambda_R, \rho_R)$
2. Transition

Base Rates $\lambda_A > \lambda_R$

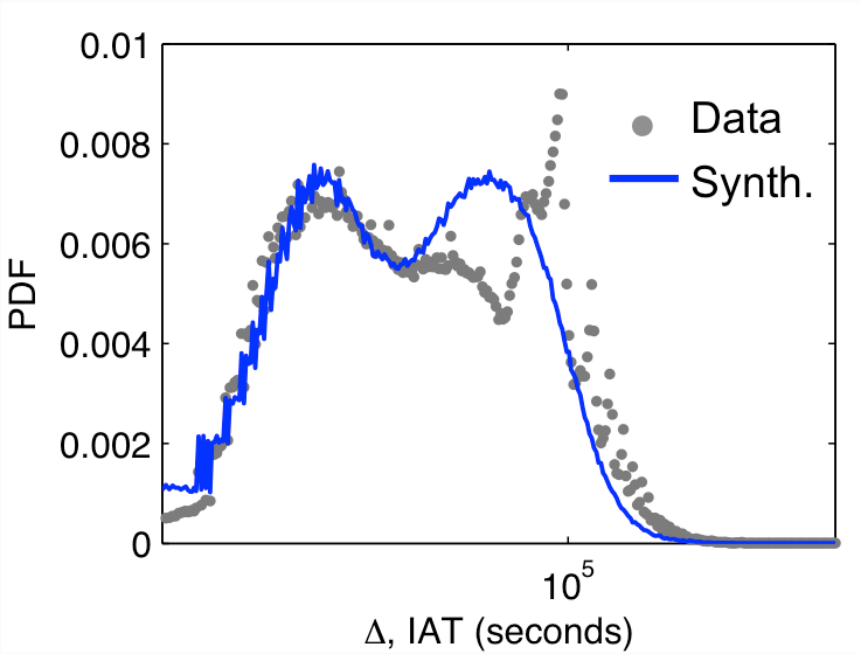
- Active state avg. wait time is smaller than rest state avg. wait time



Act-M Model: Markov Chain

Act-M Model: SCorr | Markov Chain | Clock

IAT Log-binned Histogram



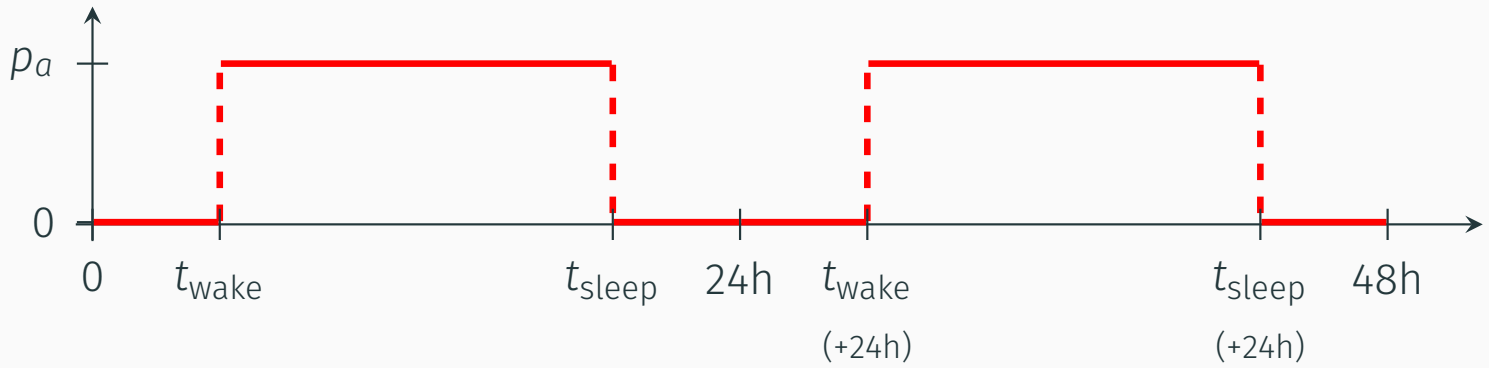
Patterns:

- Correlated IAT
- Heavy-tail
- Bimodality
- ~~Periodic Spikes~~

Act-M Model: Clock

Act-M Model: SCorr | Markov Chain | Clock

Probability p_a of becoming active changes over time:



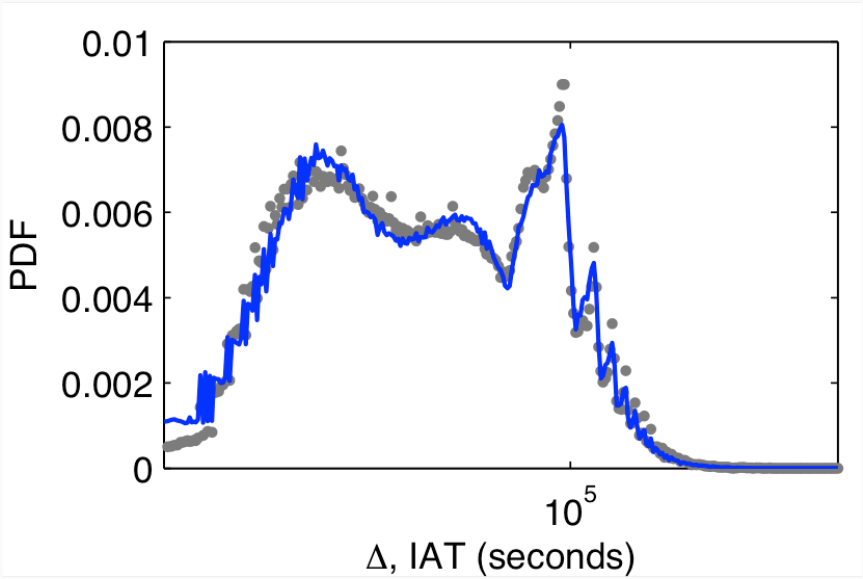
Clock variable t_{clock} keeps track of current time:

- 0:00:00h $< t_{clock} <$ 23:59:59h
- Update t_{clock} after each state transition

Act-M Model: Complete

Act-M Model: SCorr → Markov Chain → Clock

IAT Log-binned Histogram



Patterns:

- Correlated IAT
- Heavy-tail
- Bimodality
- Periodic Spikes

Parameter Estimation

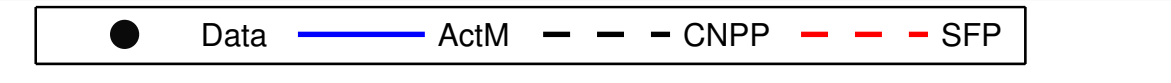
We use the Levenberg-Marquardt algorithm to minimize the difference between the Act-M and the data histograms.

Act-M Model: Experiments – Can It Match Real Data?

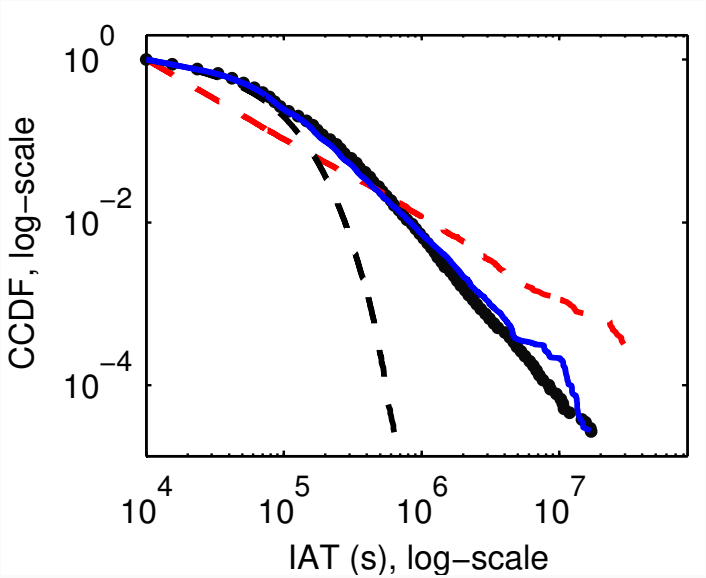
CNPP
Malmgren et. al

SFP
Vaz de Melo et. al

Act-M
Proposed Model



IAT CCDF



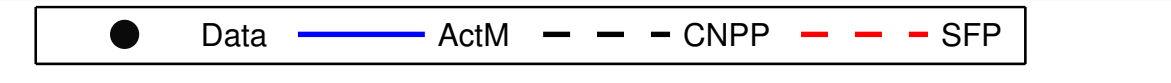
Pattern	CNPP	SFP	**Act-M**
Heavy-Tail	No	Yes	Yes
Bimodality			
Spikes			
Correlation			

Act-M Model: Experiments – Can It Match Real Data?

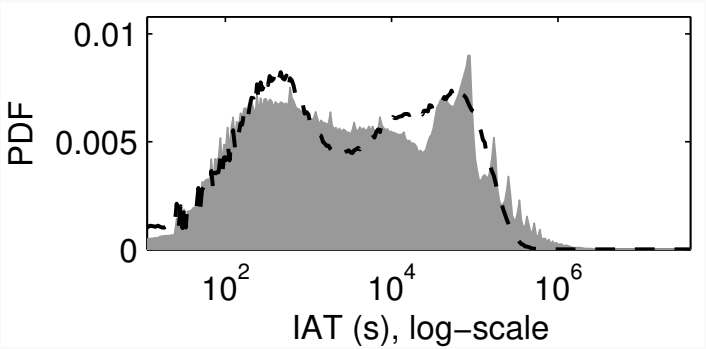
CNPP
Malmgren et. al

SFP
Vaz de Melo et. al

Act-M
Proposed Model



IAT Log-binned Histogram



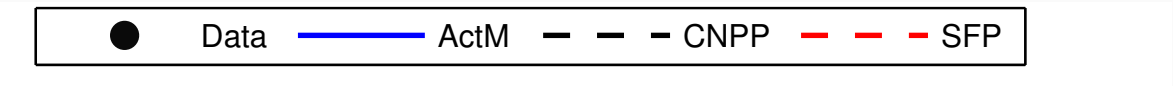
Pattern	CNPP	SFP	Act-M
Heavy-Tail	No	Yes	Yes
Bimodality	Yes		
Spikes	No		
Correlation			

Act-M Model: Experiments – Can It Match Real Data?

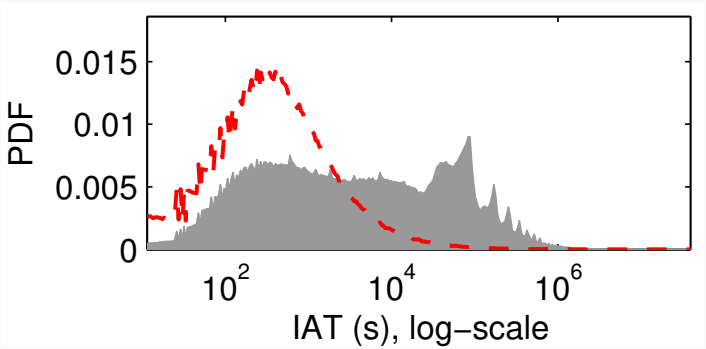
CNPP
Malmgren et. al

SFP
Vaz de Melo et. al

Act-M
Proposed Model



IAT Log-binned Histogram



Pattern	CNPP	SFP	**Act-M**
Heavy-Tail	No	Yes	Yes
Bimodality	Yes	No	
Spikes	No	No	
Correlation			

Act-M Model: Experiments – Can It Match Real Data?

CNPP

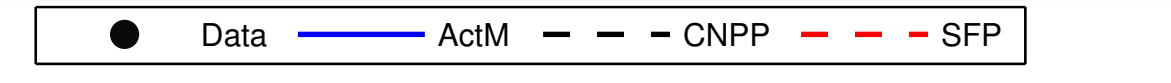
Malmgren et. al

SFP

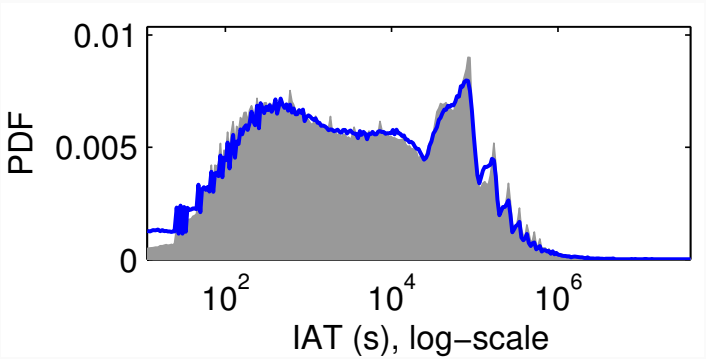
Vaz de Melo et. al

Act-M

Proposed Model



IAT Log-binned Histogram



Pattern	CNPP	SFP	**Act-M**
Heavy-Tail	No	Yes	Yes
Bimodality	Yes	No	Yes
Spikes	No	No	Yes
Correlation			

Act-M Model: Experiments – Can It Match Real Data?

CNPP

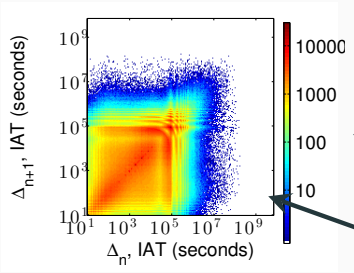
Malmgren et. al

SFP

Vaz de Melo et. al

Act-M

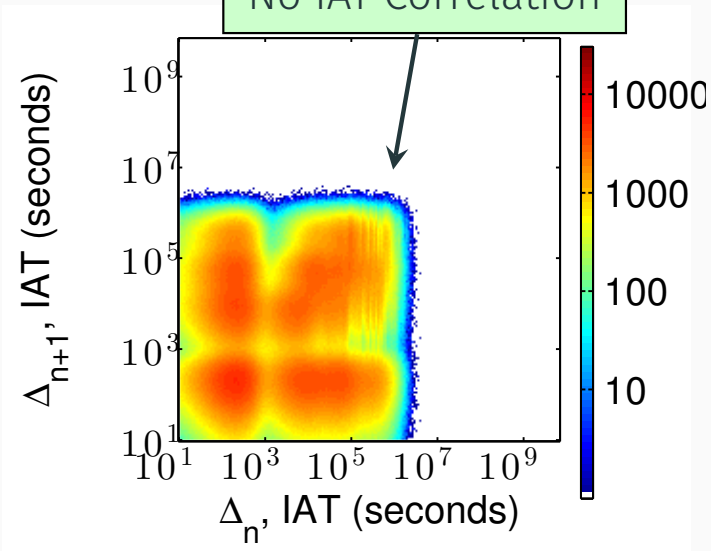
Proposed Model



Twitter Data

No IAT Correlation

Fit



Pattern	CNPP	SFP	**Act-M**
Heavy-Tail	No	Yes	Yes
Bimodality	Yes	No	Yes
Spikes	No	No	Yes
Correlation	No		

Act-M Model: Experiments – Can It Match Real Data?

CNPP

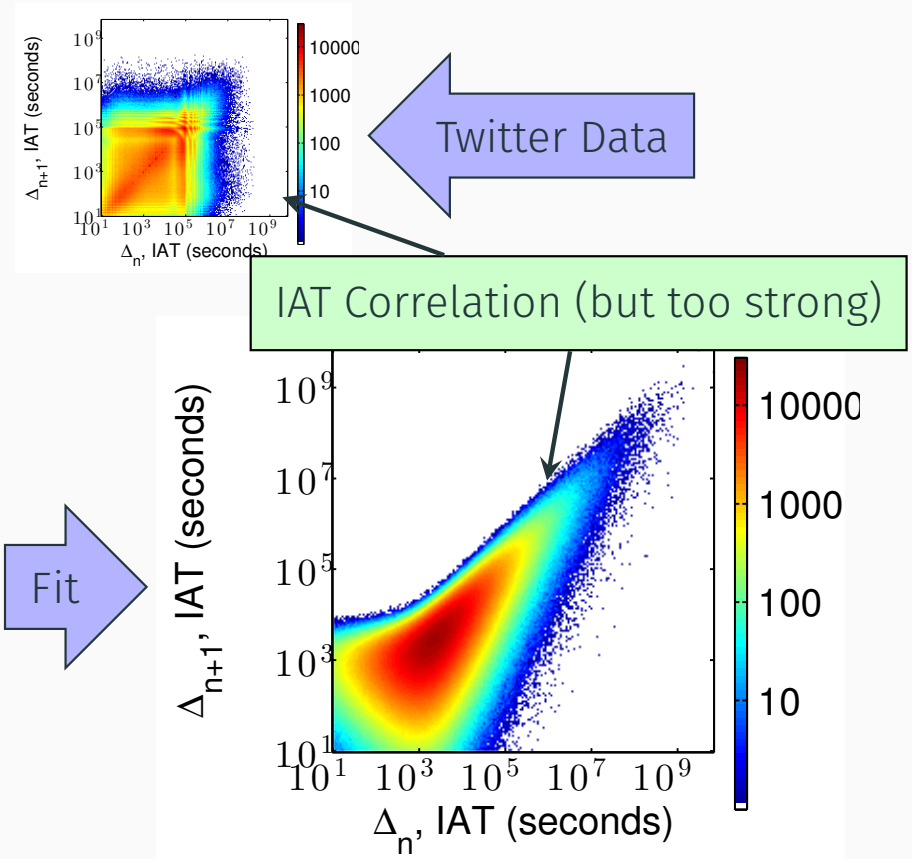
Malmgren et. al

SFP

Vaz de Melo et. al

Act-M

Proposed Model



Pattern	CNPP	SFP	Act-M
Heavy-Tail	No	Yes	Yes
Bimodality	Yes	No	Yes
Spikes	No	No	Yes
Correlation	No	Yes	

Act-M Model: Experiments – Can It Match Real Data?

CNPP

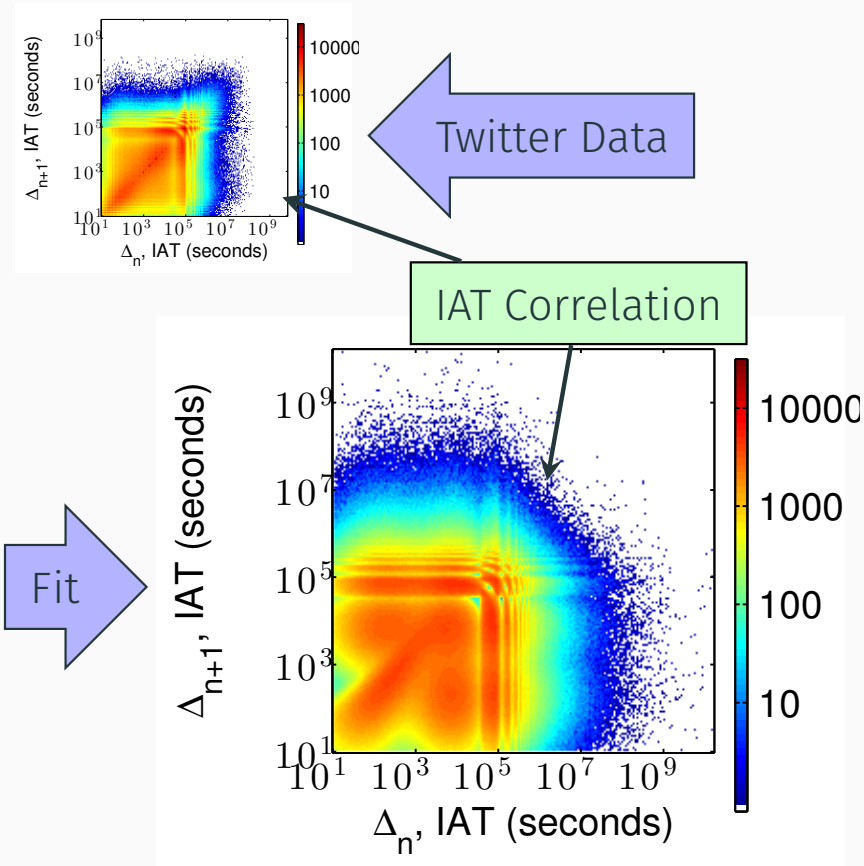
Malmgren et. al

SFP

Vaz de Melo et. al

Act-M

Proposed Model



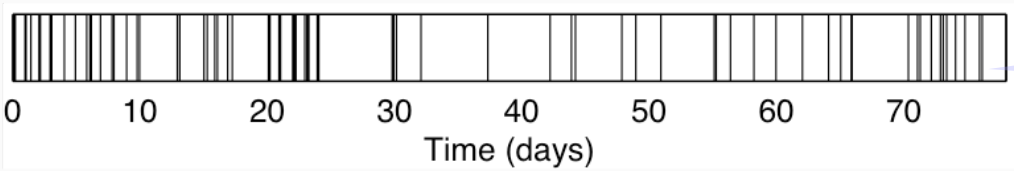
Pattern	CNPP	SFP	Act-M
Heavy-Tail	No	Yes	Yes
Bimodality	Yes	No	Yes
Spikes	No	No	Yes
Correlation	No	Yes	Yes

Act-M Model: Bot Detection

Problem: Bot Detection

Given labeled time-stamp data from a set of users $\{\mathcal{U}_1, \mathcal{U}_2, \cdots\}$ decide if an unknown user $\mathcal{U}_i$ is a human or a bot.

Time-stamps from a single user



The user that produced the time-stamps is a human or a bot?

Solution: ActM-Spotter

- Compare users' IAT to synthetic IAT generated by the Act-M
- If not similar to Act-M, then the user is likely to be a bot

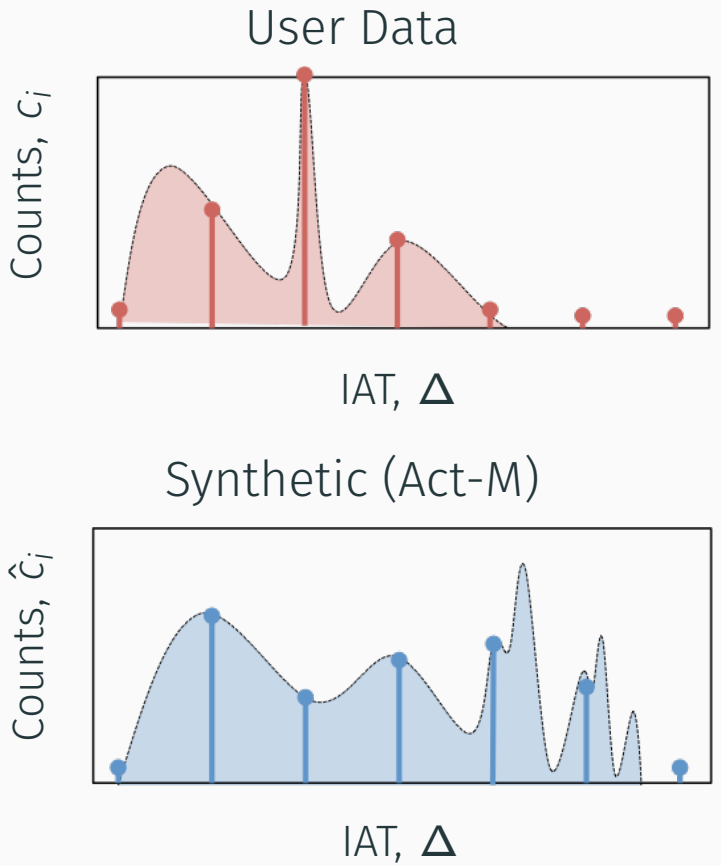
Act-M Model: Bot Detection

Training: estimate Act-M parameters

- Using time-stamps from all users from a dataset

For each user:

1. Compute the IAT histogram
2. Generate synthetic time-stamps using Act-M
   - Same number of time-stamps as the user
3. Compare user and synthetic IAT histogram
   - Cost-sensitive classification using the dissimilarity



$$\text{Dissimilarity} = \sum_{j=1}^{K} |c_j - \hat{c}_j|$$

Act-M Model: Bot Detection – Experiments

Datasets: users were manually labeled as bots or humans

1. **Reddit:** 1,963 humans / 37 bots
2. **Twitter:** 1,353 humans / 64 bots

Training:

- Same size for train and test subsets (preserved class distribution)

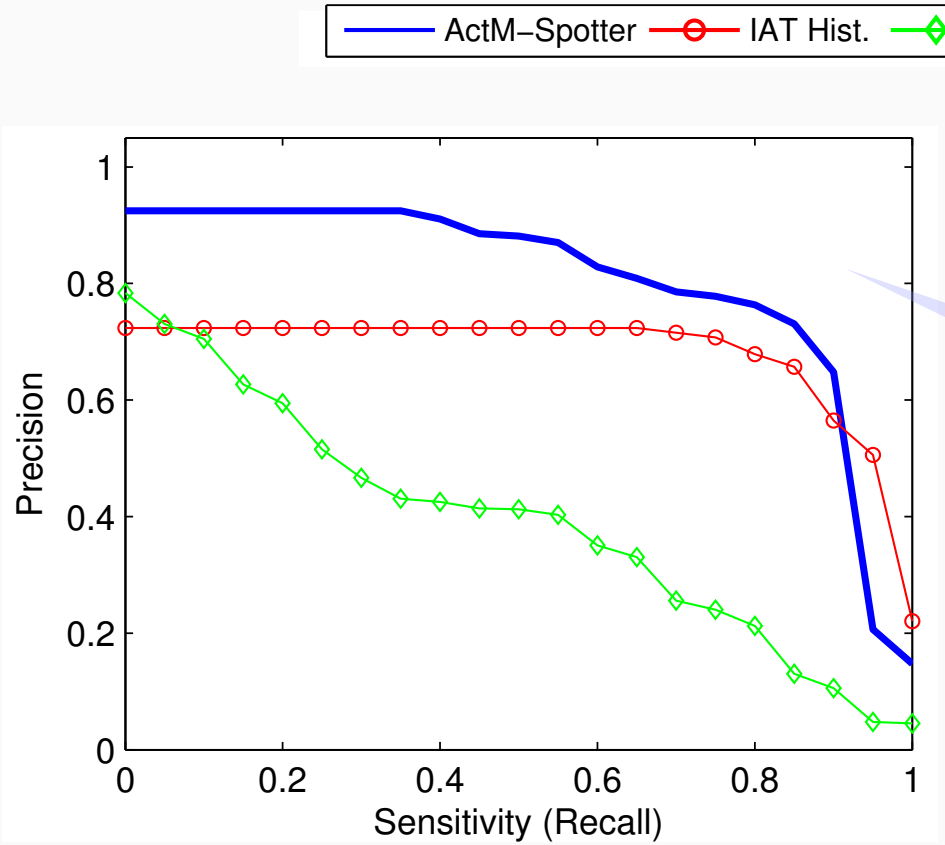
Baseline methods:

1. **IAT Histogram:** log-binned IAT histogram
2. **Entropy (Chu et al.):** entropy of the IAT distribution

Act-M Model: Bot Detection – Experiments

Precision vs. Sensitivity Curves

- Good performance: curve close to the top of the plot



Precision $\approx$ 80%
Sensitivity $\approx$ 80%
Imbalanced :

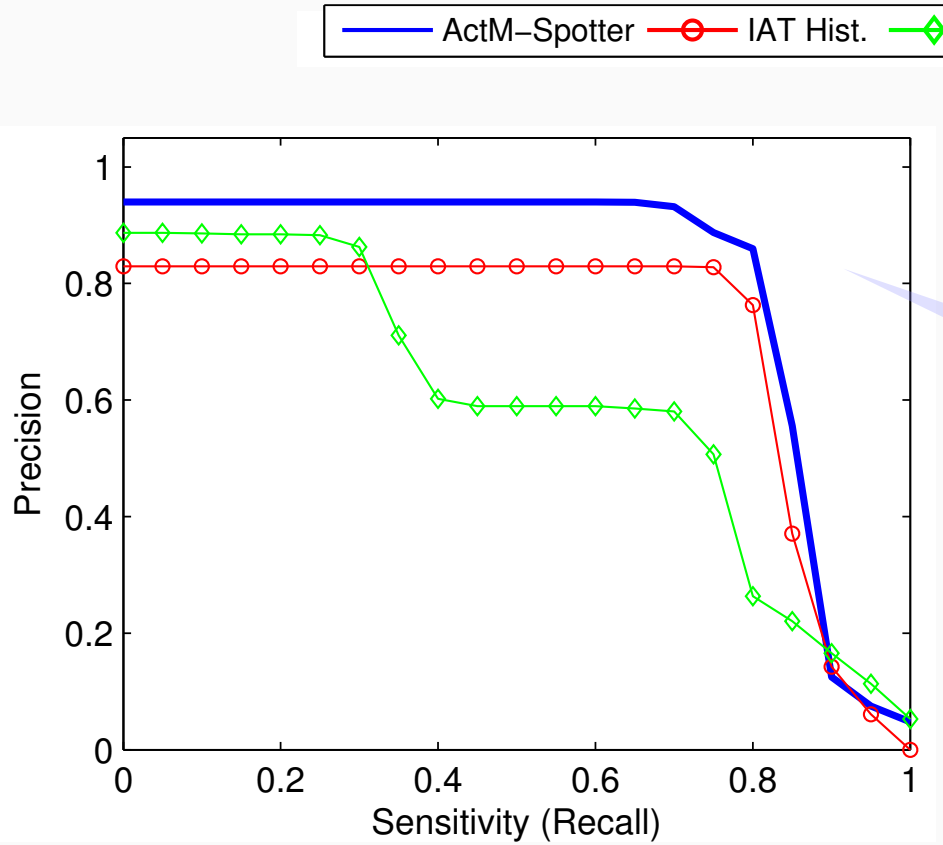
(# humans $\gg$ # bots)

Reddit Dataset

Act-M Model: Bot Detection – Experiments

Precision vs. Sensitivity Curves

- Good performance: curve close to the top of the plot



Precision ≈ 90%
Sensitivity ≈ 80%
Imbalanced datasets

(# humans ≫ # bots)

Twitter Dataset

Act-M Model: Summary

Pattern Mining:

- We showed that the IAT distribution of users' postings in social media is characterized by four patterns: heavy-tails, bimodality, periodic spikes and positive correlation

The Act-M Model:

- Mathematical model that matches the IAT distribution of social media users' postings

Bot Detection:

- We can use the Act-M to tell if a user is a bot based only on time-stamp data

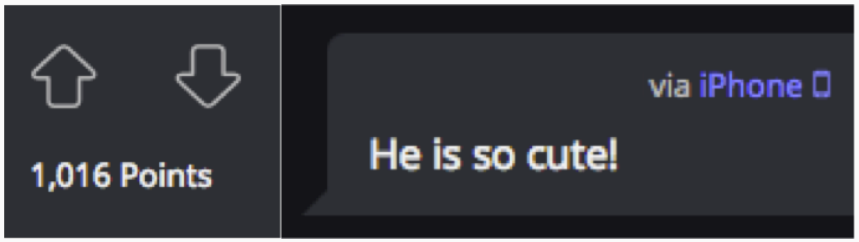
Outline

VnC Model: Introduction

In social voting services,
users can submit content
(e.g. pictures, news articles)

And other users can:

- Up-vote (like)
- Down-vote (dislike)
- Post comments



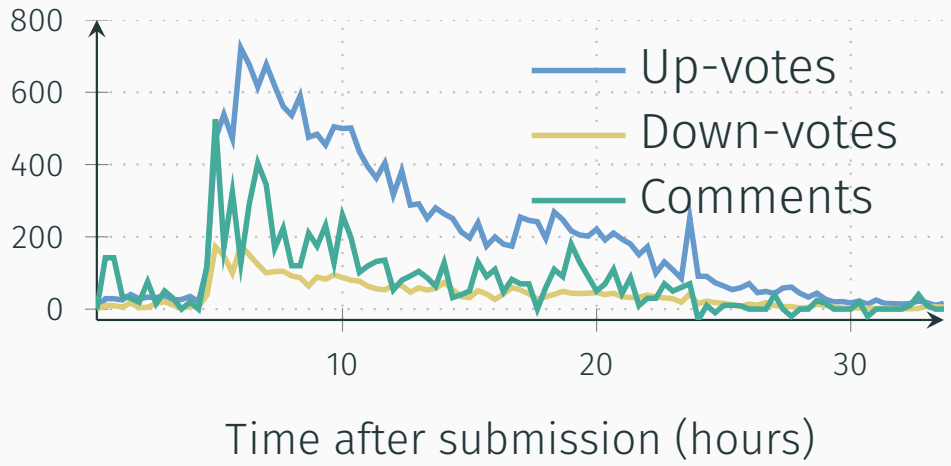
Examples of social voting services: Reddit, Imgur, Hacker-News

VnC Model: Problem Statement

For a submission, we have 3 time-series: up-votes $v_+(t)$, down-votes $v_-(t)$ and comments $c(t)$:

Submission

Time-Series



Problem

Can we explain how $v_+(t)$, $v_-(t)$ and $c(t)$ evolve over time?

Vote-and-Comment (VnC) Model

VnC: mathematical model that describes how the volume of up-votes, down-votes and comments changes over time

VnC is composed of 3 submodels that describe the following relationships:

1. Up-votes over time
2. Up-votes vs. down-votes
3. Comments vs. votes

VnC Model: Up-votes Over Time

The number $v_+(t)$ of up-votes received by a submission at time t is a function of:

1. $P(t)$: probability of a user up-voting at time t
2. N_+: population of potential voters
3. $V_+(t)$: number of votes *accumulated* at time t

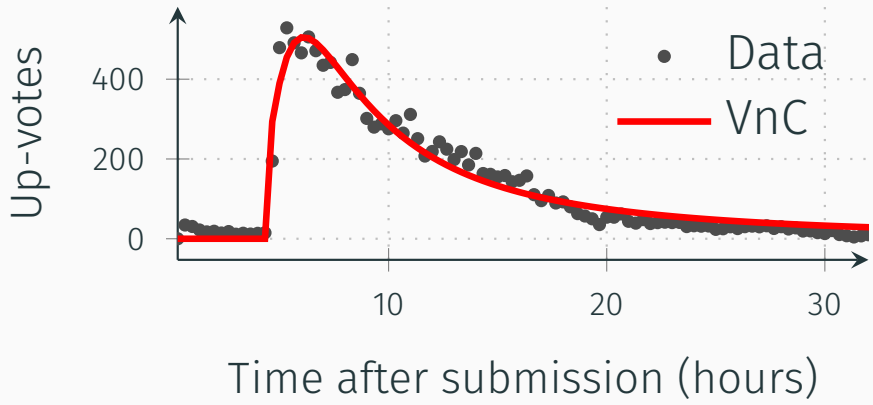
VnC Model: Up-votes Over Time

The number $v_+(t)$ of up-votes received by a submission at time t is a function of:

1. $P(t)$: probability of a user up-voting at time t
2. N_+: population of potential voters
3. $V_+(t)$: number of votes *accumulated* at time t

Up-votes over Time

$$v_+(t+1) = \underbrace{[N_+ - V_+(t)]}_{\text{users that can vote}} \cdot P(t)$$

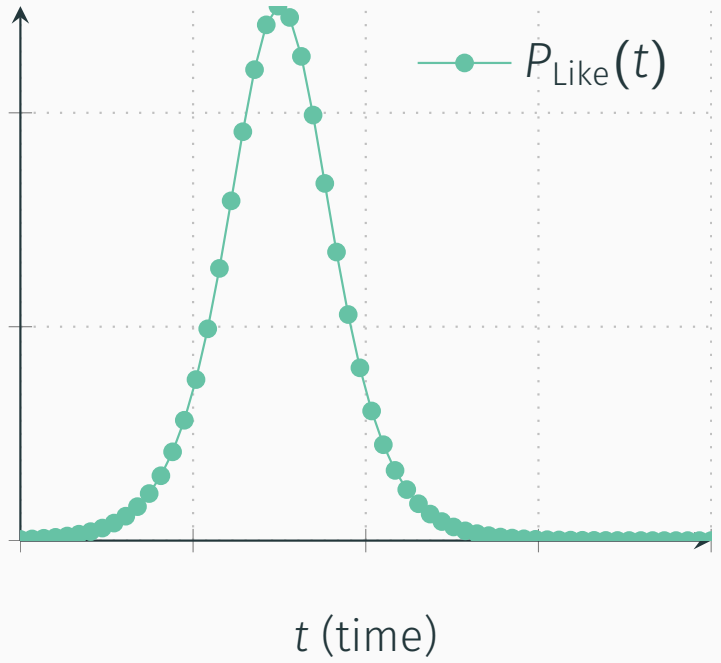


Next Step: How can we model the probability $P(t)$?

VnC Model: Up-voting Probability

$P_{\text{Like}}(t; \beta_+, \xi_+)$: Probability of liking a submission

- **Cascading Mechanism:** popularity affects probability
- $P_{\text{Like}}(t) = \xi_+ + \beta_+ \cdot V_+(t)/N_+$



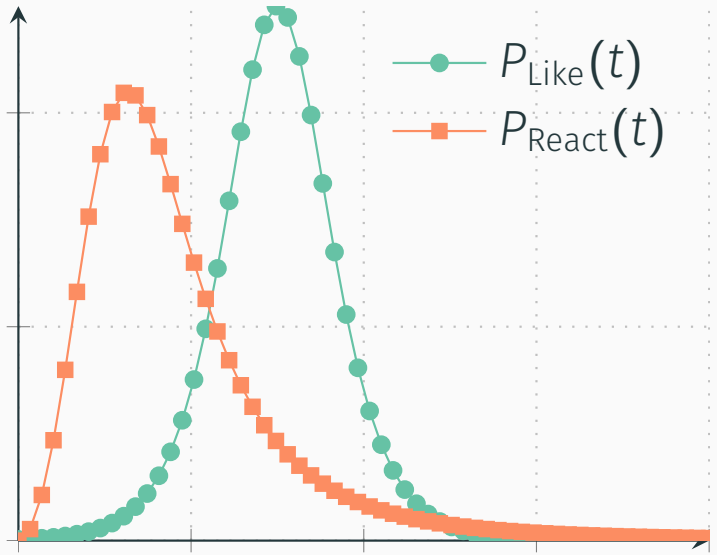
VnC Model: Up-voting Probability

$P_{\text{Like}}(t; \beta_+, \xi_+)$: Probability of liking a submission

- **Cascading Mechanism:** popularity affects probability
- $P_{\text{Like}}(t) = \xi_+ + \beta_+ \cdot V_+(t)/N_+$

$P_{\text{React}}(t; \mu, s)$: Probability that a user reacts at time t

- Log-logistic with parameters μ and s Details



VnC Model: Up-voting Probability

$P_{\text{Like}}(t; \beta_+, \xi_+)$: Probability of liking a submission

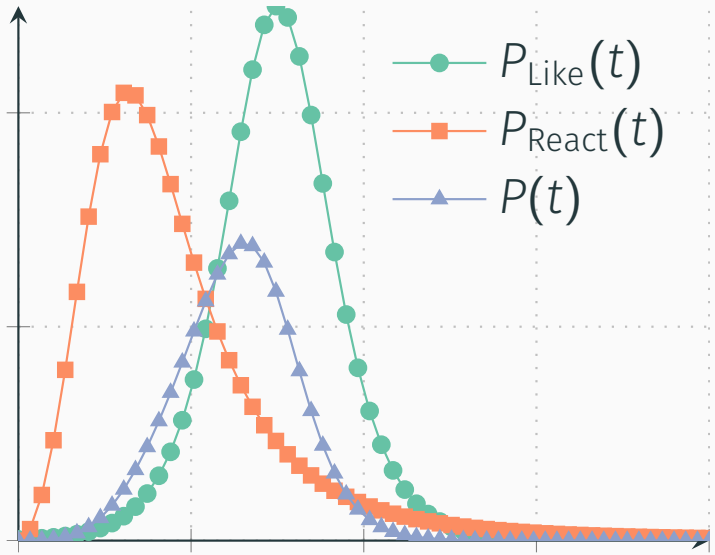
- **Cascading Mechanism:** popularity affects probability
- $P_{\text{Like}}(t) = \xi_+ + \beta_+ \cdot V_+(t)/N_+$

$P_{\text{React}}(t; \mu, s)$: Probability that a user reacts at time t

- Log-logistic with parameters μ and s Details

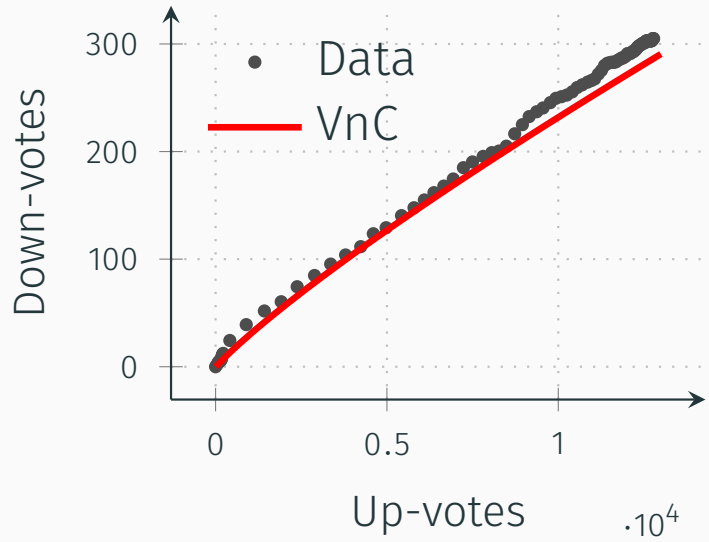
$P(t)$: Probability of a user up-voting at time t

- $P(t) = P_{\text{Like}}(t) \cdot P_{\text{React}}(t)$



VnC Model: Up-votes vs. Down-votes

$$v_-(t+1) = [N_- - V_-(t)] \cdot P_{\text{Like}}(t; \beta_-, \xi_-) \cdot P_{\text{React}}(t; \mu, s)$$

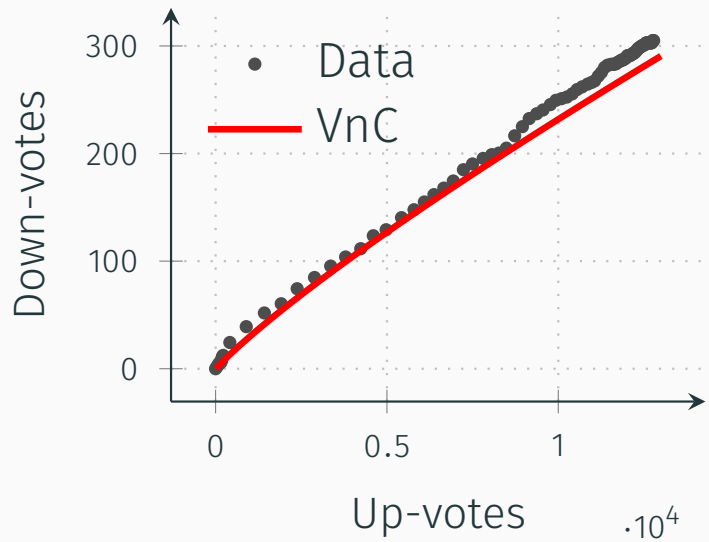


VnC Model: Up-votes vs. Down-votes

$$v_-(t+1) = [N_- - V_-(t)] \cdot \overbrace{P_{\text{Like}}(t;\beta_-,\xi_-)}^{\text{Cascading}} \cdot \overbrace{P_{\text{React}}(t;\mu,s)}^{\text{Reaction Times}}$$

The down-vote time-series $v_-(t)$ also follows:

1. A cascading mechanism
2. Log-logistic reaction times

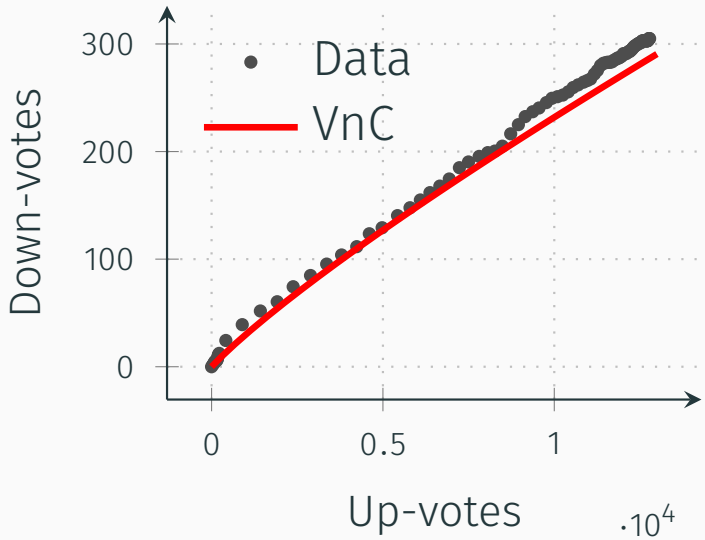


VnC Model: Up-votes vs. Down-votes

$$v_-(t+1) = \underbrace{[N_- - V_-(t)] \cdot \overbrace{P_{\text{Like}}(t;\beta_-,\xi_-)}^{\text{Cascading}}}_{\text{Not-shared Parameters}} \cdot \underbrace{\overbrace{P_{\text{React}}(t;\mu,s)}^{\text{Reaction Times}}}_{\text{Shared}}$$

The down-vote time-series $v_-(t)$ also follows:

1. A cascading mechanism
2. Log-logistic reaction times



Sharing parameters with the up-vote time-series:

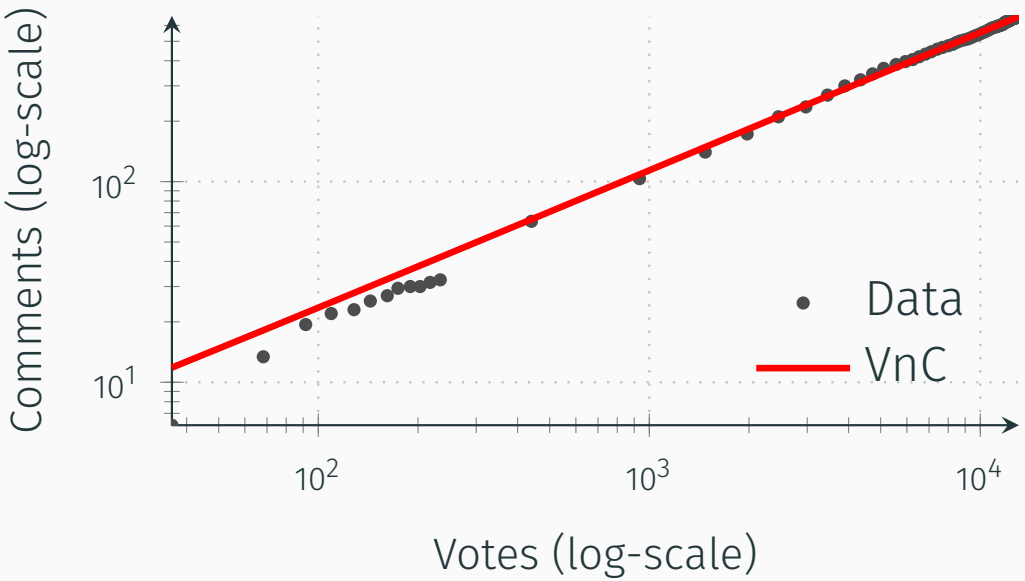
1. Shared: μ and s
2. Not shared: N_-, β_- and ξ_-

VnC Model: Comments vs. Votes

VnC models the number of comments $C(t)$ as a power law on the number of votes:

$$C(t) = k \cdot [V_+(t) + V_-(t)]^{\alpha}$$

The power-law — matches the data



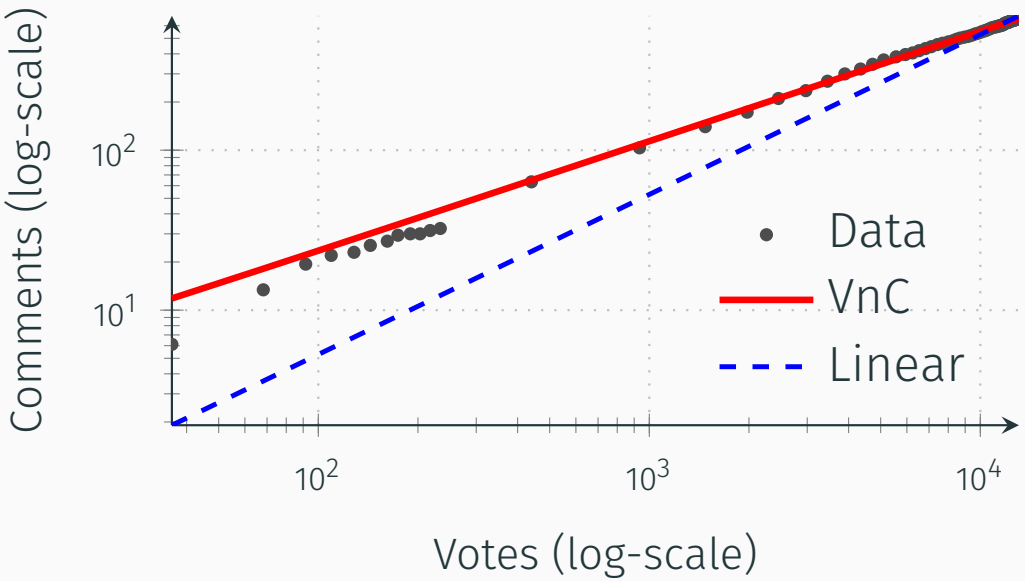
VnC Model: Comments vs. Votes

VnC models the number of comments $C(t)$ as a power law on the number of votes:

$$C(t) = k \cdot [V_+(t) + V_-(t)]^{\alpha}$$

The power-law —— matches the data

- The linear relationship fails to match the data



VnC Model: Experiments – Questions

Our goal is to answer the following questions:

- **Q1 – Fit Accuracy:** Is VnC more accurate than existing models when fitting social voting data?
- **Q2 – Popularity Decay:** Can VnC model the popularity decay of submissions?
- **Q3 – Coevolution:** Can VnC model the coevolution of up-votes, down-votes and comments time-series?

VnC Model: Experiments – Datasets

Our crawler tracked Reddit and Imgur submissions:

- Collected the number of votes and comments every 20 minutes
- Submissions were tracked for 33 hours after their creation
- Submissions with less than 100 up-votes were discarded

Digg dataset publicly avaiable (Lerman and Ghosh, 2010):

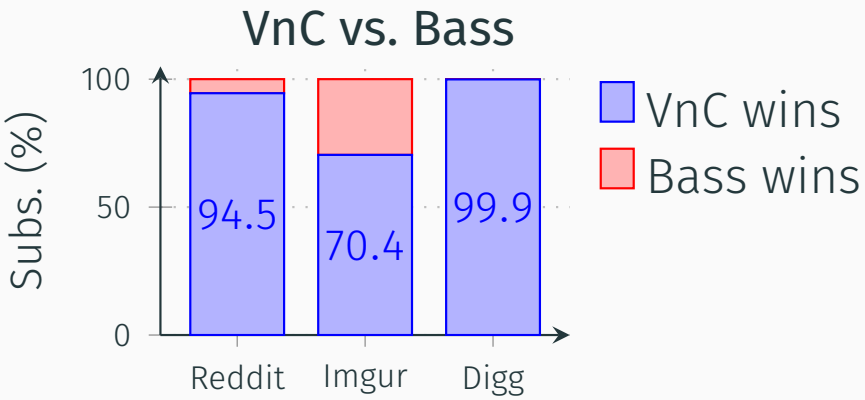
- Only up-votes (no down-votes and comments data)

Dataset	# Submissions	# User Interactions
Reddit	17,205	113,331,266
Imgur	724	2,107,576
Digg	3,553	5,149,170

VnC Model: Experiments – Q1. Fit Accuracy

Percentage of up-vote time-series that were best fit by each model

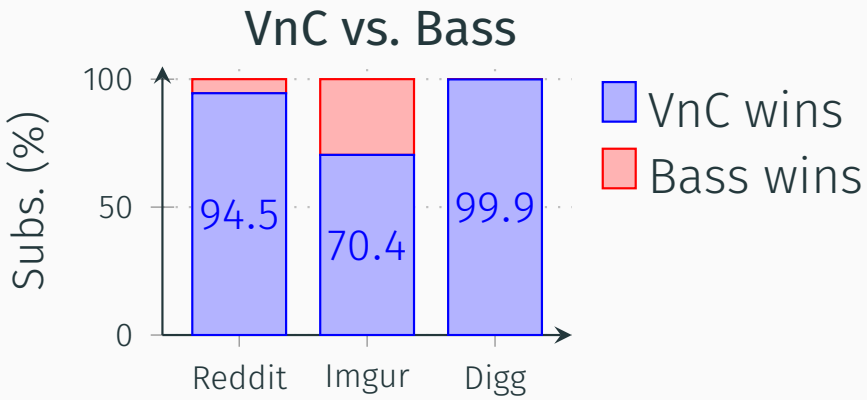
- Best fit determined by smaller root-mean-square error Details

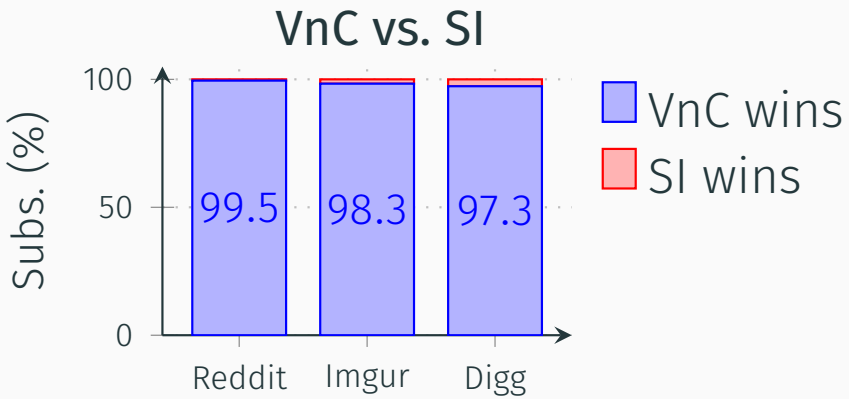


VnC Model: Experiments – Q1. Fit Accuracy

Percentage of up-vote time-series that were best fit by each model

- Best fit determined by smaller root-mean-square error Details

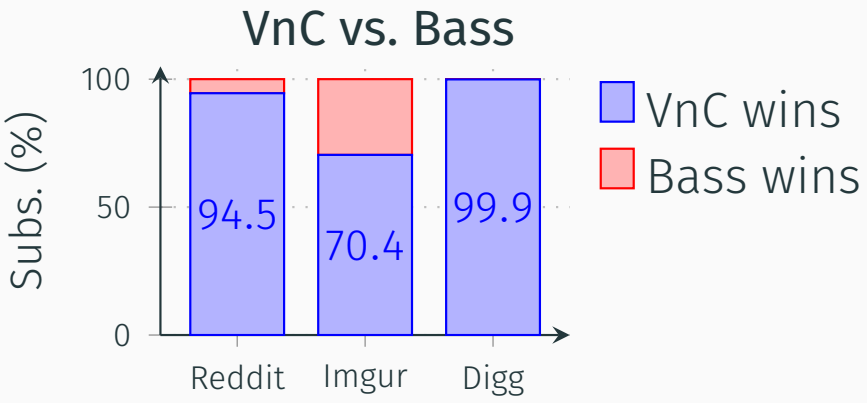


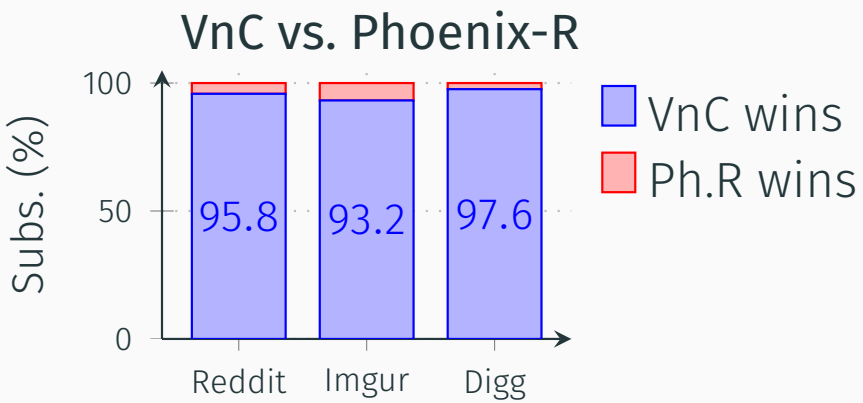


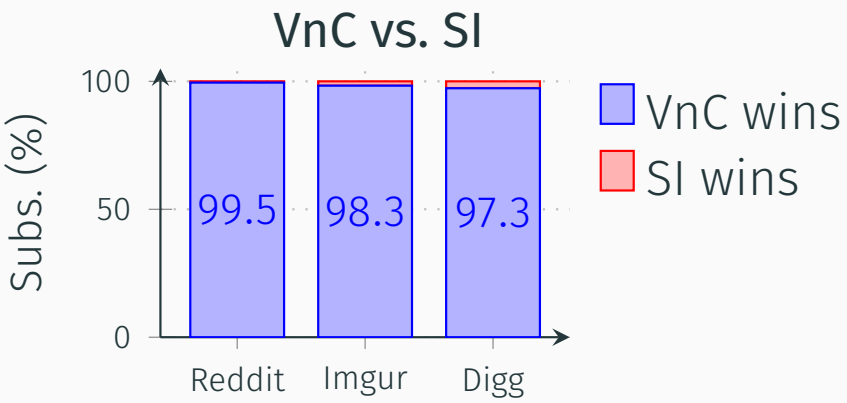
VnC Model: Experiments – Q1. Fit Accuracy

Percentage of up-vote time-series that were best fit by each model

- Best fit determined by smaller root-mean-square error Details



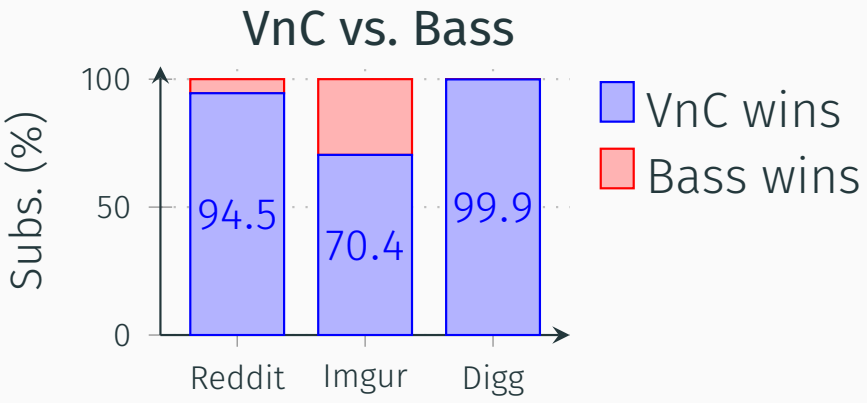


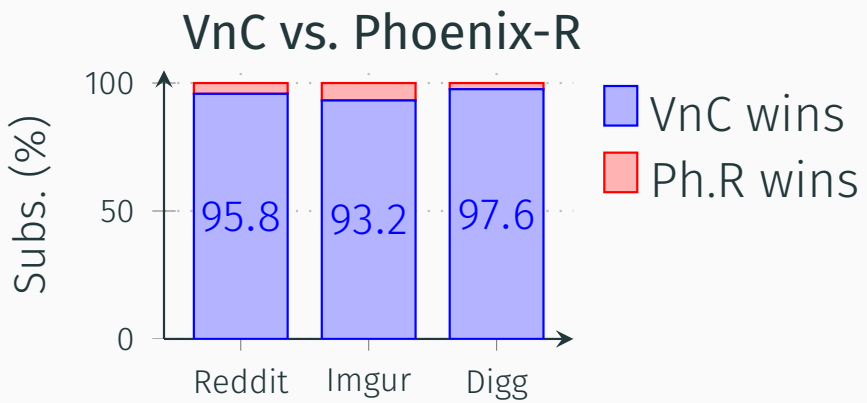


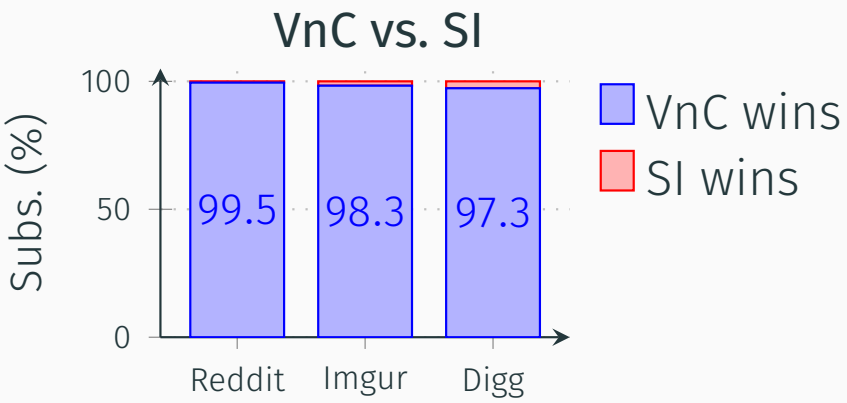
VnC Model: Experiments – Q1. Fit Accuracy

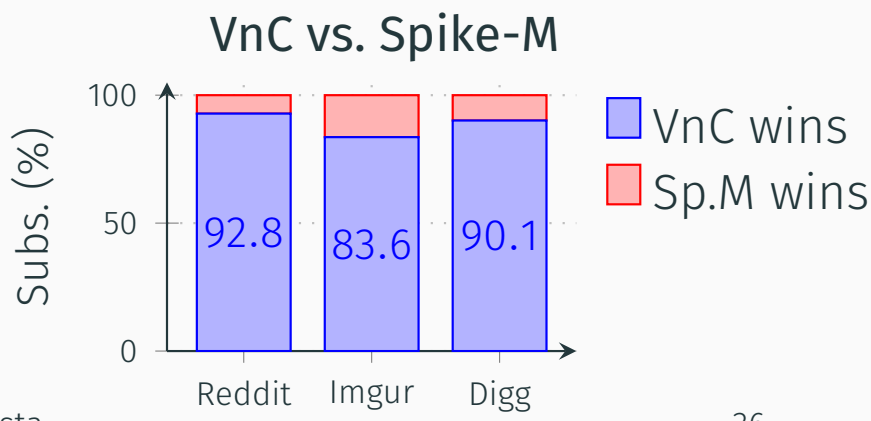
Percentage of up-vote time-series that were best fit by each model

- Best fit determined by smaller root-mean-square error Details



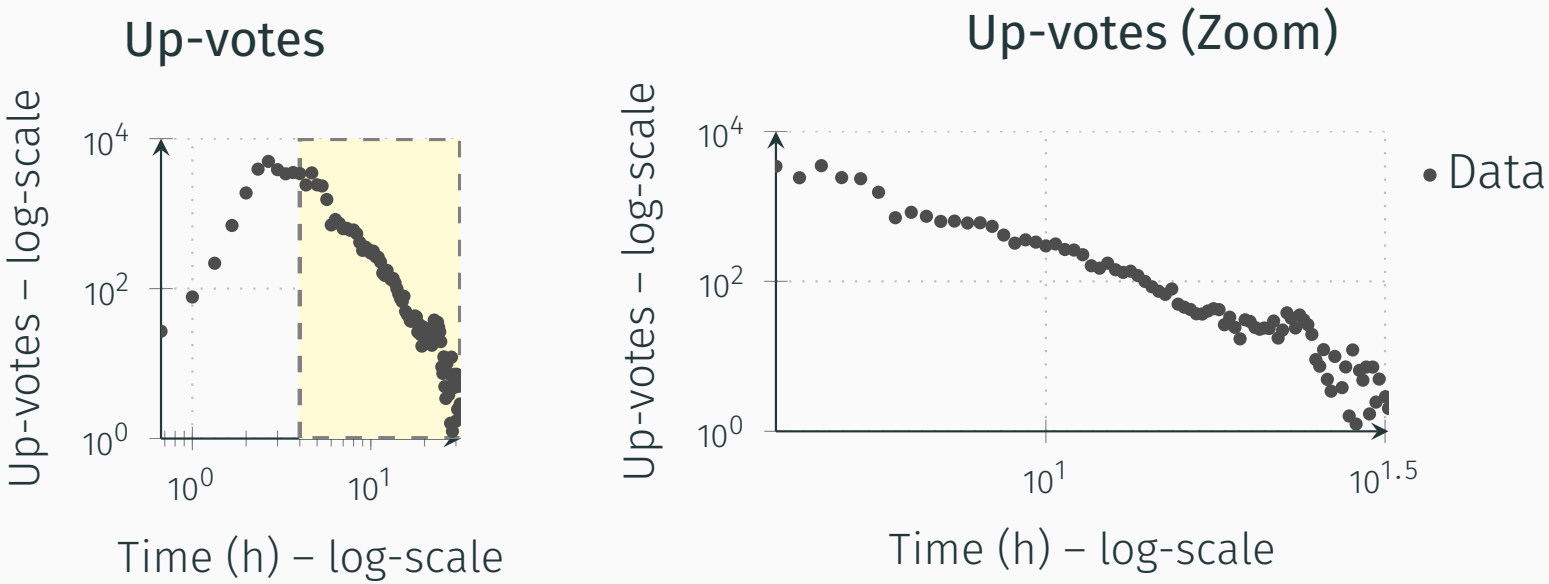






VnC Model: Experiments – Q2. Popularity Decay

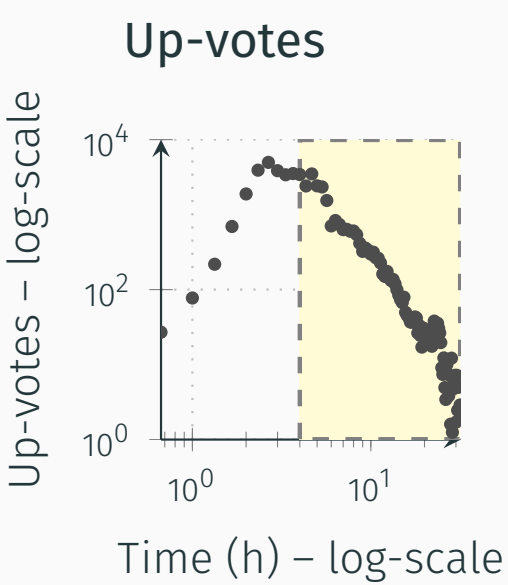
Up-vote time-series have a heavy-tail decay

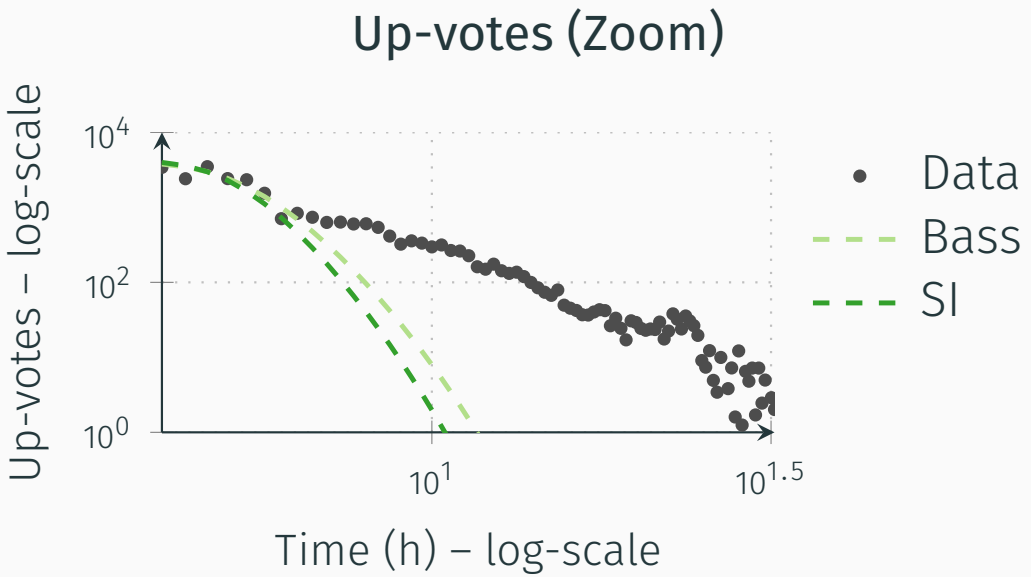


VnC Model: Experiments – Q2. Popularity Decay

Up-vote time-series have a heavy-tail decay

Bass and SI models generate unrealistic exponential decays



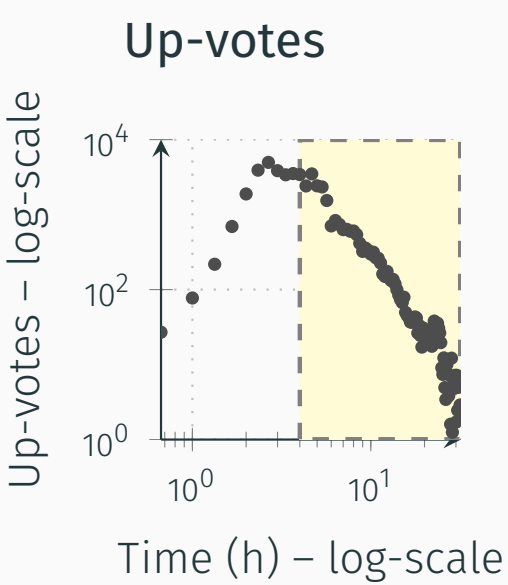


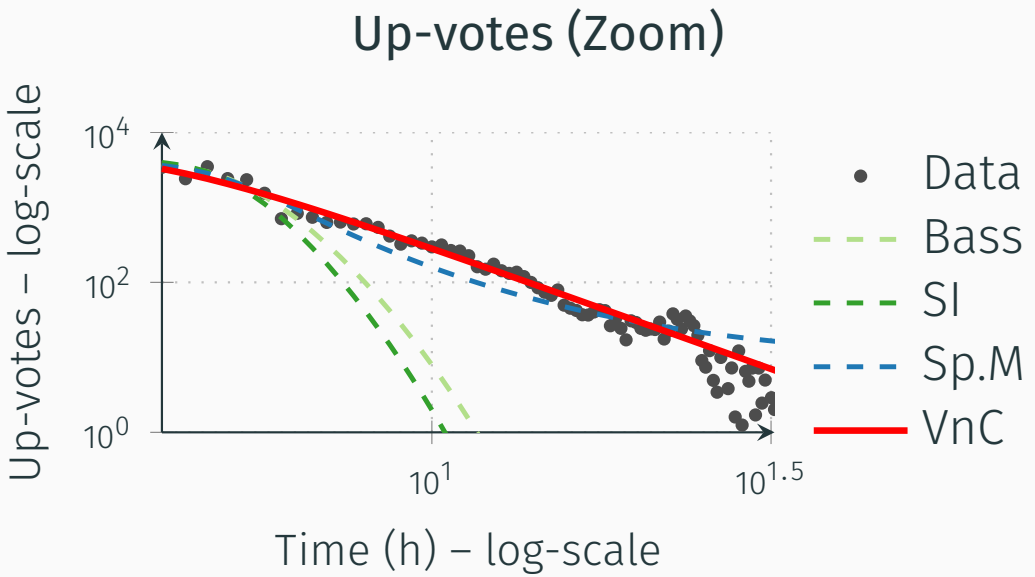
VnC Model: Experiments – Q2. Popularity Decay

Up-vote time-series have a heavy-tail decay

Bass and SI models generate unrealistic exponential decays

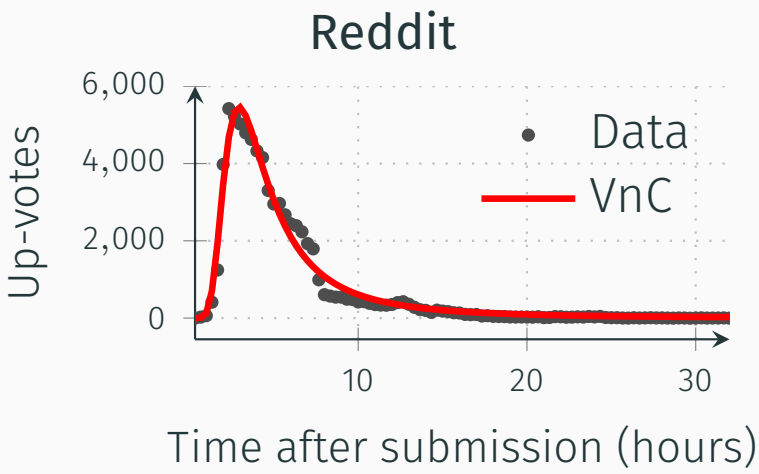
VnC and Spike-M are able to match the heavy tail decay

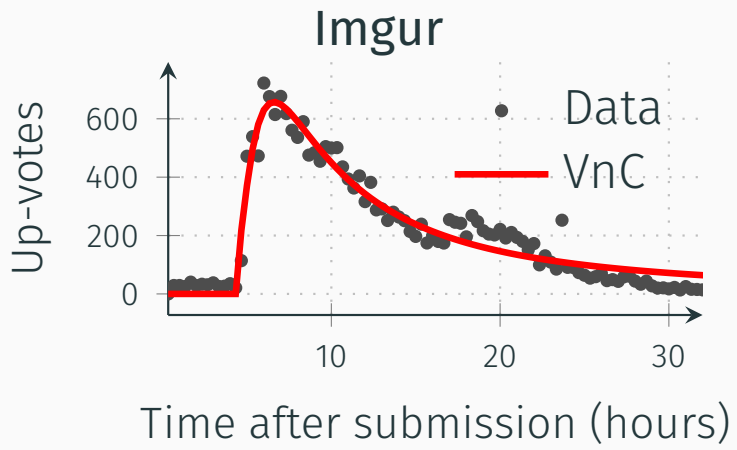


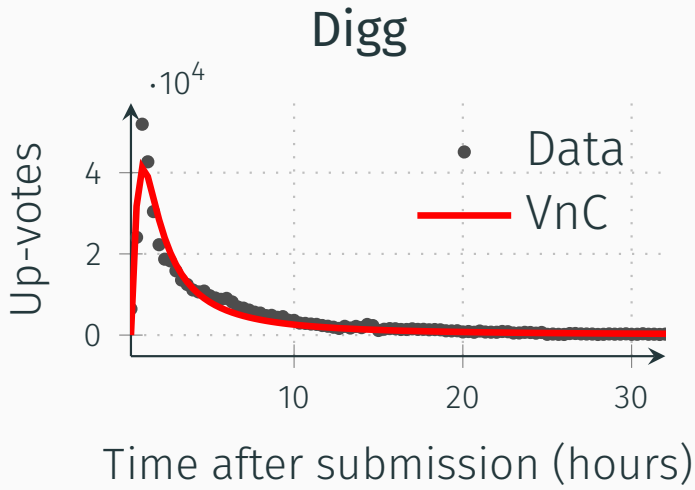


VnC Model: Experiments – Q3. Coevolution

VnC up-vote time-series fits for the most voted submissions in each dataset

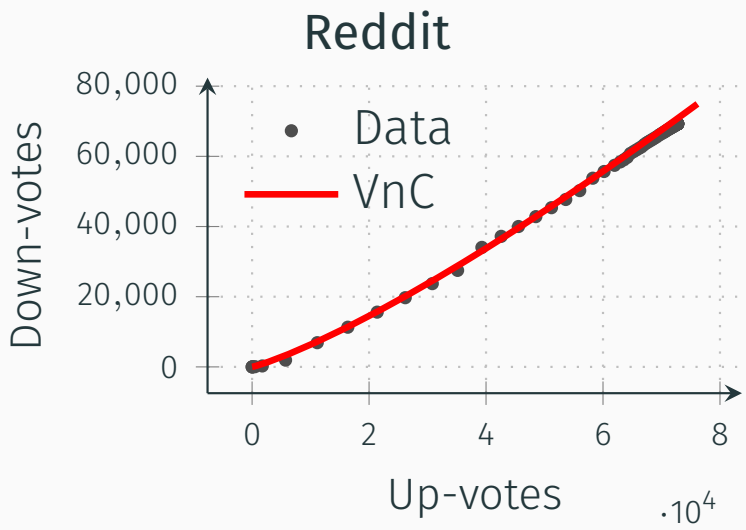


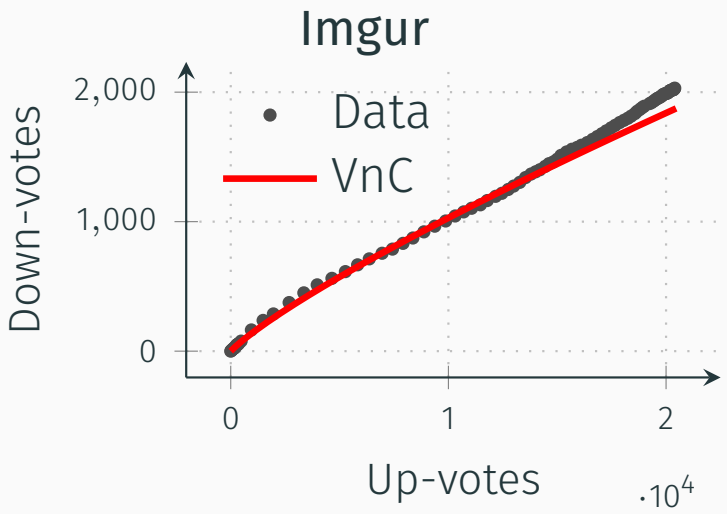




VnC Model: Experiments – Q3. Coevolution

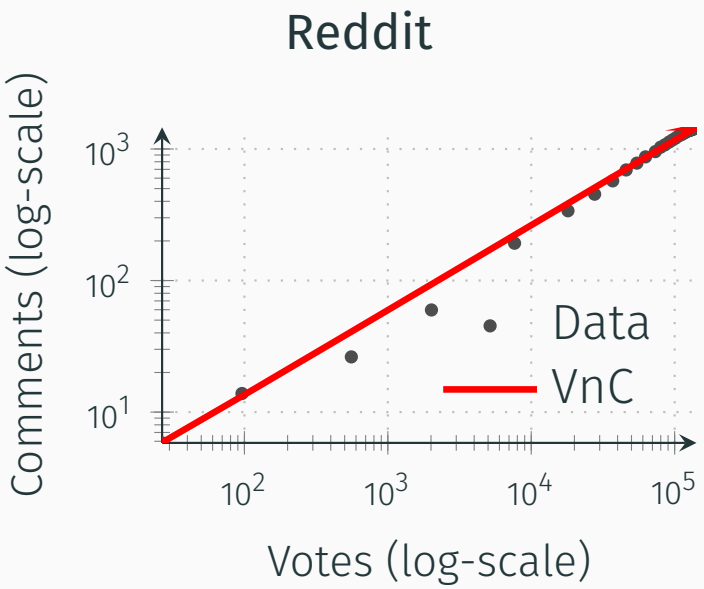
VnC fit for the relationship between up-votes and down-votes received by a submission

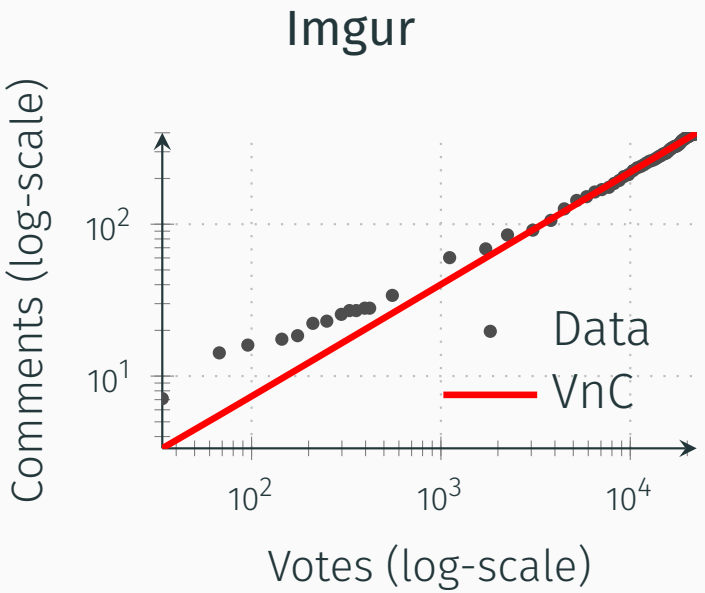




VnC Model: Experiments – Q3. Coevolution

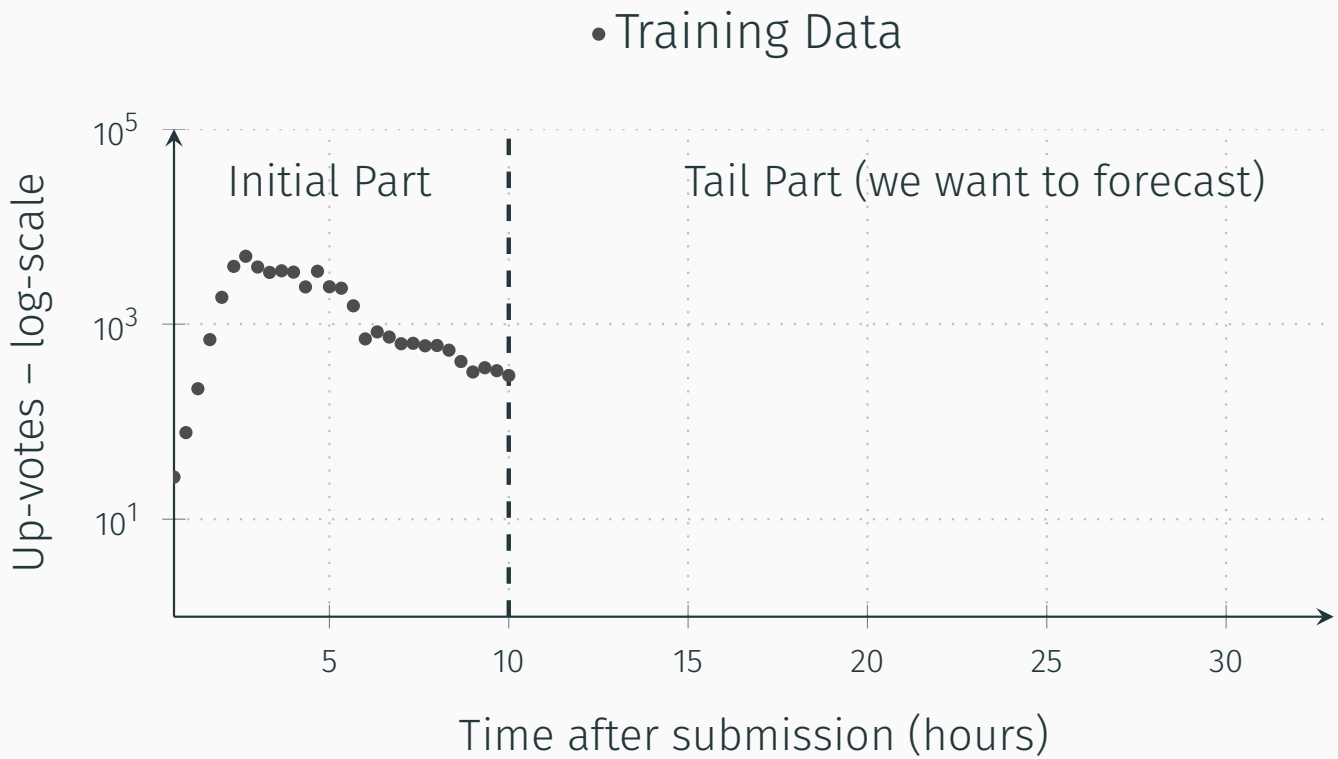
VnC fit for the relationship between total votes (up-votes + down-votes) and comments received by a submission





VnC Model: Applications – Forecasting

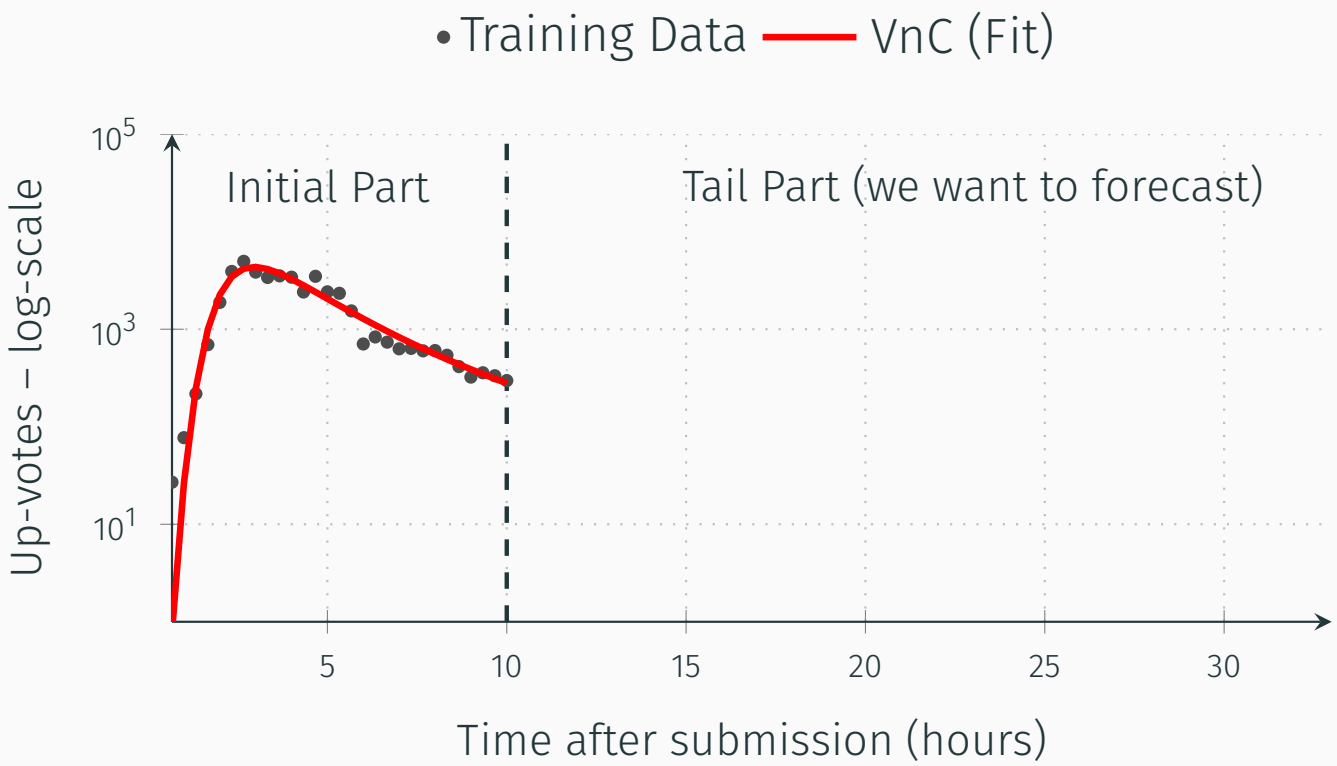
Problem: Given the initial part of a social voting time-series, predict the tail part



VnC Model: Applications – Forecasting

Problem: Given the initial part of a social voting time-series, predict the tail part

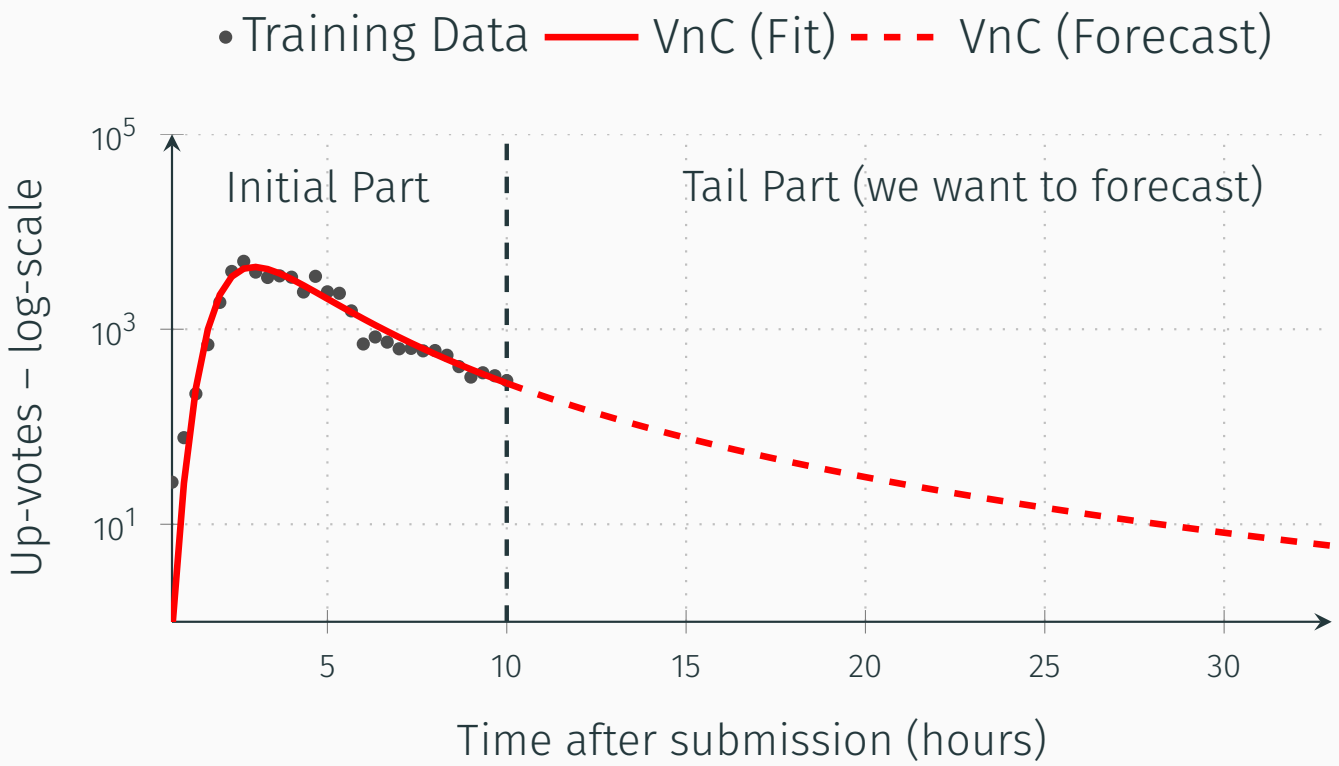
1. Estimate VnC parameters using the initial part



VnC Model: Applications – Forecasting

Problem: Given the initial part of a social voting time-series, predict the tail part

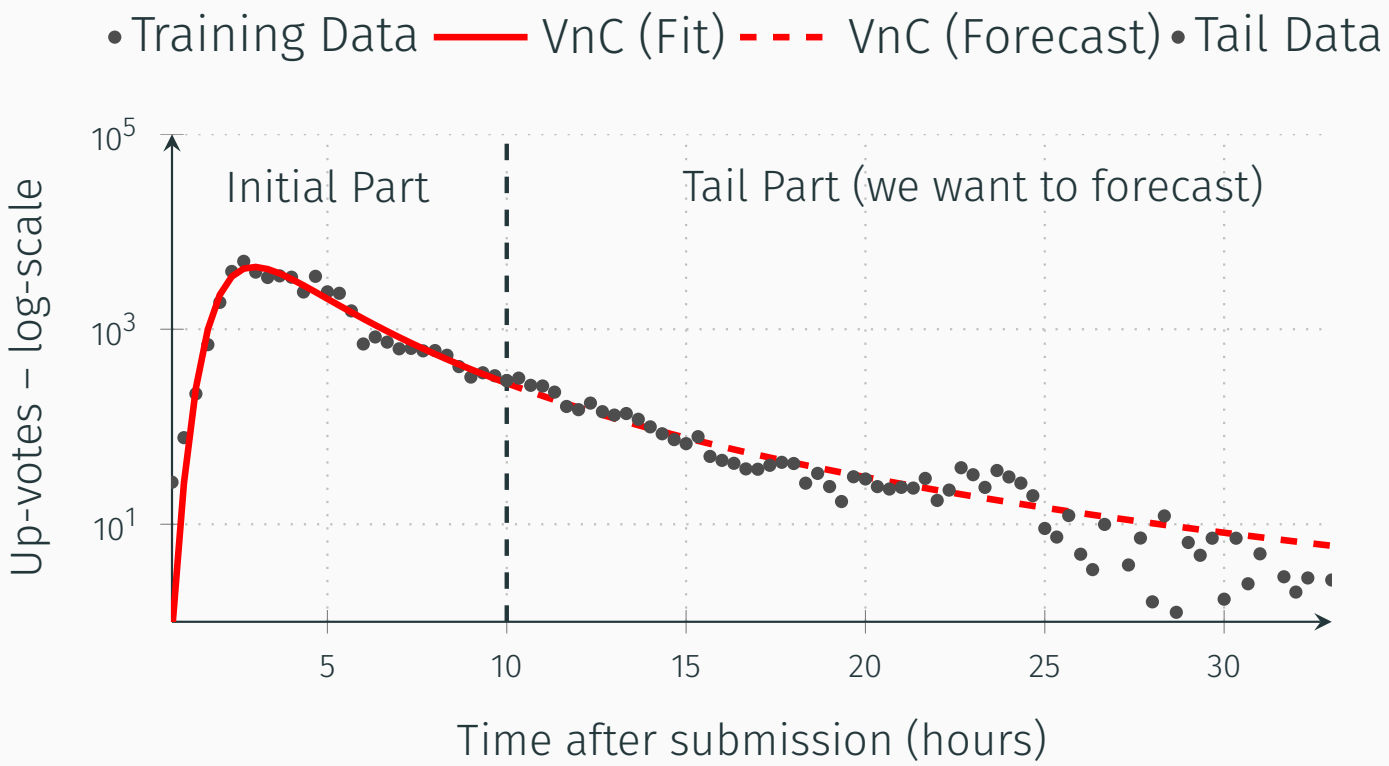
1. Estimate VnC parameters using the initial part
2. Use the parameters to forecast the tail part



VnC Model: Applications – Forecasting

Problem: Given the initial part of a social voting time-series, predict the tail part

1. Estimate VnC parameters using the initial part
2. Use the parameters to forecast the tail part



VnC Model: Applications – Forecasting

Absolute Percetage Error (APE) $= |\text{Actual} - \text{Forecasted}|/(\text{Actual})$

Median APE

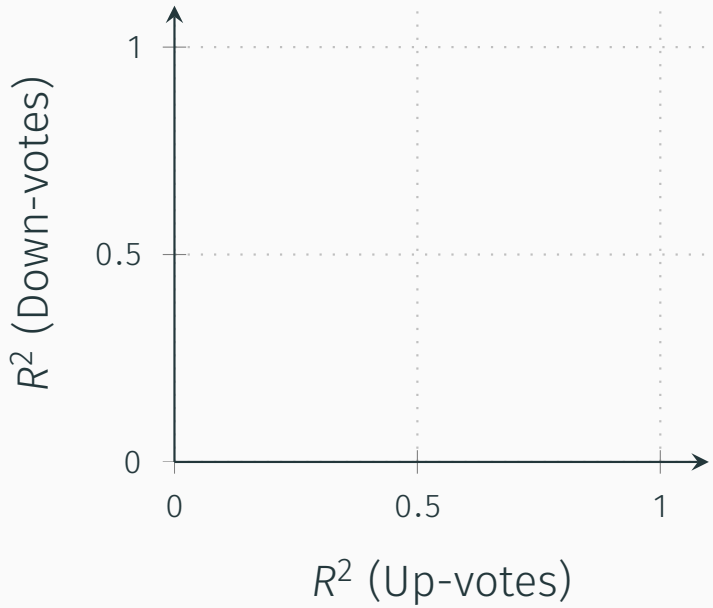
		Bass	SI	Phoenix-R	Spike-M	**VnC**
Reddit	v_+	0.57	0.57	0.82	0.42	**0.39**
	v_-	0.67	0.64	0.86	**0.55**	**0.53**
	c	0.62	0.64	0.86	0.73	**0.39**
Imgur	v_+	**0.69**	**0.65**	0.81	0.98	**0.71**
	v_-	**0.65**	0.68	0.84	0.98	**0.65**
	c	0.59	0.58	0.84	1.15	**0.47**
Digg	v_+	0.87	0.89	0.98	**0.52**	0.77

VnC Model: Applications – Outlier Detection

To detect outliers we use VnC's R^2

- R^2 measures fit accuracy
- R^2 values closer to 1 indicate better fits

R^2 vs. R^2 plot:



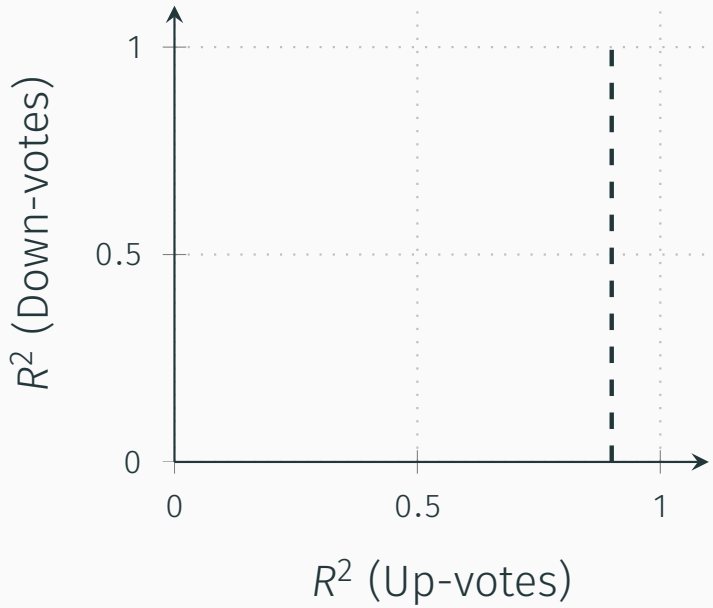
VnC Model: Applications – Outlier Detection

To detect outliers we use VnC's R^2

- R^2 measures fit accuracy
- R^2 values closer to 1 indicate better fits

R^2 vs. R^2 plot:

1. Compute R^2 for the up-vote time-series



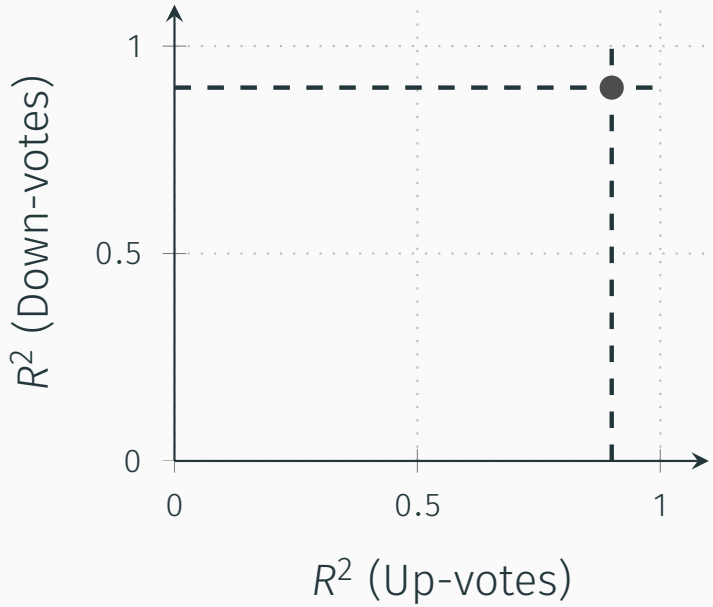
VnC Model: Applications – Outlier Detection

To detect outliers we use VnC's R^2

- R^2 measures fit accuracy
- R^2 values closer to 1 indicate better fits

R^2 vs. R^2 plot:

1. Compute R^2 for the up-vote time-series
2. Compute R^2 for the down-vote time-series



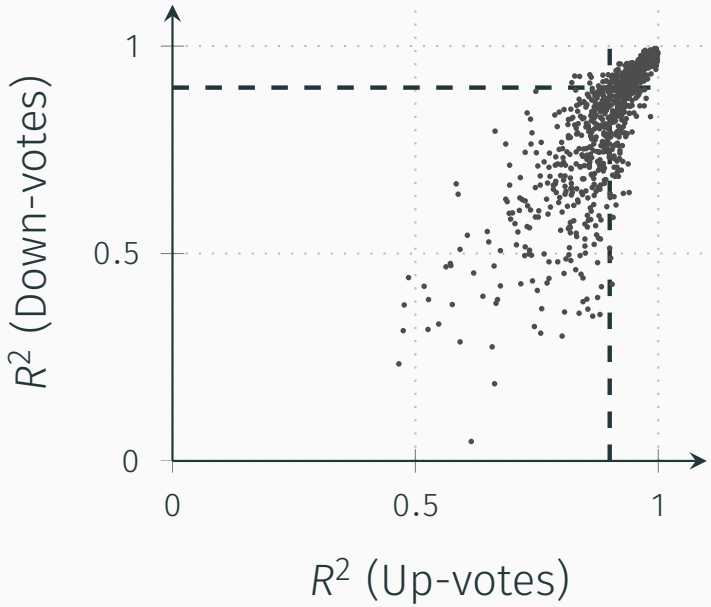
VnC Model: Applications – Outlier Detection

To detect outliers we use VnC's R^2

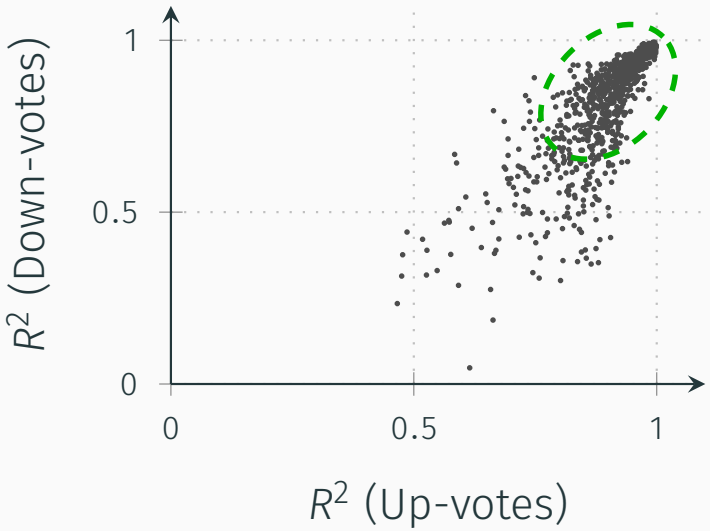
- R^2 measures fit accuracy
- R^2 values closer to 1 indicate better fits

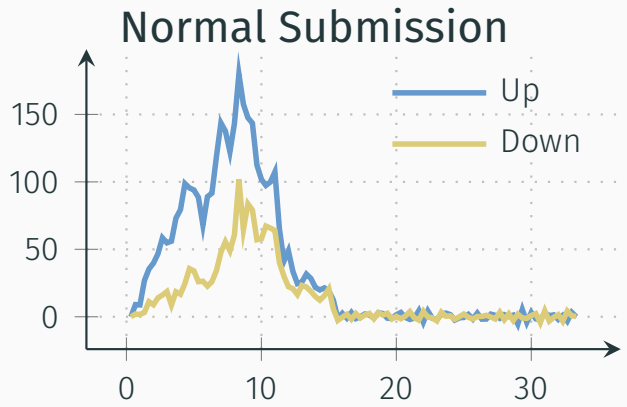
R^2 vs. R^2 plot:

1. Compute R^2 for the up-vote time-series
2. Compute R^2 for the down-vote time-series
3. Repeat for all submissions



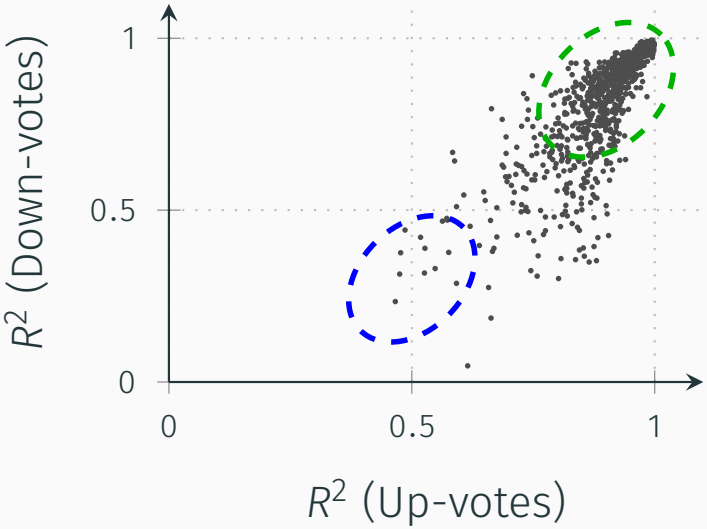
VnC Model: Applications – Outlier Detection

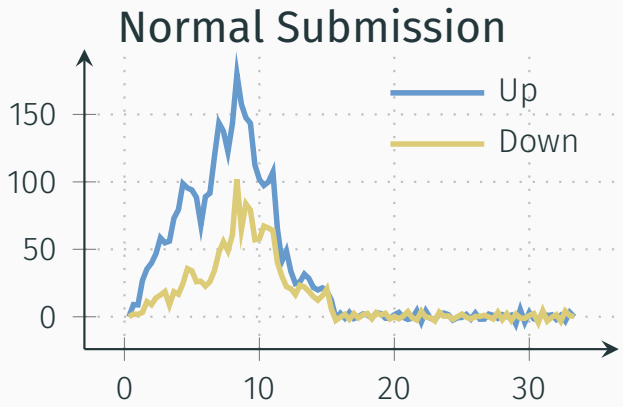




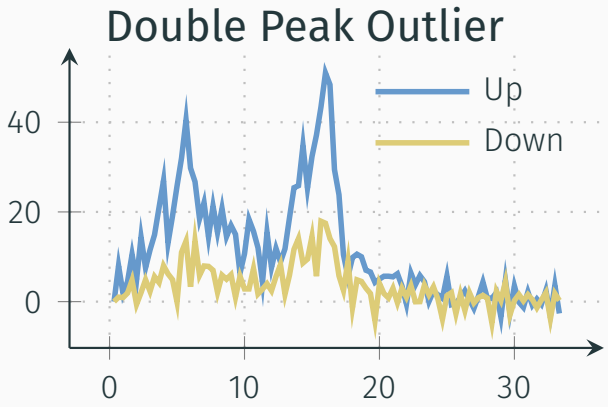
Over 95% of the submissions

VnC Model: Applications – Outlier Detection



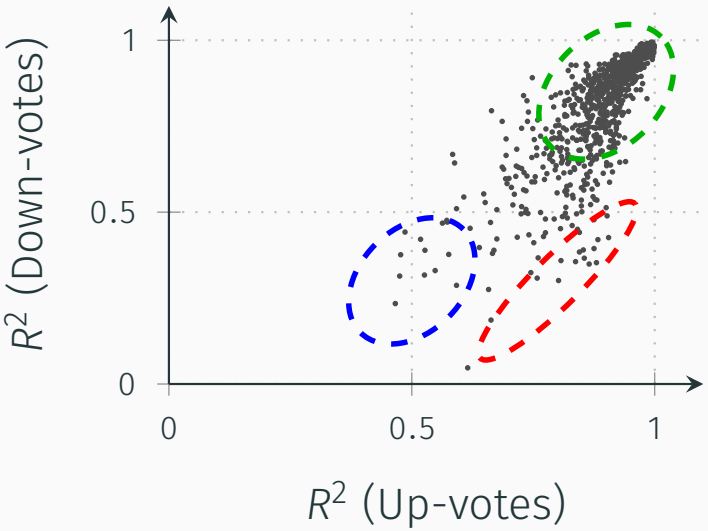


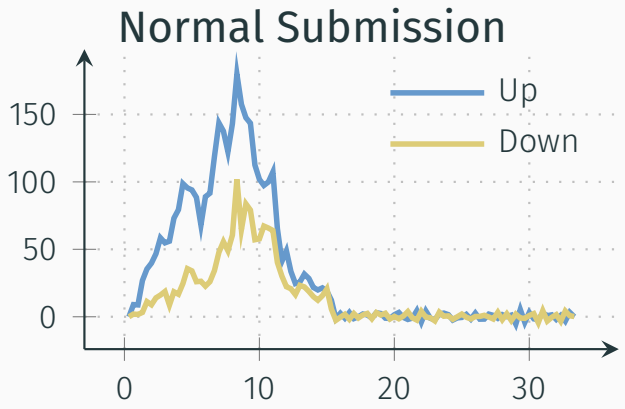
Over 95% of the submissions



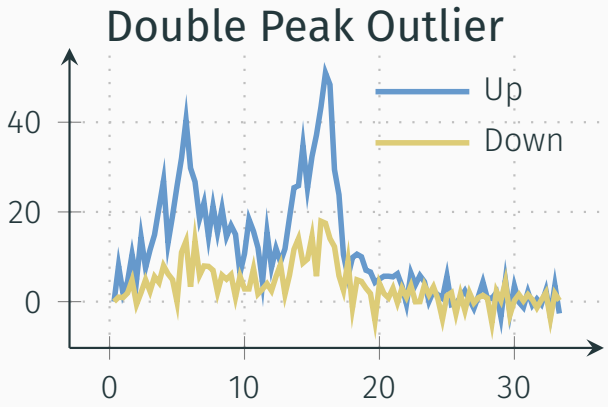
Late-night submissions: 1st peak at night, 2nd peak at morning

VnC Model: Applications – Outlier Detection

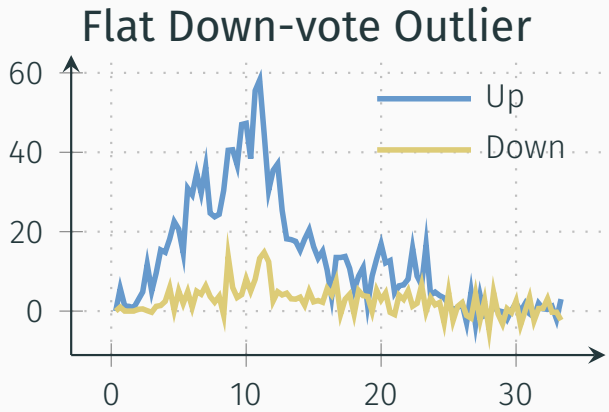




Over 95% of the submissions

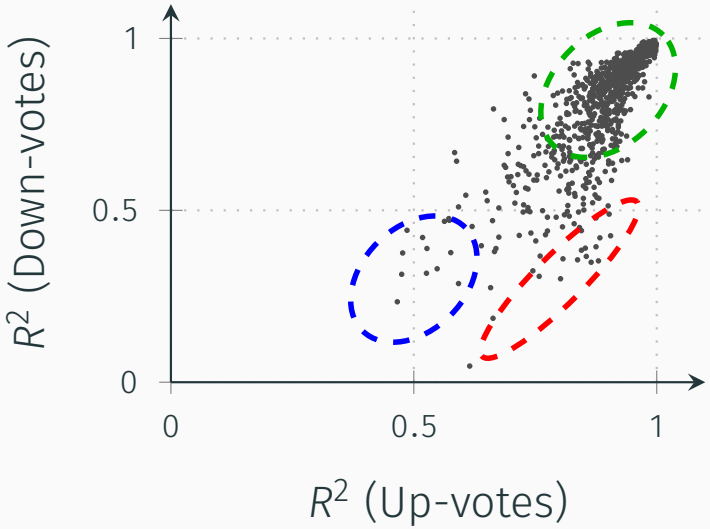


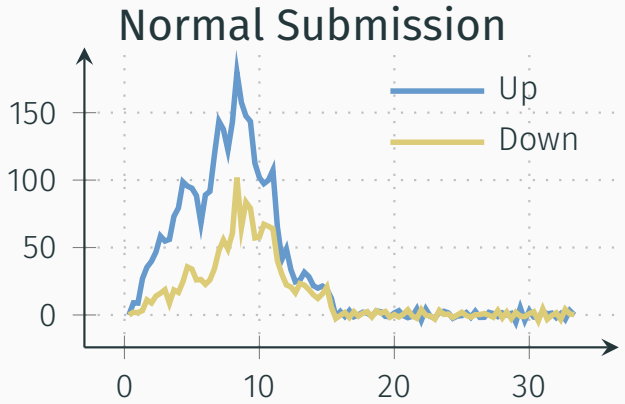
Late-night submissions: 1st peak at night, 2nd peak at morning



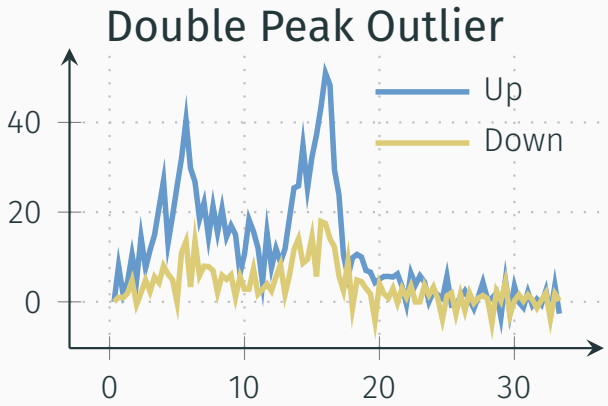
Most are pictures of animals

VnC Model: Applications – Outlier Detection

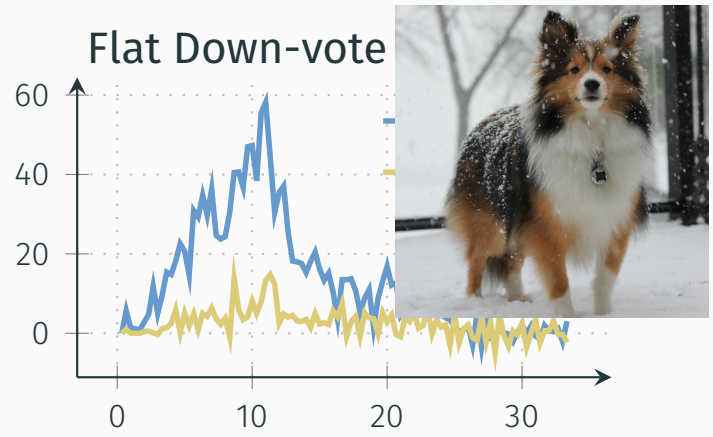




Over 95% of the submissions



Late-night submissions: 1st peak at night, 2nd peak at morning



Most are pictures of animals

VnC Model: Summary

VnC has the following advantages:

- Coevolution Modeling: Describes up-vote, down-vote and comments time-series
- Praticality: Matches data from several social voting web sites
- Usefulness: Forecasting and outlier detection

Outline

MFS-Map: Introduction

How can we automatically find a set of textual annotations to describe a social media image?

Annotations

Gyeongju, Korea,

Temple, Bulguksa

Important Applications:

- Image search (even if we have only visual content)

MFS-Map: Problem Definition

Input: social media image $\mathcal{I}$, consisting of:

- An image I (i.e. a matrix of pixels)
- A set of textual tags $L = \{\ell_1, \ell_2, \cdots\}$

Output: set of textual image annotations $A = \{a_1, a_2, \cdots\}$

Note: tags $\neq$ annotations

- Tags: subjective, noisy, missing...
- Annotations: describe the images' visual content

Input Image, I

Tags, $L = \{\ell_1, \ell_2, \cdots\}$

crane, gru, sunset,
hdr, tramonto, $\cdots$

Annotations, $A = \{a_1, a_2, \cdots\}$

sky, clouds,
sunset, $\cdots$

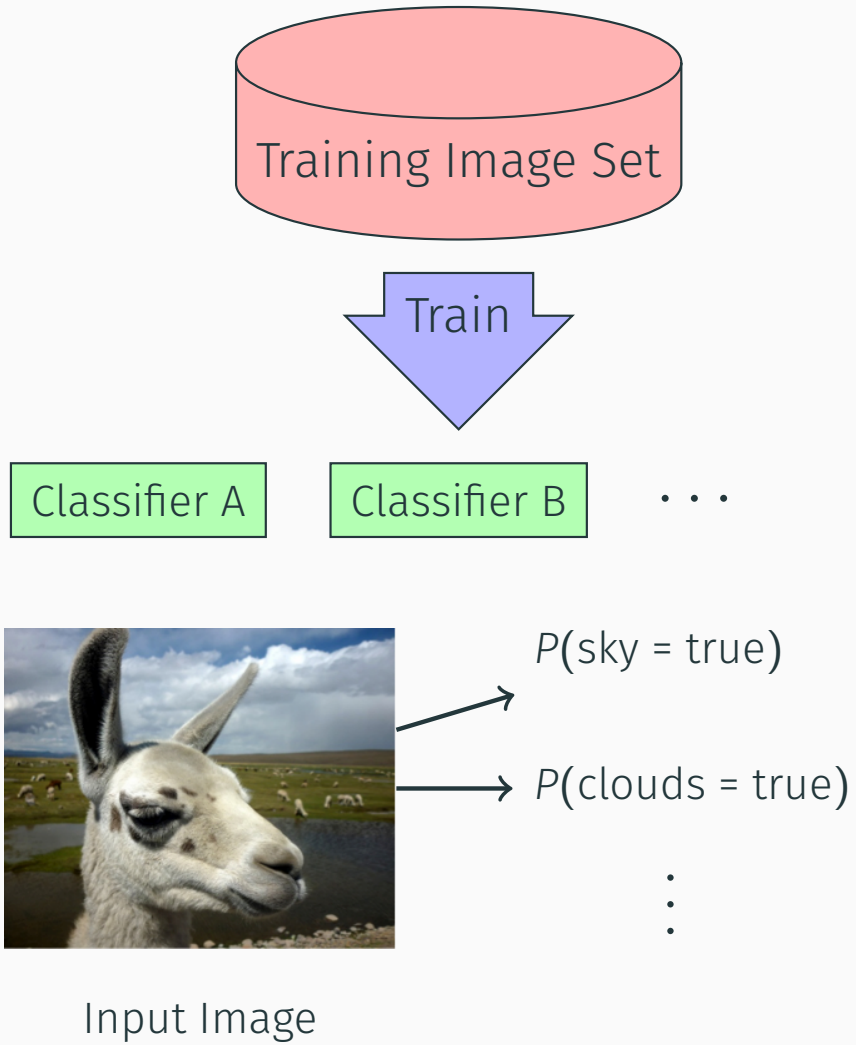
MFS-Map: Baseline Solution

Baseline Solution:

1. Train a binary classifier for each possible annotation
2. Given an input image, predict the relevance of each annotation

Problems:

- Class imbalance
- High computational cost



MFS-Map

The MFS-Map is divided into three steps:

1. Feature Discretization
    - **Goal:** learn a map function that maps images to sets of discrete feature items
2. Rule Generation
    - **Goal:** learn rules that associate visual features and tags to annotations
3. Prediction of Annotation Relevance
    - **Goal:** use the rules to assign annotations to images

MFS-Map: Visual Feature Discretization

MFS-Map: Discretization | Rule Generation | Annotation

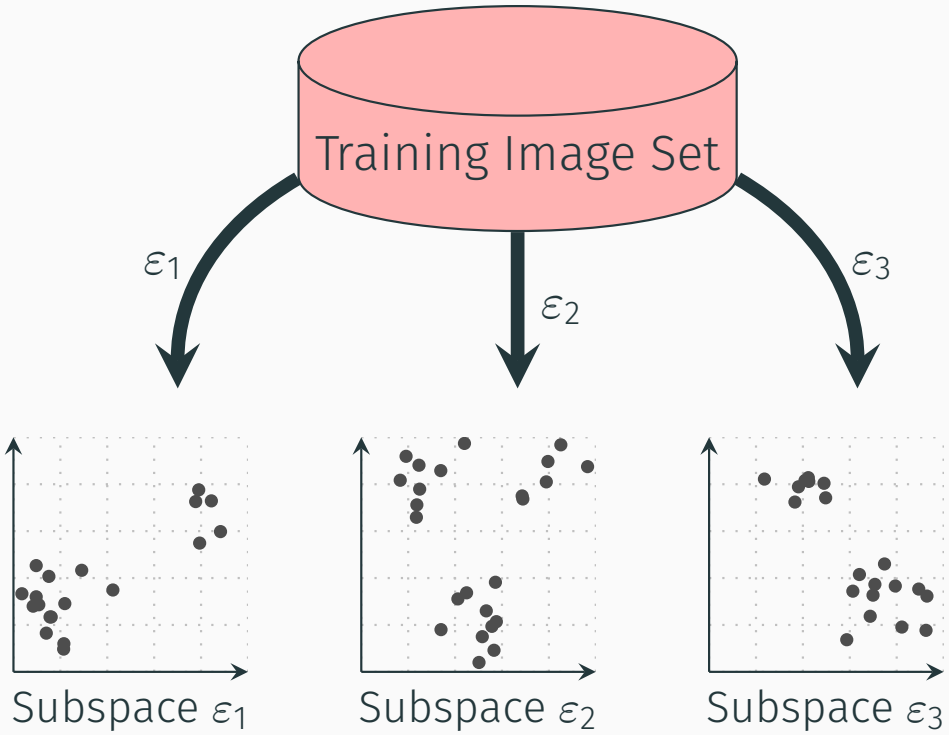
Goal: learn a map function $m(I) \rightarrow \{f_1, f_2, \cdots, f_k\}$ that maps the input image I into a set of discrete feature items $f_i \in \mathbb{Z}$.

Algorithm

Input: Training set of images and feature extraction functions $\{\varepsilon_1, \varepsilon_2, \cdots\}$.

For each extractor ε_i:

1. Apply ε_i to all images



MFS-Map: Visual Feature Discretization

MFS-Map: Discretization | Rule Generation | Annotation

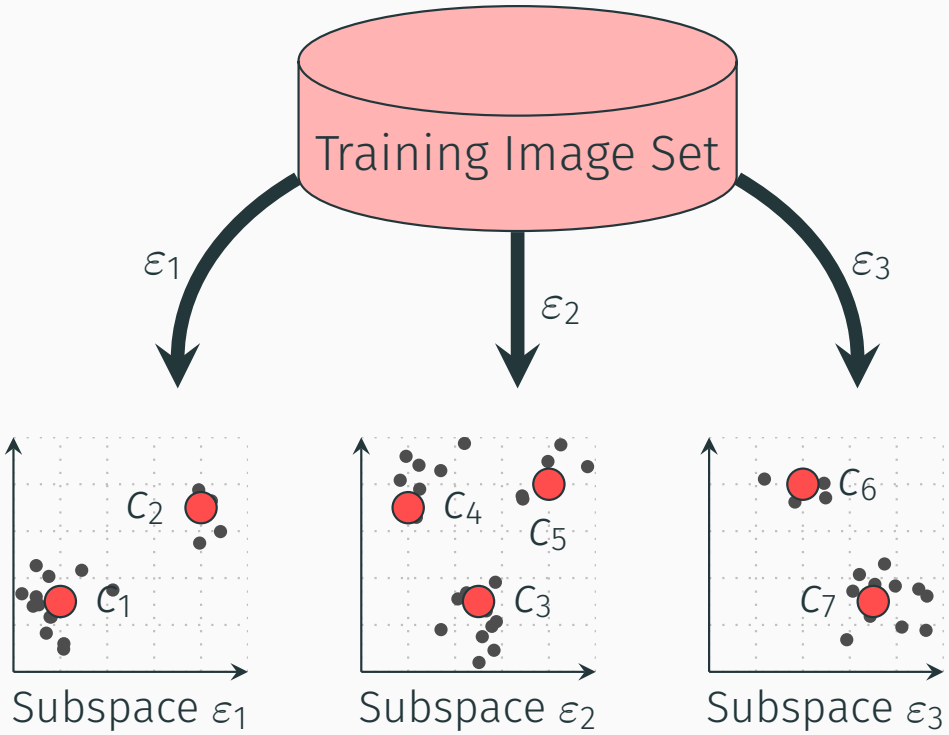
Goal: learn a map function $m(I) \rightarrow \{f_1, f_2, \cdots, f_k\}$ that maps the input image I into a set of discrete feature items $f_i \in \mathbb{Z}$.

Algorithm

Input: Training set of images and feature extraction functions $\{\varepsilon_1, \varepsilon_2, \cdots\}$.

For each extractor ε_i:

1. Apply ε_i to all images
2. Apply k-means and store the centroids

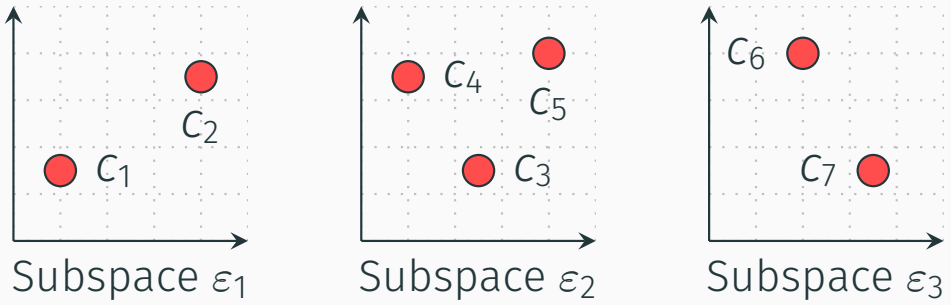


MFS-Map: Visual Feature Discretization

MFS-Map: Discretization | Rule Generation | Annotation

The map function assigns a feature item to each feature vector representing the image I.

For each extractor ε_i:

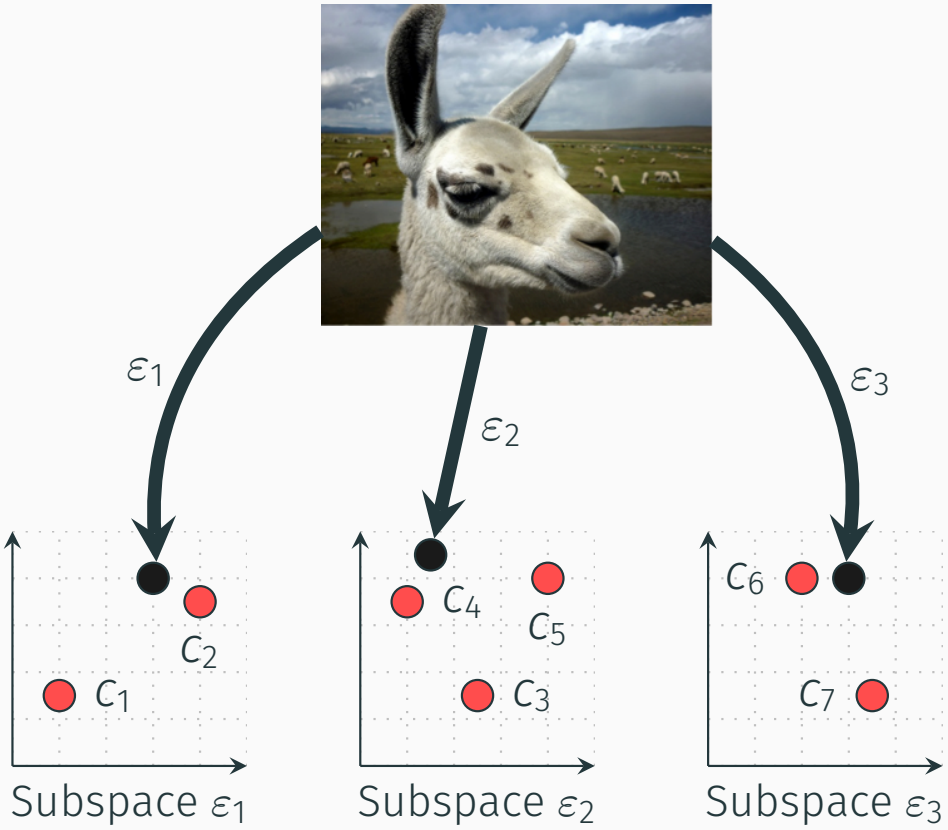


MFS-Map: Visual Feature Discretization

The map function assigns a feature item to each feature vector representing the image I.

For each extractor ε_i:

1. Apply ε_i to the input image

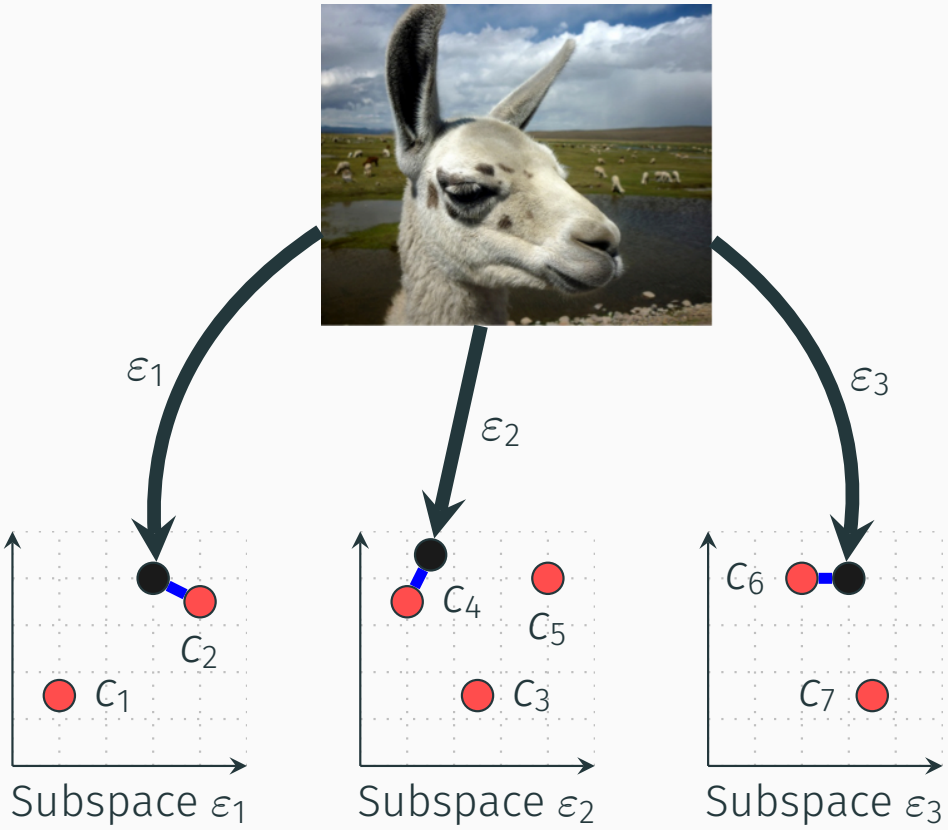


MFS-Map: Visual Feature Discretization

The map function assigns a feature item to each feature vector representing the image I.

For each extractor ε_i:

1. Apply ε_i to the input image
2. Find the nearest centroid



MFS-Map: Visual Feature Discretization

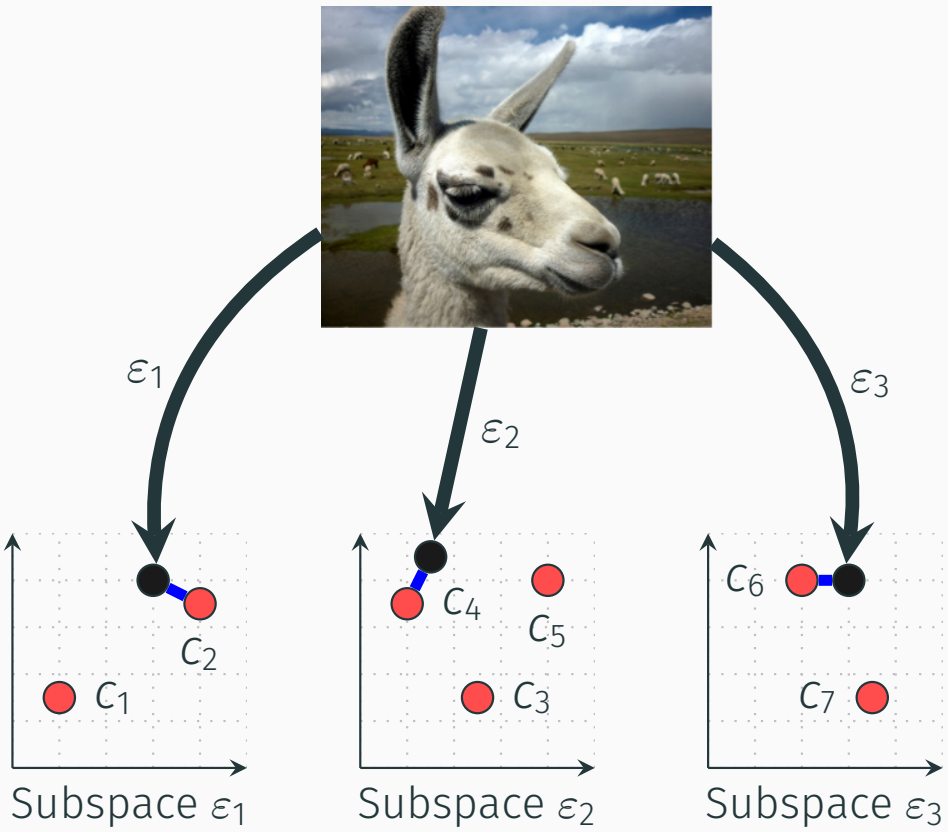
MFS-Map: Discretization > Rule Generation > Annotation

The map function assigns a feature item to each feature vector representing the image I.

For each extractor ε_i:

1. Apply ε_i to the input image
2. Find the nearest centroid
3. Use the centroid labels as feature items

$m(I) \rightarrow \{c_2, c_4, c_6\}$



MFS-Map: Rule Generation

MFS-Map: Discretization > Rule Generation > Annotation

Before mining rules, MFS-Map generates an itemset representation of the images in the training set

Itemset Representation: $\{\underbrace{c_1, c_2, c_3, \cdots}_{\text{cluster centroids}} \overbrace{\ell_1, \ell_2, \ell_3, \cdots}^{\text{tags}} \underbrace{a_1, a_2, a_3, \cdots}_{\text{annotations}}\}$

Input Image, I

Tags

crane, gru, hdr

Annotations

sky, clouds, sunset

$m(I)$

$\{c_1, c_2, \text{crane}, \text{gru}, \text{hdr}, \text{sky}, \text{clouds}, \text{sunset}\}$

MFS-Map: Rule Generation

MFS-Map: Discretization | Rule Generation | Annotation

Input: Image itemsets

Itemsets (Training)

$\{c_2, c_3, c_6, \text{crane}, \text{gru}, \text{hdr}, \text{clouds}, \text{sky}, \text{sunset}\}$
$\{c_1, c_4, c_7, \text{peru}, \text{travelanimal}, \text{landscape}\}$
$\{c_2, c_4, c_5, \text{rocks}, \text{sunset}, \text{clouds}\}$
...

MFS-Map: Rule Generation

MFS-Map: Discretization | Rule Generation | Annotation

Input: Image itemsets

1. Generate rules in the format:
   - (cluster) $\rightarrow$ (annot.)
   - (tag) $\rightarrow$ (annot.)

Itemsets (Training)

$\{c_2, c_3, c_6, \text{crane}, \text{gru}, \text{hdr}, \text{clouds}, \text{sky}, \text{sunset}\}$
$\{c_1, c_4, c_7, \text{peru}, \text{travelanimal}, \text{landscape}\}$
$\{c_2, c_4, c_5, \text{rocks}, \text{sunset}, \text{clouds}\}$
...

Rules

$\{c_2\} \rightarrow \{\text{clouds}\}$
$\{c_2\} \rightarrow \{\text{bird}\}$
$\{\text{hdr}\} \rightarrow \{\text{sunset}\}$
...

MFS-Map: Rule Generation

MFS-Map: Discretization > Rule Generation > Annotation

Input: Image itemsets

1. Generate rules in the format:
   - (cluster) $\rightarrow$ (annot.)
   - (tag) $\rightarrow$ (annot.)
2. Compute confidence to measure rule "quality" Details

Itemsets (Training)

$\{c_2, c_3, c_6, \text{crane}, \text{gru}, \text{hdr}, \text{clouds}, \text{sky}, \text{sunset}\}$
$\{c_1, c_4, c_7, \text{peru}, \text{travelanimal}, \text{landscape}\}$
$\{c_2, c_4, c_5, \text{rocks}, \text{sunset}, \text{clouds}\}$
...

Rules

$\{c_2\} \rightarrow \{\text{clouds}\}$ $(\text{conf} = 0.67)$
$\{c_2\} \rightarrow \{\text{bird}\}$ $(\text{conf} = 0.04)$
$\{\text{hdr}\} \rightarrow \{\text{sunset}\}$ $(\text{conf} = 0.78)$
...

MFS-Map: Rule Generation

MFS-Map: Discretization | Rule Generation | Annotation

Input: Image itemsets

1. Generate rules in the format:
   - (cluster) $\rightarrow$ (annot.)
   - (tag) $\rightarrow$ (annot.)
2. Compute confidence to measure rule "quality" Details
3. Discard rules if confidence is smaller than minConf

Itemsets (Training)

$\{c_2, c_3, c_6, \text{crane}, \text{gru}, \text{hdr}, \text{clouds}, \text{sky}, \text{sunset}\}$
$\{c_1, c_4, c_7, \text{peru}, \text{travelanimal}, \text{landscape}\}$
$\{c_2, c_4, c_5, \text{rocks}, \text{sunset}, \text{clouds}\}$
...

Rules

$\{c_2\} \rightarrow \{\text{clouds}\}$ $(\text{conf} = 0.67)$
~~$\{c_2\} \rightarrow \{\text{bird}\}$ $(\text{conf} = 0.04)$~~
$\{\text{hdr}\} \rightarrow \{\text{sunset}\}$ $(\text{conf} = 0.78)$
...

MFS-Map: Annotation

MFS-Map: Discretization > Rule Generation > Annotation

1. Extract itemset

Input Image, I

$\{c_2, c_4, \text{hdr}\}$

Rules

$\{c_1\} \rightarrow \{\text{animal}\}$	$(\text{conf} = 0.67)$
$\{c_4\} \rightarrow \{\text{sunset}\}$	$(\text{conf} = 0.82)$
$\{c_2\} \rightarrow \{\text{city}\}$	$(\text{conf} = 0.42)$
$\{\text{hdr}\} \rightarrow \{\text{sunset}\}$	$(\text{conf} = 0.93)$

MFS-Map: Annotation

MFS-Map: Discretization > Rule Generation > Annotation

1. Extract itemset
2. Find matching rules

Input Image, I

$\{c_2, c_4, \text{hdr}\}$

Rules

~~$\{c_1\} \rightarrow \{\text{animal}\}$~~	~~$(\text{conf} = 0.67)$~~
$\{c_4\} \rightarrow \{\text{sunset}\}$	$(\text{conf} = 0.82)$
$\{c_2\} \rightarrow \{\text{city}\}$	$(\text{conf} = 0.42)$
$\{\text{hdr}\} \rightarrow \{\text{sunset}\}$	$(\text{conf} = 0.93)$

MFS-Map: Annotation

MFS-Map: Discretization > Rule Generation > Annotation

1. Extract itemset
2. Find matching rules
3. Compute mean confidence

Input Image, I

$\{c_2, c_4, \text{hdr}\}$

Rules

~~$\{c_1\} \rightarrow \{\text{animal}\}$~~	~~$(\text{conf} = 0.67)$~~
$\{c_4\} \rightarrow \{\text{sunset}\}$	$(\text{conf} = 0.82)$
$\{c_2\} \rightarrow \{\text{city}\}$	$(\text{conf} = 0.42)$
$\{\text{hdr}\} \rightarrow \{\text{sunset}\}$	$(\text{conf} = 0.93)$

sunset	(score = 0.88)
city	(score = 0.72)
animal	(score = 0.00)

MFS-Map: Experiments

Datasets (Flickr images):

1. **MIR-Flickr:** 25,000 images / 25 annotations
2. **Image CLEF:** 18,000 images / 98 annotations

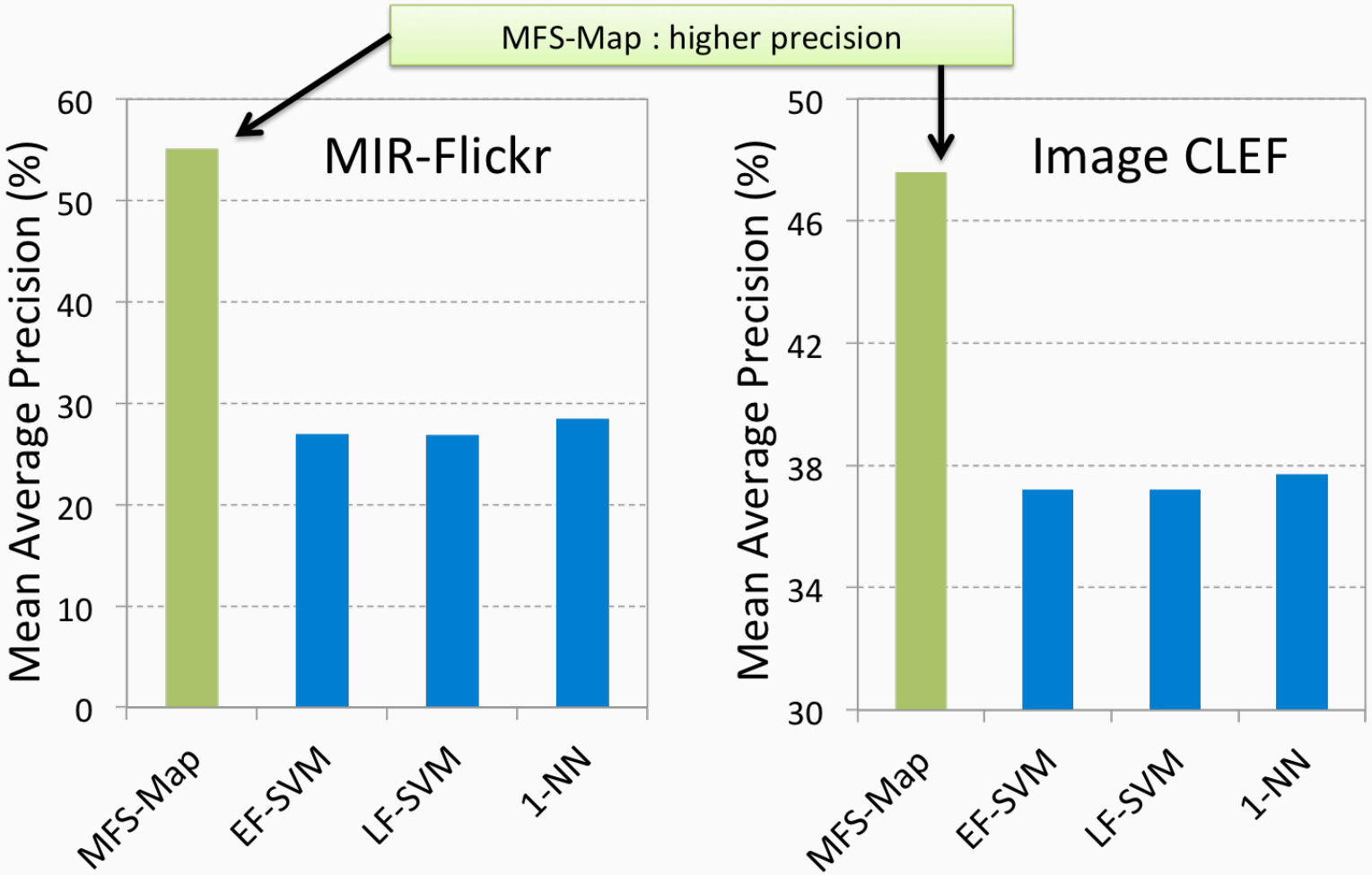
Annotation methods:

- **MFS-Map** (proposed method)
- **EF-SVM:** Binary classifier + Early Fusion
- **LF-SVM:** Binary classifier + Late Fusion
- **1-NN:** Nearest neighbors

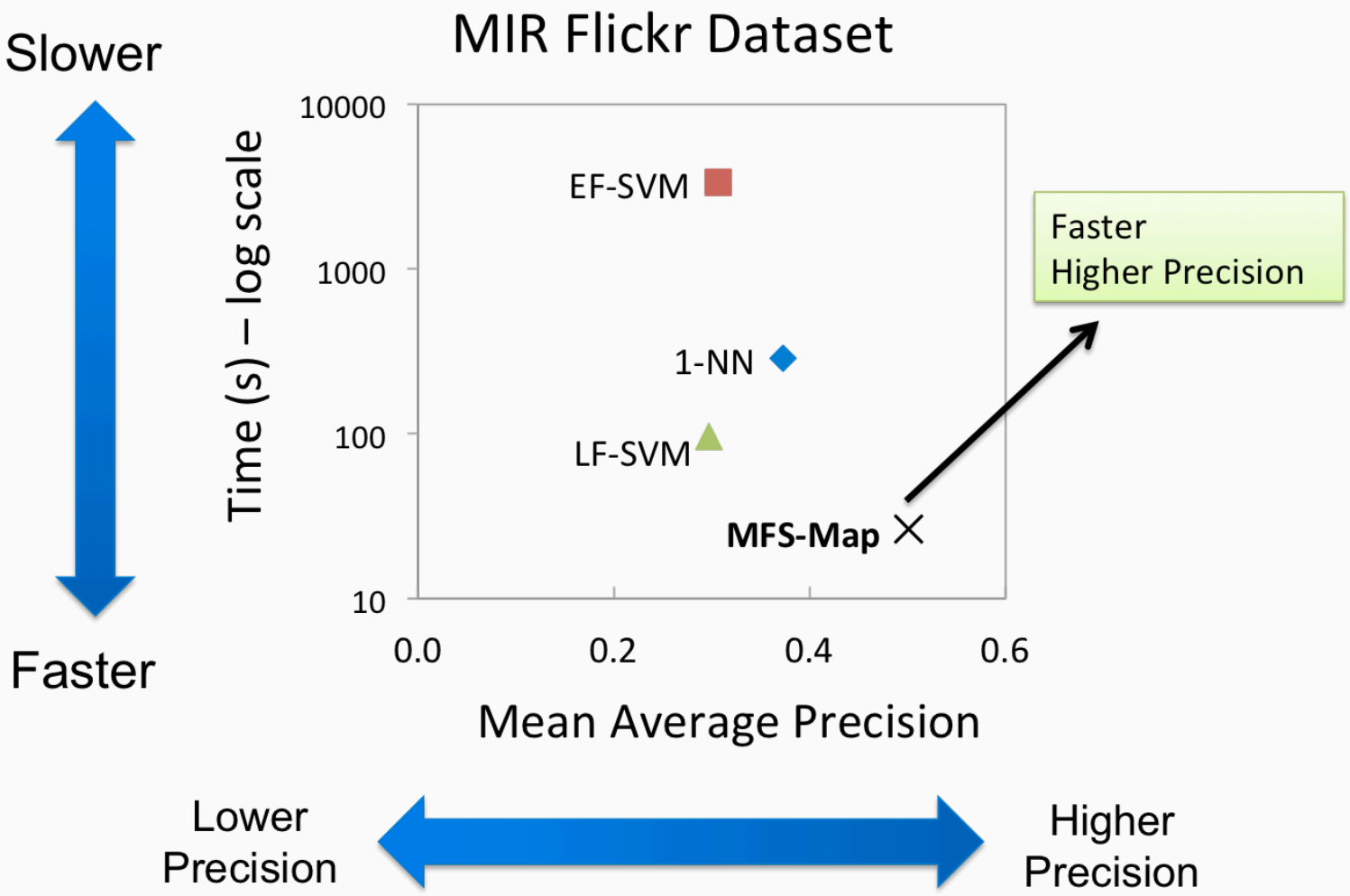
Features:

- Visual: SIFT, Gist, SFTA (texture), RGB histogram
- Flickr user tags

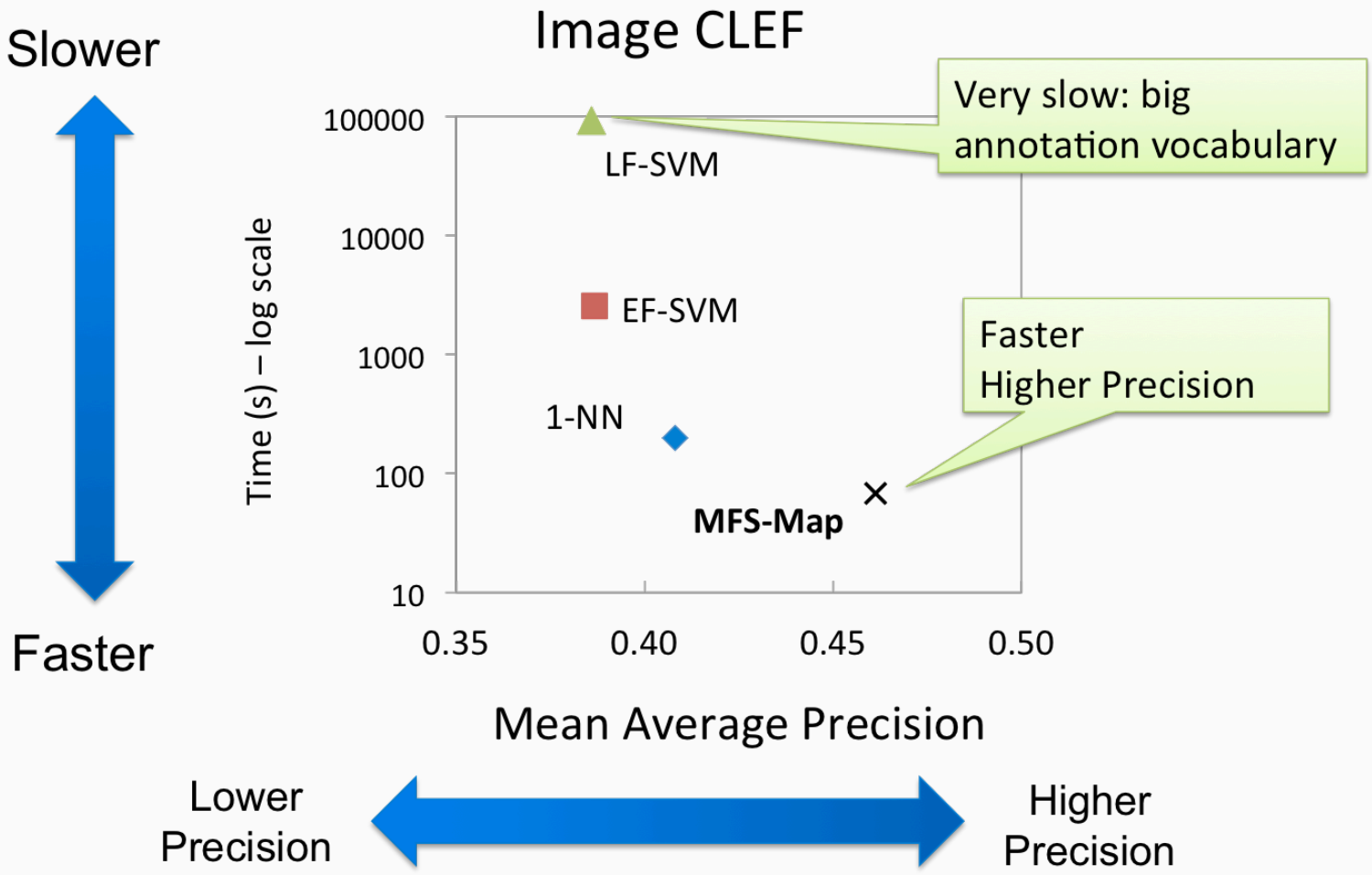
MFS-Map: Experiments



MFS-Map: Experiments



MFS-Map: Experiments



MFS-Map: Summary

MFS-Map highlights:

1. **Feature Combination:** is able to annotate images using both visual and textual features
2. **Scalability:** faster than techniques based on classifiers
3. **Precision:** competitive precision results in publicly available datasets

Outline

Conclusions

Thesis

Data from social media services **obey patterns** that can be explored to design specialized methods for mining the raw data and improve the accuracy of prediction, anomaly detection and forecasting tasks when compared to traditional non-specialized data mining methods.

Patterns:

1. IAT distribution:
   - Heavy-tails;
   - Bimodality;
   - Periodic spikes;
   - Correlation
2. User interactions:
   - Coevolution patterns;
3. Social media images:
   - Visual content + textual tags;

Conclusions

Thesis

Data from social media services obey patterns that can be explored **to design specialized methods** for mining the raw data and improve the accuracy of prediction, anomaly detection and forecasting tasks when compared to traditional non-specialized data mining methods.

Models/Method:

1. Act-M Model:
   - Bot-detection based only on time-stamp data
2. VnC Model:
   - Forecasting, anomaly detection
3. MFS-Map Method:
   - Annotation of images combining visual and textual information

Publications

Main publications:

- **Act-M Model:**
  - Conference paper: **ACM KDD**
  - Journal paper: **ACM TKDD**
- **VnC Model:**
  - Conference paper: **IEEE ICDM**
- **MFS-Map Method:**
  - Conference paper: **ACM SAC**

Collaborations:

- Anomaly detection (VolTime method):
  - Conference paper: **SIAM SDM**
- Computer vision (fire and lung disease detection):
  - Conference papers: **ACM SAC**, **ICEIS** and **CBMS**

Thank you!

Mining user activity data in social media services

Alceu Ferraz Costa

Supervisor: Prof. Dr. Agma Juci Machado Traina

Co-supervisor: Prof. Dr. Christos Faloutsos

Funding:

FAPESP

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

CAPES

VnC Model: Reaction Probability (Details)

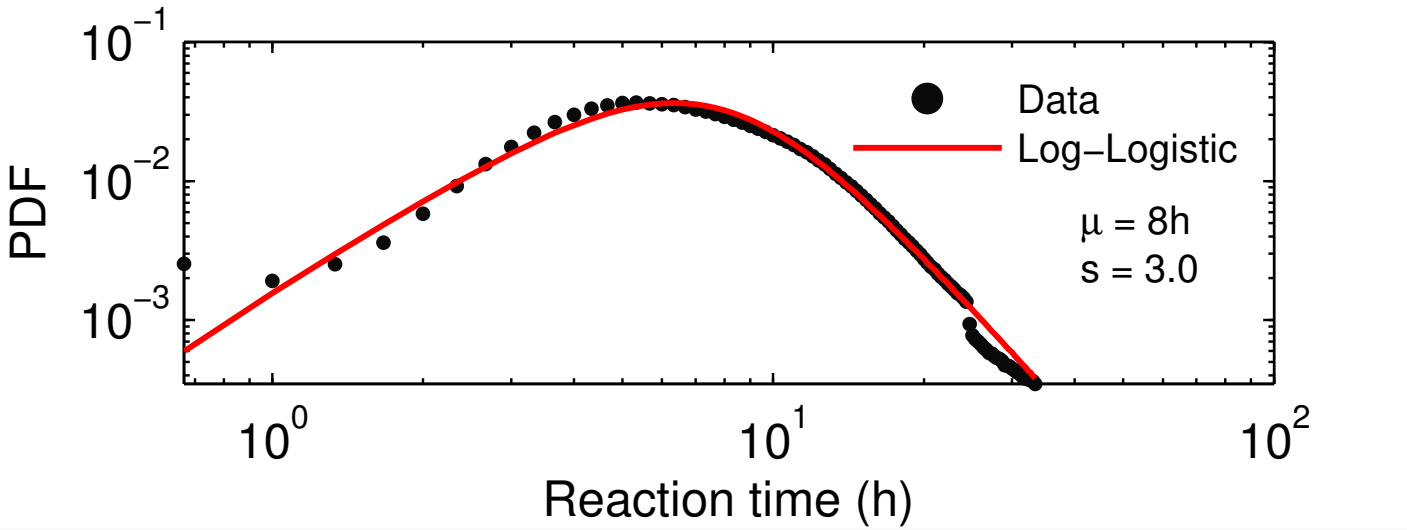
Definition (Reaction Time)

The reaction time corresponds to the time interval between the instant in which a submission is created and the instant in which an interaction (vote or comment) occurs.

Data: Reaction times of up-votes, down-votes and comments from the Reddit, Imgur and Digg datasets.

Log-Logistic PDF

$$f(x) = \frac{(s/\mu)(x/\mu)^{s-1}}{(1+(x/\mu)^s)^2}$$



VnC Model: Fit Accuracy (Details)

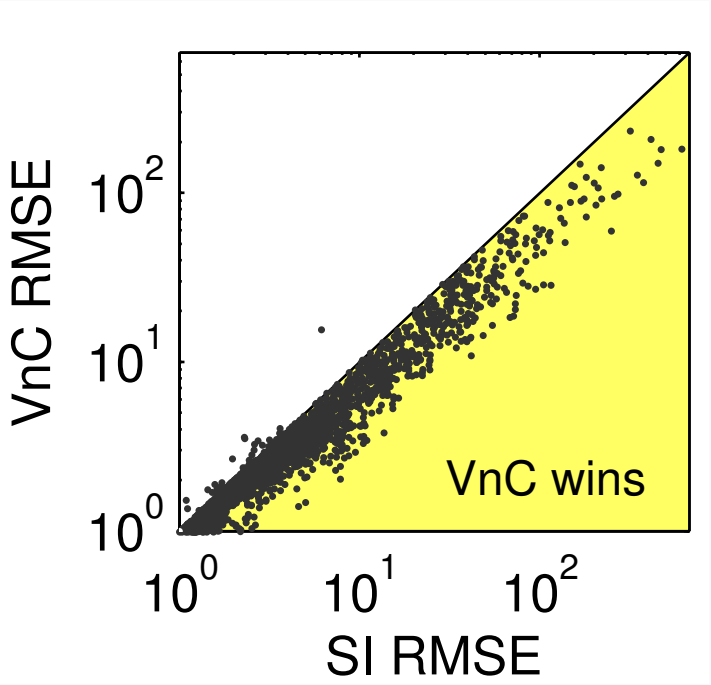
Root-Mean-Square Error

- Lower RMSE indicates a better fit
- Points below the diagonal: time-series that were best fit by VnC

Is VnC more accurate than:

- SI Model? **Yes**
- Bass Model?
- Phoenix-R?
- Spike-M?

VnC vs. SI Model



VnC is more accurate than **SI model** for **99%** of the submissions

VnC Model: Fit Accuracy (Details)

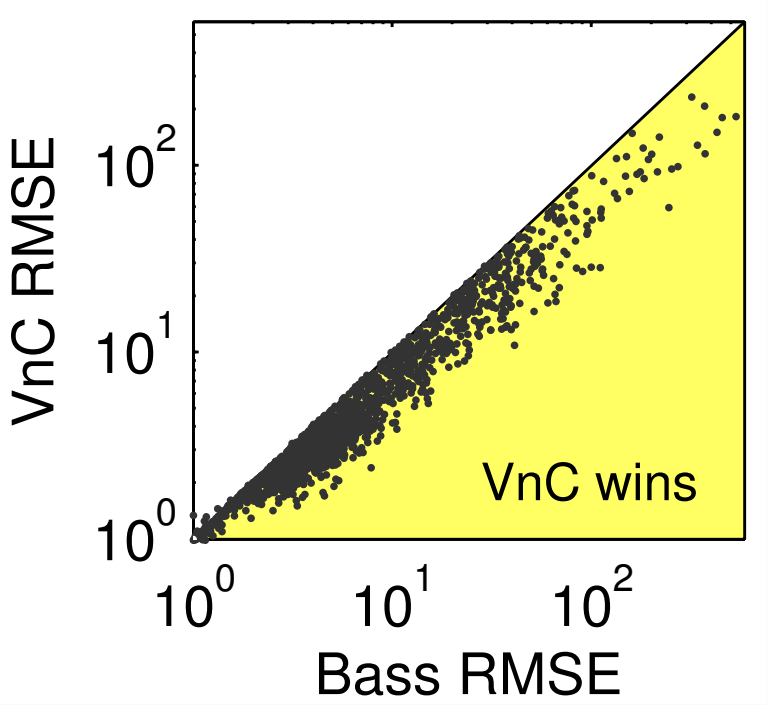
Root-Mean-Square Error

- Lower RMSE indicates a better fit
- Points below the diagonal: time-series that were best fit by VnC

Is VnC more accurate than:

- SI Model? **Yes**
- Bass Model? **Yes**
- Phoenix-R?
- Spike-M?

VnC vs. Bass Model



VnC is more accurate than **Bass Model** for **91%** of the submissions

VnC Model: Fit Accuracy (Details)

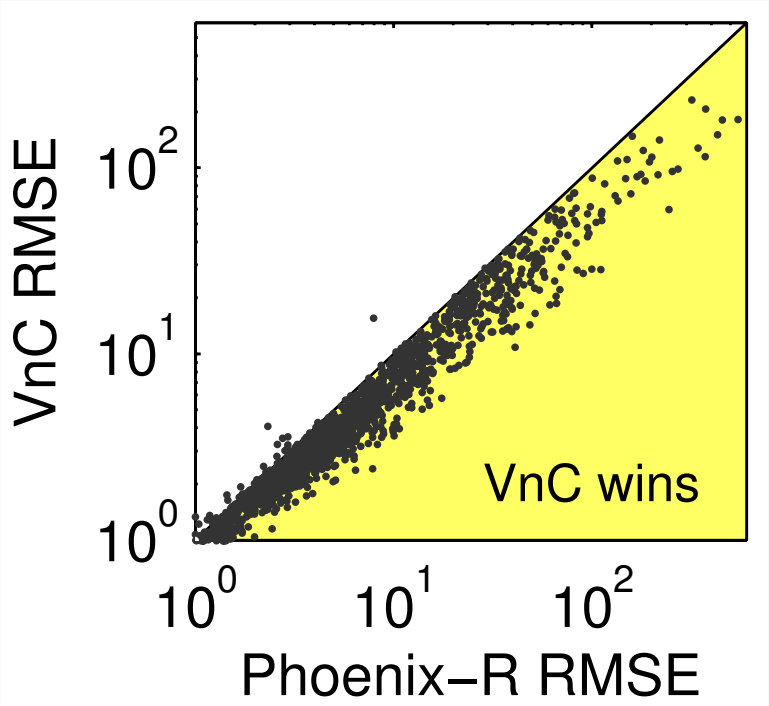
Root-Mean-Square Error

- Lower RMSE indicates a better fit
- Points below the diagonal: time-series that were best fit by VnC

Is VnC more accurate than:

- SI Model? **Yes**
- Bass Model? **Yes**
- Phoenix-R? **Yes**
- Spike-M?

VnC vs. Phoenix-R



VnC is more accurate than **Phoenix-R** for **96%** of the submissions

VnC Model: Fit Accuracy (Details)

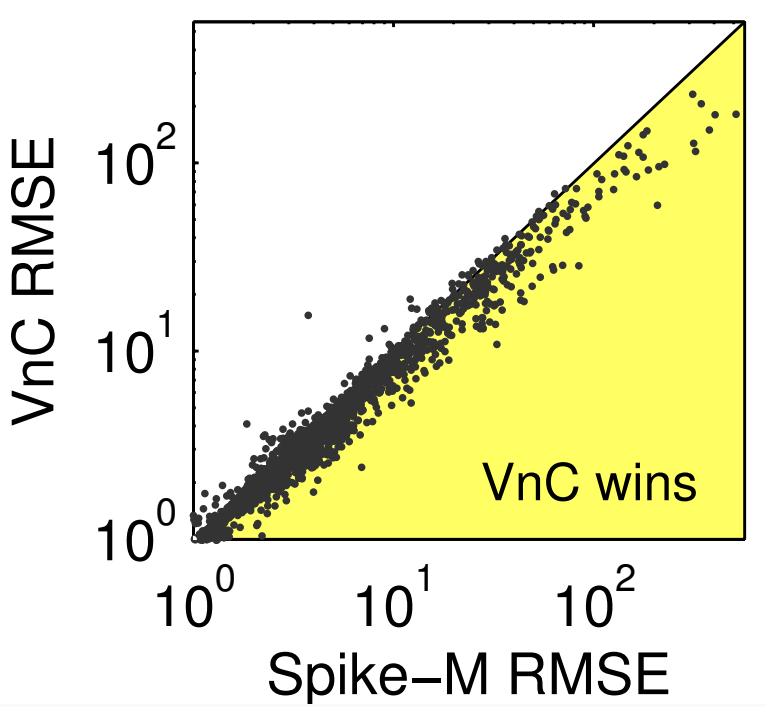
Root-Mean-Square Error

- Lower RMSE indicates a better fit
- Points below the diagonal: time-series that were best fit by VnC

Is VnC more accurate than:

- SI Model? **Yes**
- Bass Model? **Yes**
- Phoenix-R? **Yes**
- Spike-M? **Yes**

VnC vs. Spike-M



VnC is more accurate than **Spike-M** for **90%** of the submissions

MFS-Map: Rule Generation

MFS-Map: Discretization | Rule Generation | Annotation

For each rule, compute its confidence:

- Compute the confidence of all pairs $\text{conf}(f_i, a_j)$
- Not all pairs may occur on the dataset: use a hash-map to save memory

$$\text{conf}(f_i, a_j) = \frac{\#\text{ itemsets containing } f_i \text{ and } a_j}{\#\text{ itemsets containing } f_i}$$

Discard rules with conf $<$ minConf

- minConf is a threshold (user defined parameter)