Protecting Web Servers From Web Robot Traffic

Derek Doran
derek.doran@wright.edu

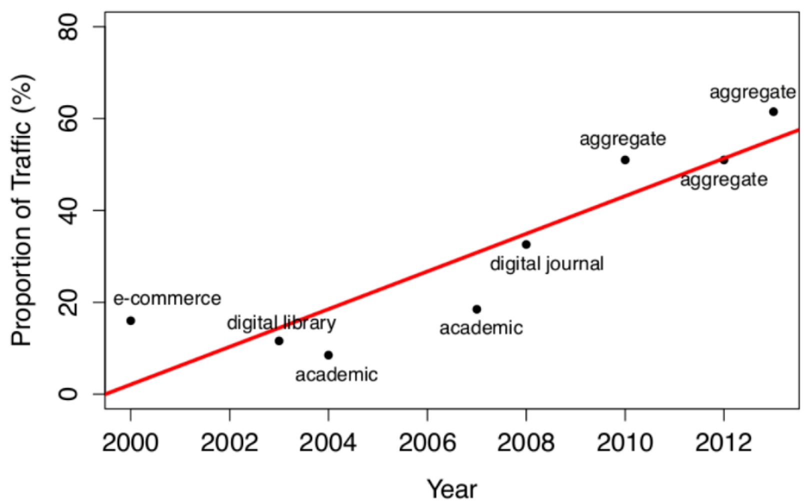
Department of Computer Science & Engineering
Kno.e.sis Research Center
Wright State University, Dayton, OH, USA

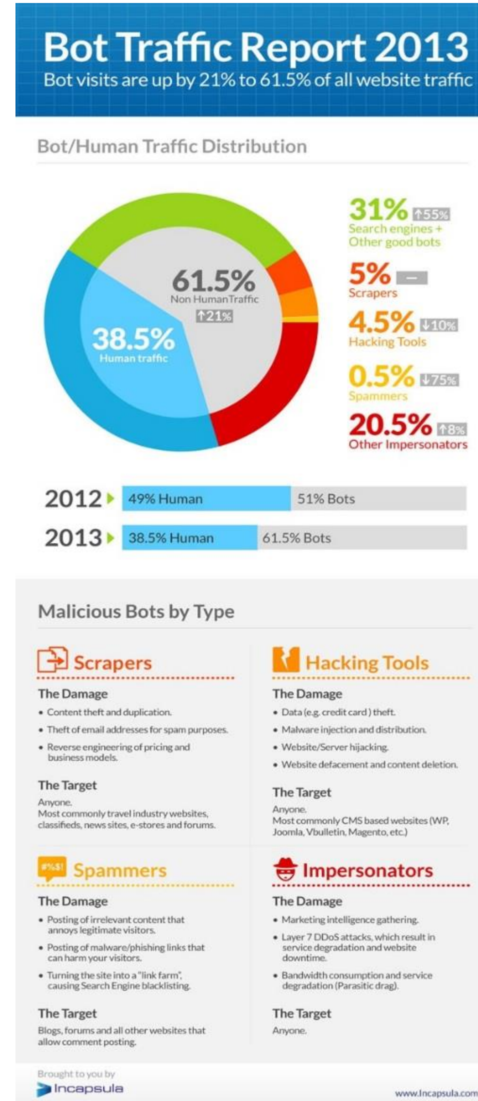
Outline

- Introduction and motivation
- Analysis of Web robot traffic:
  - Robot detection
  - Performance Optimization: Predictive Caching
- Future research

Introduction and motivation

- Web robots are critical to many functions and services:
  - Internet Search
  - E-Business (shopbots)
  - Private, Proprietary Systems
- Latest reports (Dec. 2013): over 60% of Web traffic! http://www.incapsula.com/blog/bot-traffic-report-2013.html





Introduction and motivation

- Within the past 5 years: fundamental shifts in how the Web is used to communicate and share information
  - *Dynamic* vs. static pages
  - Users *produce* vs. consume information
  - *Subscriptions* vs. searching
- Now, data on the Web has never been more valuable
  - 25% of search results for the largest commercial brands are for *user-generated content*
  - 34% of bloggers post opinions about brands
  - 78% of users trust peer recommendations over ads
  - 80% of organizations incorporate social network data in recruitment practices
- Organizations seek to leverage this valuable, dynamic, time-sensitive data, to stay relevant

A New Web Economy…

Data Scraping for NFL & Fantasy Football Stats – .NET MySQL Administration PHP Scraping

View Details

Posted: Sep 4, 2013 Location: United States

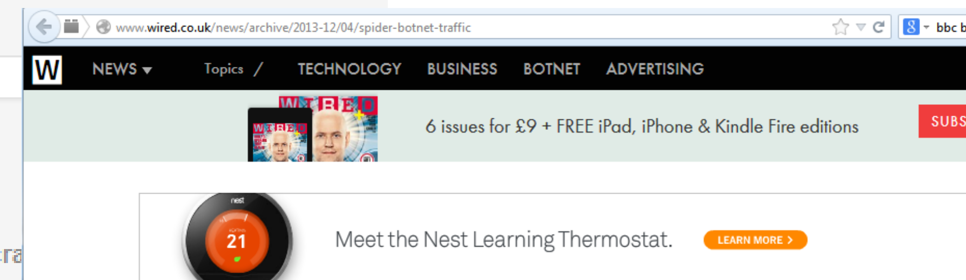
that knows a little about fantasy football. I've attached the list of analyst and the publication. Desi
.NET MySQL Administration PHP **Scraping**

Web scraping, automated database creation

Posted: Sep 4, 2013 Location: United States

ulling/**scraping** information from two sections:

onsible for scra



Some publishers are optimising their sites for bot-generated traffic

TECHNOLOGY / 04 DECEMBER 13 / by OLIVIA SOLON

WordPress Database Web Scraping

Posted: Sep 4, 2013 Location: United States

Wordpress, Database with search functions, web **scraping** capabilities, mobile responsive design, mobile...
Also inquiring if web **scraping** capabilities can be...

Introduction and motivation

- The volume and intensity of robot traffic will further grow over time!
- Web servers optimized only to service *human traffic* with very high performance
  - Workload generation
  - Predictive and proxy caching
  - Optimal queuing, scheduling
- Unprepared to handle robot traffic - current knowledge of Web traffic may not transcend to robots!

- Objective: To perform a comprehensive analysis of Web robot traffic, and to prepare Web servers to handle robot requests with high performance

Outline

- Introduction and motivation
- Analysis of Web robot traffic
  - Robot Detection
  - Preparing Web Servers: Predictive Caching
- Future research

Robot detection

- Deficiency in state-of-the-art: focuses on finding *commonalities* across robot sessions
  - Behavior changes over time, and from robot to robot
- Requirements for more accurate and reliable detection
  - Find *distinctions* between robots and humans rather than commonalities between robots
  - Root detection on a *fundamental* difference between human and robot behavior
    - No matter how robots evolve, this difference remains
  - Analytical, self-updateable model
    - As behaviors change over time, so does the detection algorithm

Robot detection

- Fundamental difference: *Session request pattern:*
  - The order in which resources are requested during a session
- Properties of human session request pattern:
  - Governed by a Web browser
  - Associated with site structure
  - Target specific resources
- Properties of robot session request pattern:
  - No governing interface
  - Requests any resources, at any time
  - May target very specific resources depending on functionality

Session request pattern

- Request patterns must be generic enough to characterize many different sessions in a similar way
- Partition resources into various classes

Class	Extensions
text	txt,xml,sty,tex,cpp,java
web	html,asp,jsp,php,cgi,js
img	png,tiff,jpg,ico,raw
doc	xls,doc,ppt,pdf,ps,dvi
av	avi,mp3,wmv,mpg
prog	exe,dll,dat,msi,jar
compressed	zip,rar,gzip,tar,gz,7z
malformed	Req. strings not well-formed
no extension	Request for dir. Contents

Detection Algorithm

- Encode session request patterns of robots and humans into two different discrete time Markov Chains (DTMCs) R = ($s_{r,}$ P_r) and H = R = ($s_{r,}$ P_r)
  - Parameters estimated from logs

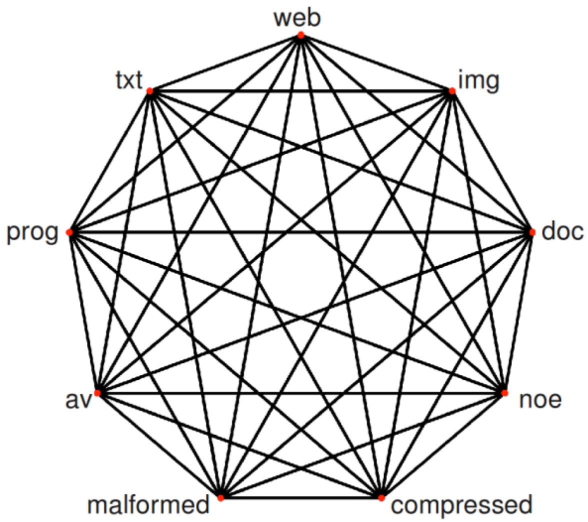
- Detection algorithm
  - For an unlabeled session

$$x = (x^1, x^2, \dots, x^n)$$

Compute probability R or H generates x:

$$\log(Pr(x|s_r, P_r)) = \log(x_r^1) + \sum_{i=2:n} \log[P_r]_{x^{i-1}, x^i}$$

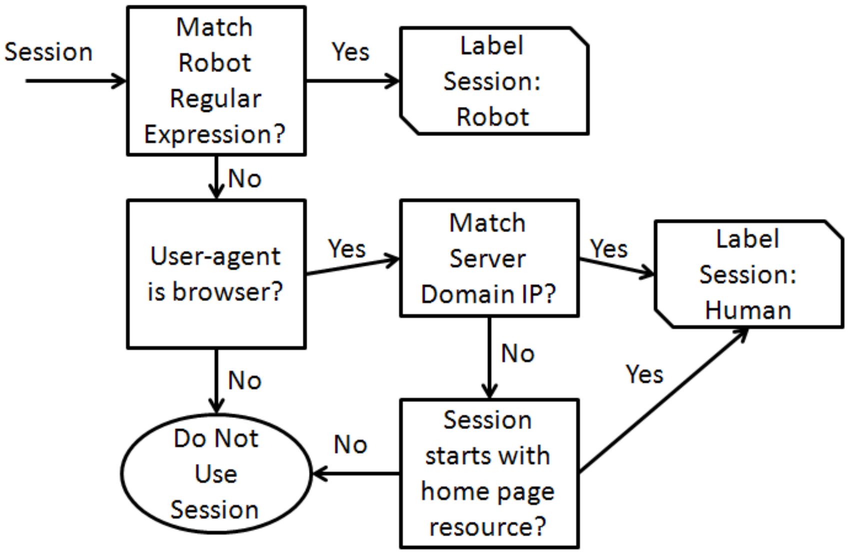
Label x as a robot if $\Pr(x|s_r, P_r) > \Pr(x\ |s_h, P_h)$



Datasets

- We consider data from one-year access logs over a variety of servers:
  - Academic: University school of Engineering
  - E-commerce: Univ. of Connecticut University bookstore
  - Digital Archive: Online database of United States Public Opinion Information

- Millions of access logs across each Web server
- Using a heuristic approach, divided the logs into robot and human requests

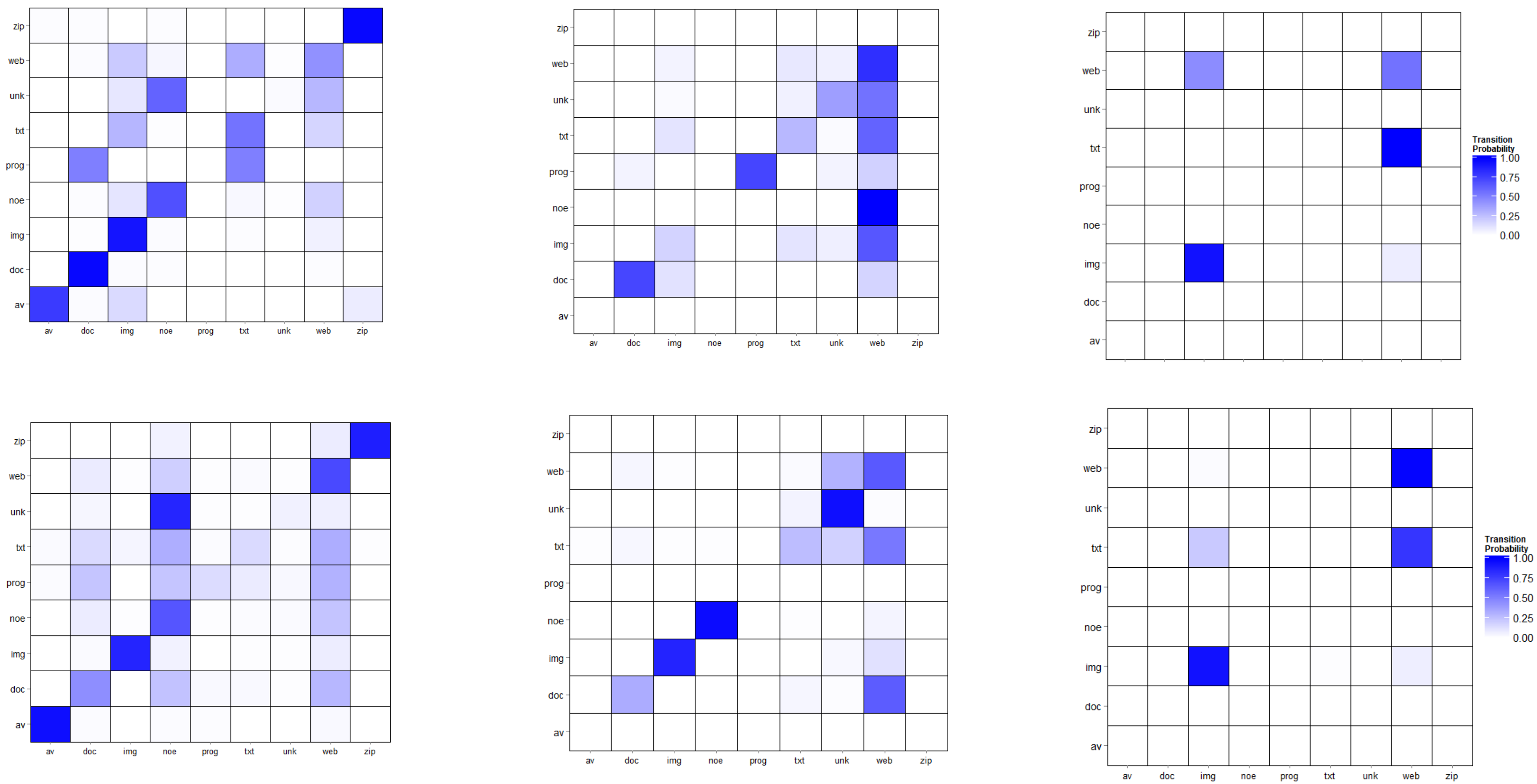


Test Set	Date	Robots	Human
Academic	Mar 2011	4322	6121
Digital Archive	Dec 2009	3752	1178
E-commerce	Aug 2008	1419	556

DTMC Comparison (Behavior Fingerprints)

R

H



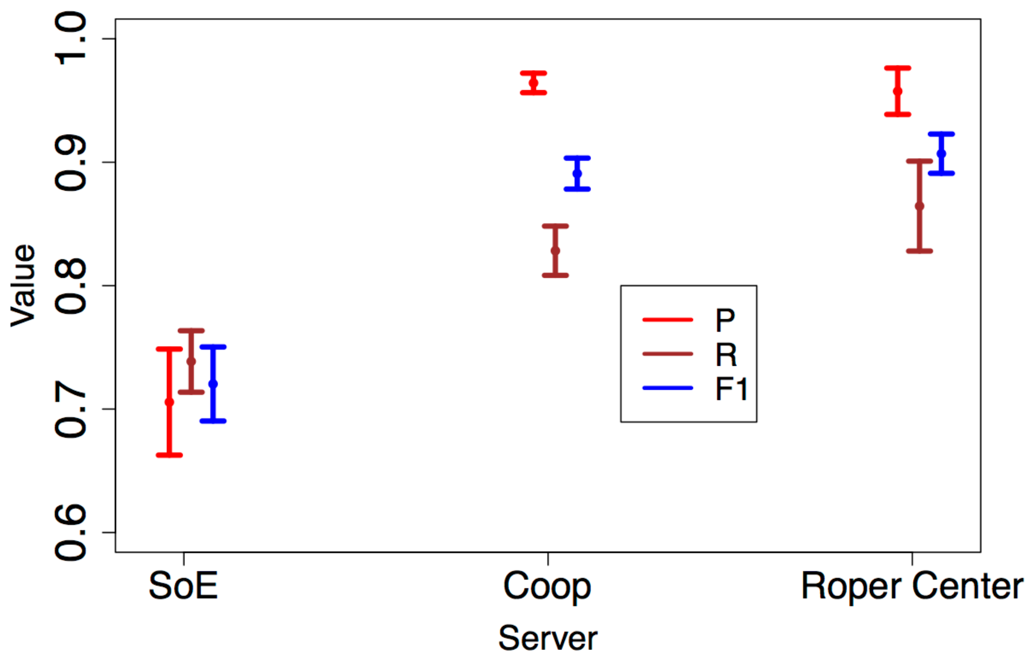
Academic

Digital Archive

E-Commerce

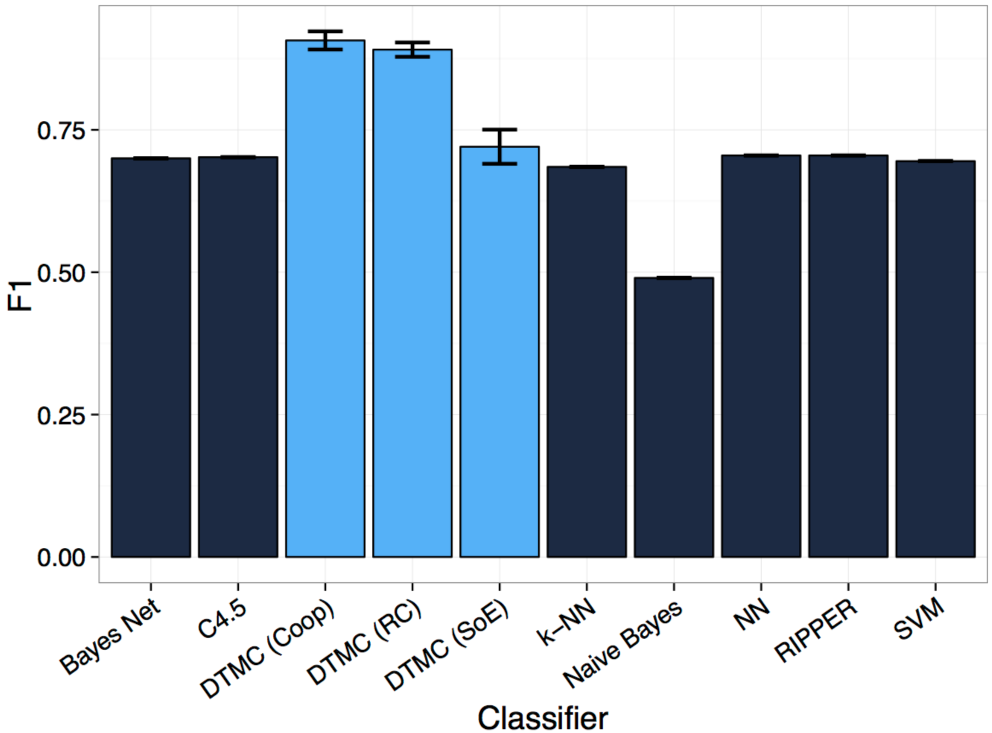
Offline detection

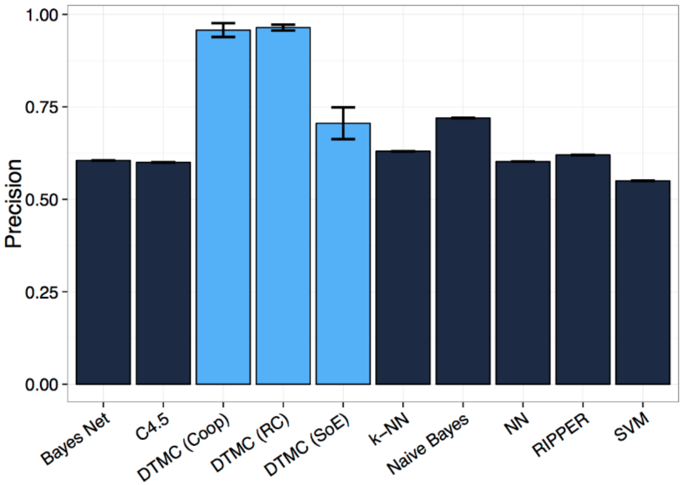
- Performance evaluated using precision, recall and F1
  - Precision: true pos. count / true pos. + false pos. count
  - Recall: true pos. count / true pos. + false neg. count
  - F1: harmonic mean of precision, recall

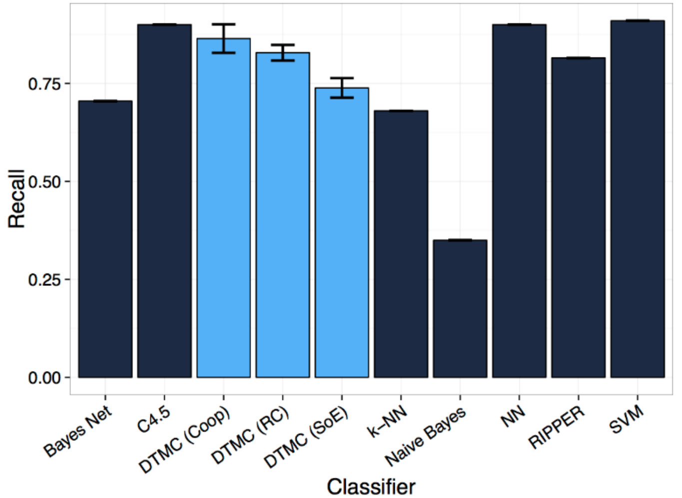


Comparative Analysis

- Versus state-of-the-art results using various supervised learners







Real-time detection

KNO.E.SIS

- Offline detection is an `after-the-fact' analysis
  - Great for log processing; statistical analysis
  - "Damage survey"

- Real-time detection catches robots in the act
  - Differentiable treatment of robots and humans
  - Control and handle crawling activities
  - "Damage *control*"

- State-of-the-art methods offer an *engineered* solution
  - Painful for the users (CAPTCHA)
  - Complex server-side systems target specific classes of robot traffic
  - Difficult to implement and maintain in practice

Real-time detection

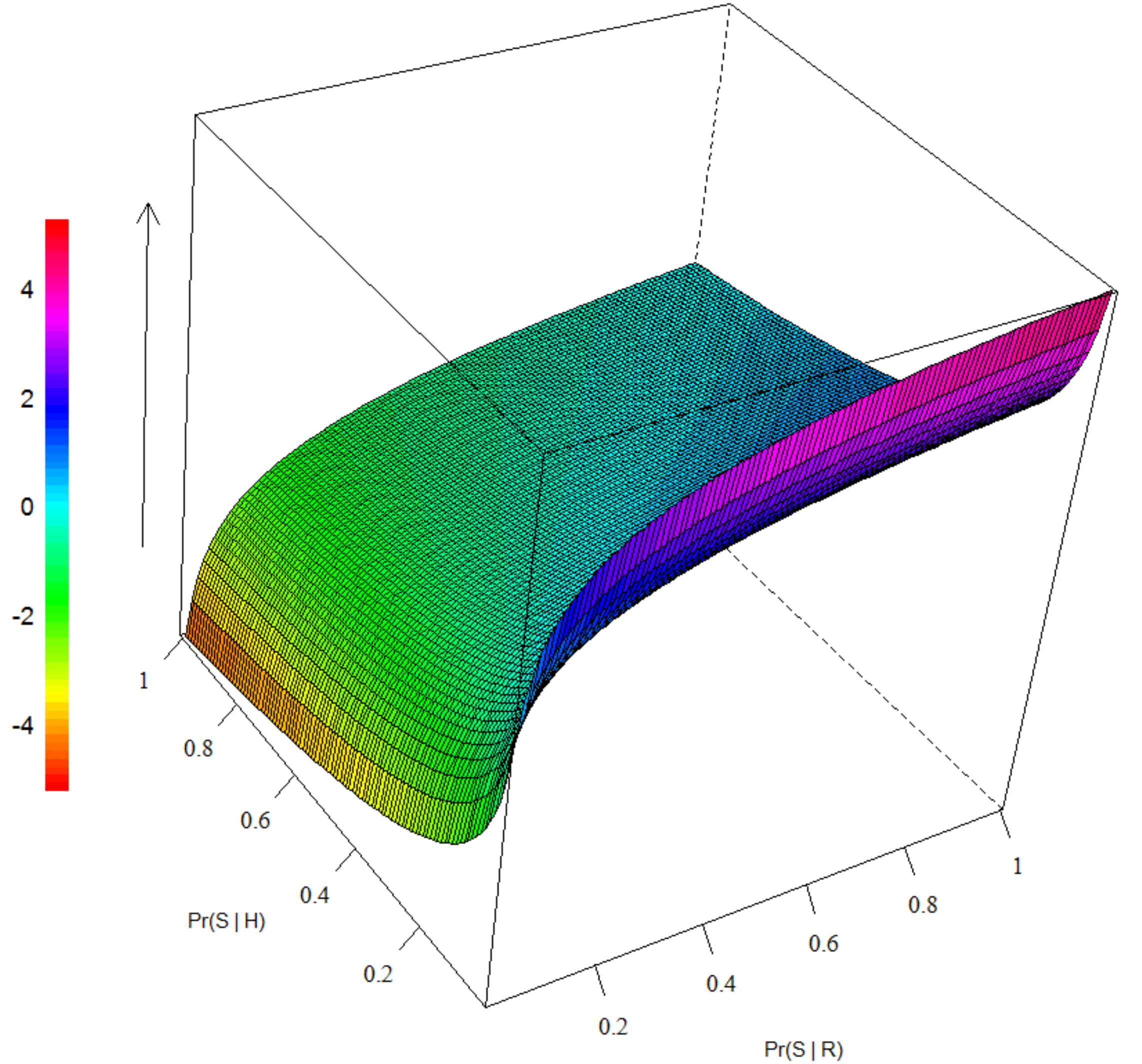
- We can adopt our offline algorithm to run in real-time:
  1. For every active session s, maintain Pr(s | R); Pr(s | H)
  2. On new request, update Pr(s|R), Pr(s|H).
  3. If number of requests is > *k* and the difference in log-probabilities exceeds a threshold Δ, classify.

Parameter functions:

- *k* – give Pr(s|R), Pr(s|H) chance to stabilize
- Δ – tune tradeoff between reliability and need to classify
  - Low Δ: We classify more sessions, but may be less accurate
  - High Δ: Very confident classifications, but sessions may go unlabeled
- Choice of Δ depends on the Web server

Choices of Δ

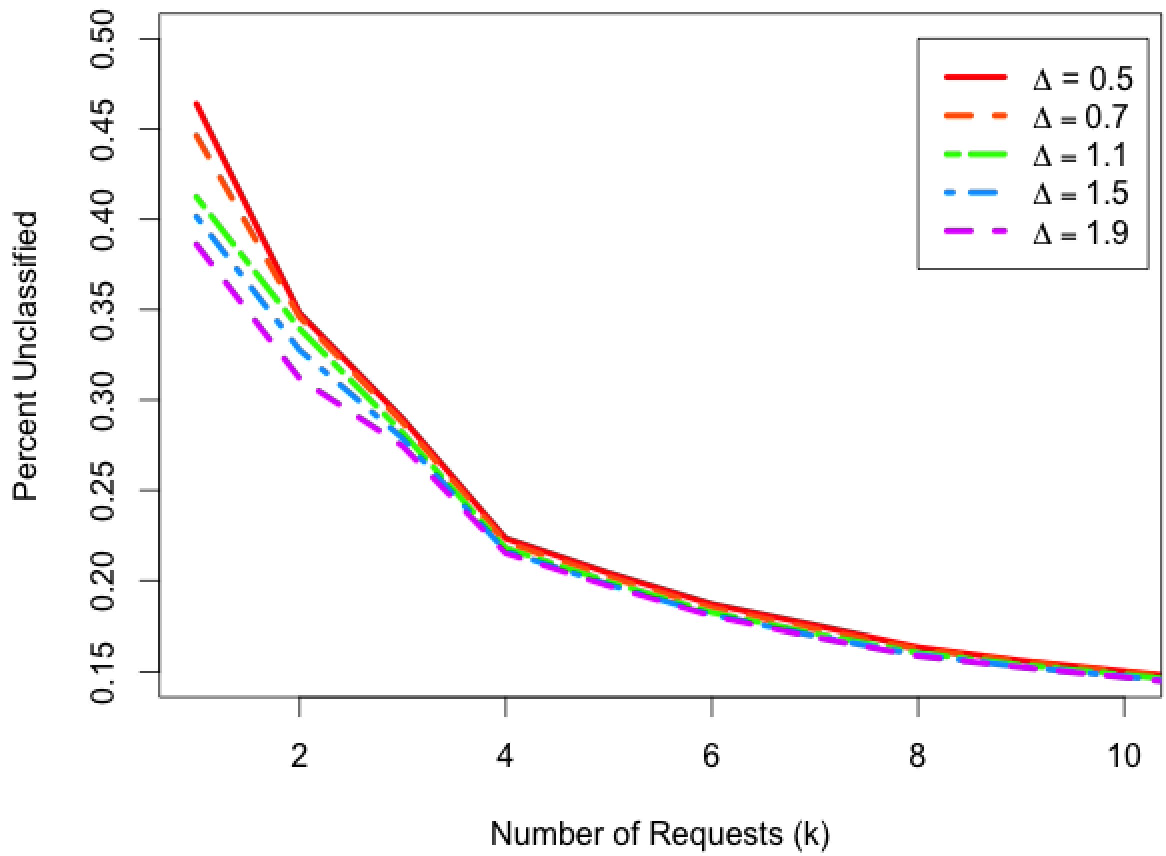
0.5 < Δ < 2 offers broad degrees of confidence



Effect of k, Δ on sessions missed

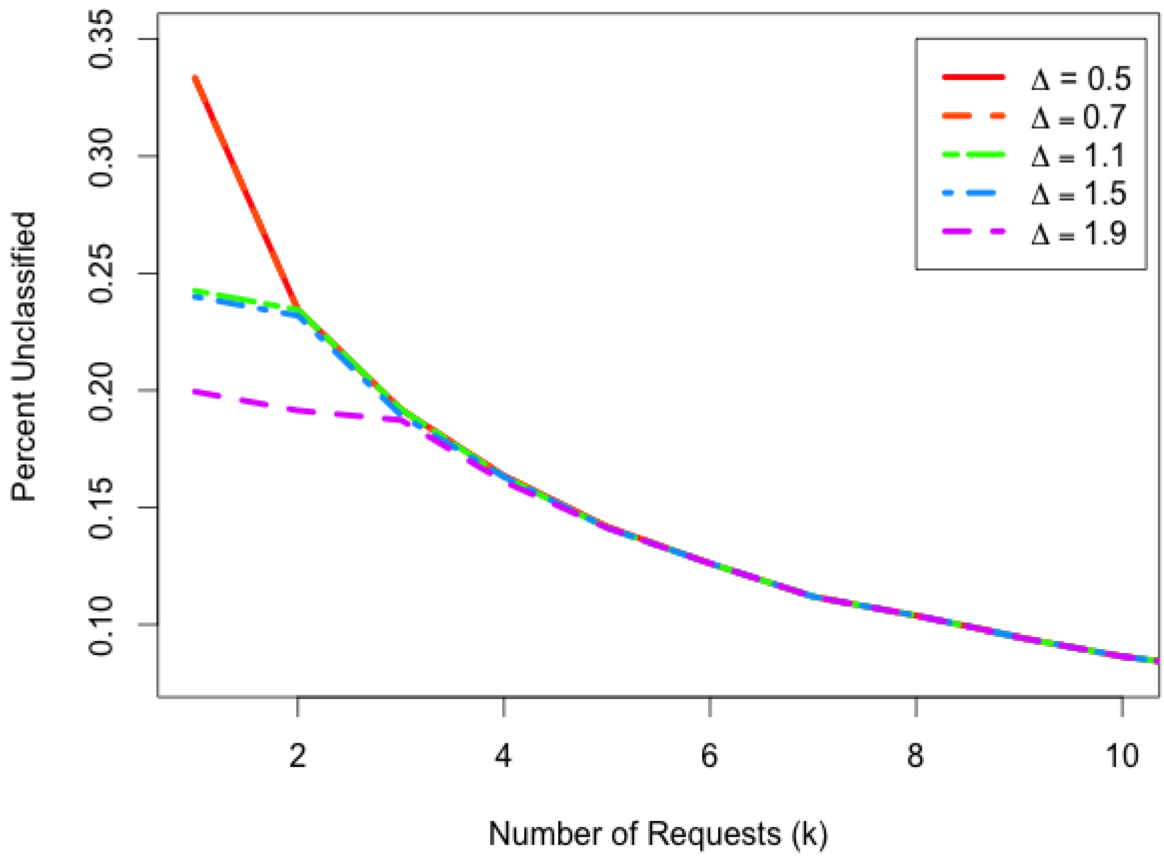
Academic
Δ = 1.5; k > 6:
~ 20% of sessions
go unclassified

Note: Δ = 1.5 is very broad
Ex: if Pr(s|R) = 0.7, we require Pr(s|H) < 0.173 before the log-probability difference exceeds Δ



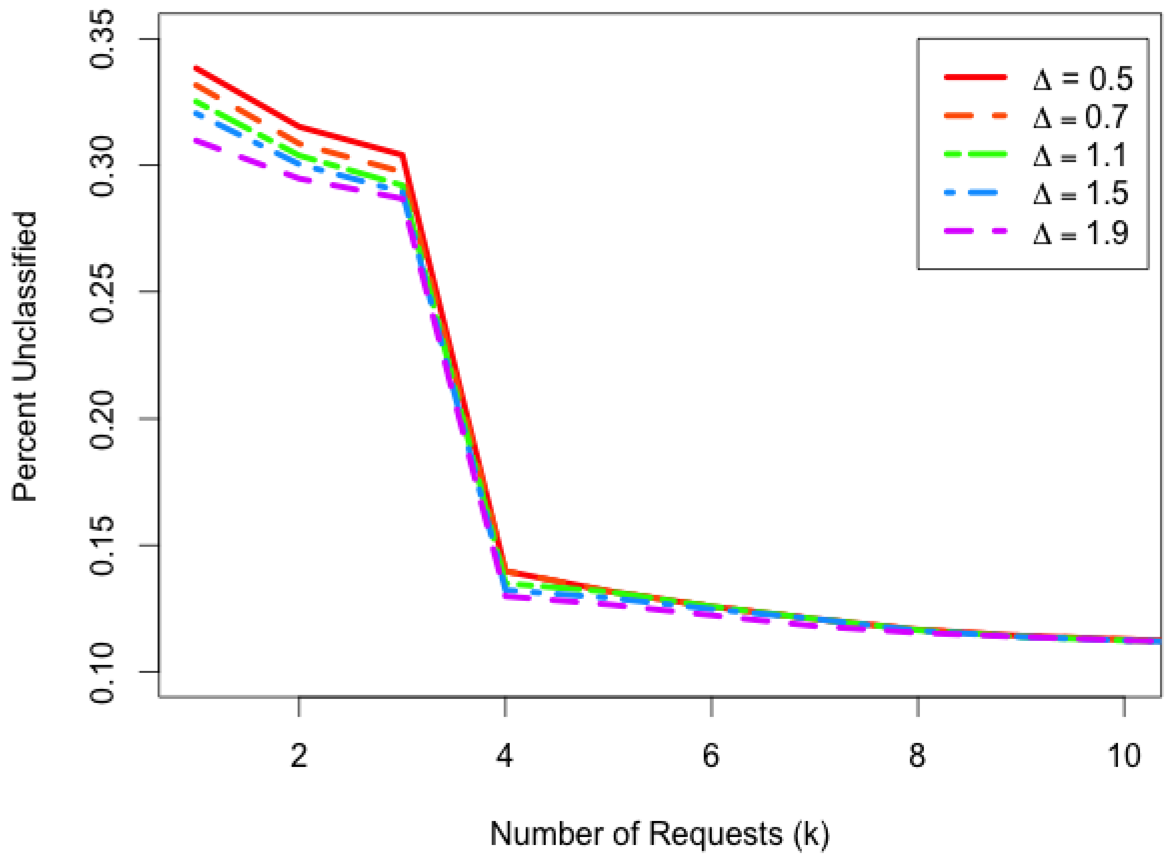
Effect of k, Δ on sessions missed

E-Commerce
Δ = 1.1; k > 6:
~ 12% of sessions go unclassified



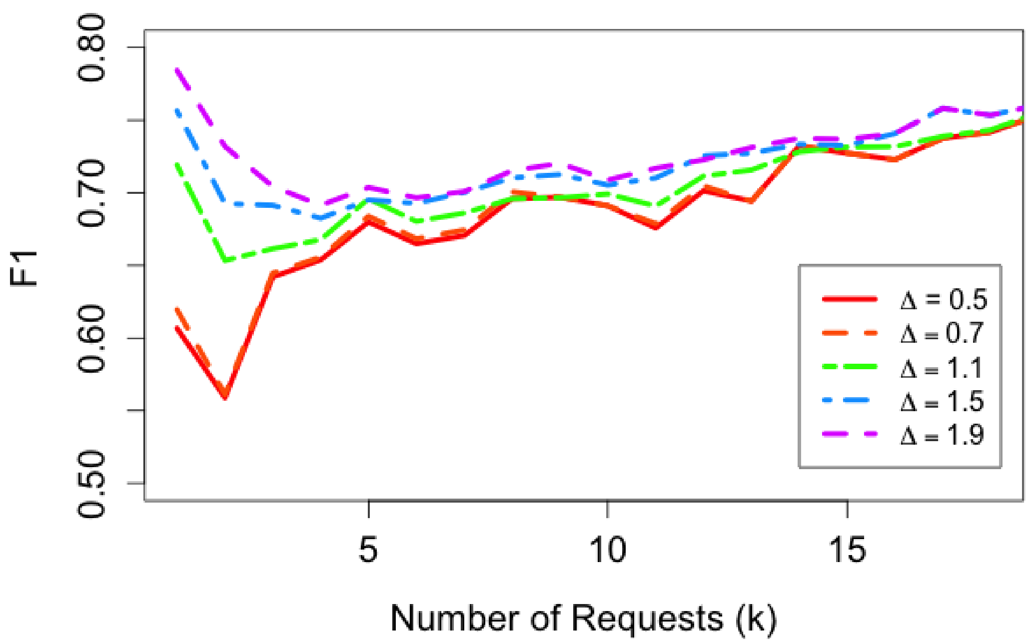
Effect of k, Δ on sessions missed

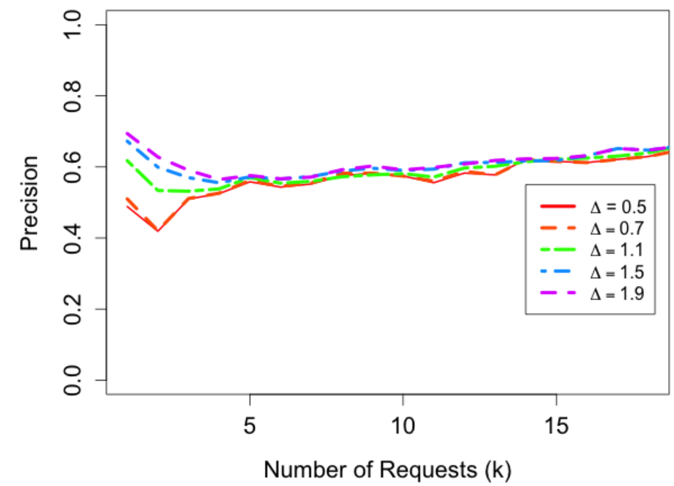
Digital Library
Δ = 1.1; **k > 4**:
~ 12% of sessions
go unclassified

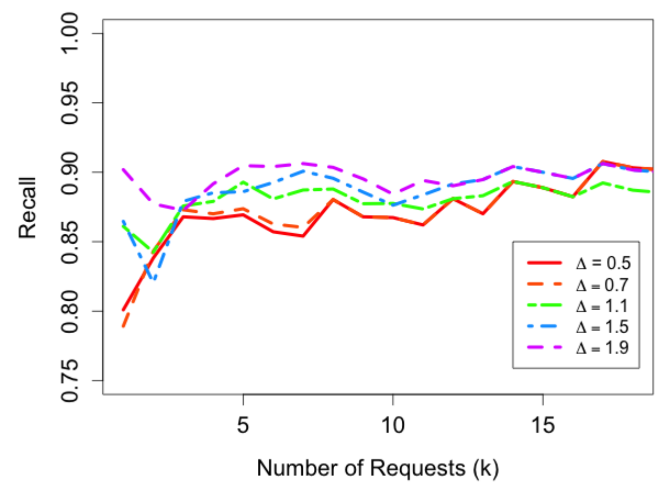


Real-time detection performance

- Academic Server
  - Good results (F1 > 0.7 at k > 10)
  - False positive rate pulls down F1
  - FP rate improves with larger requests processed



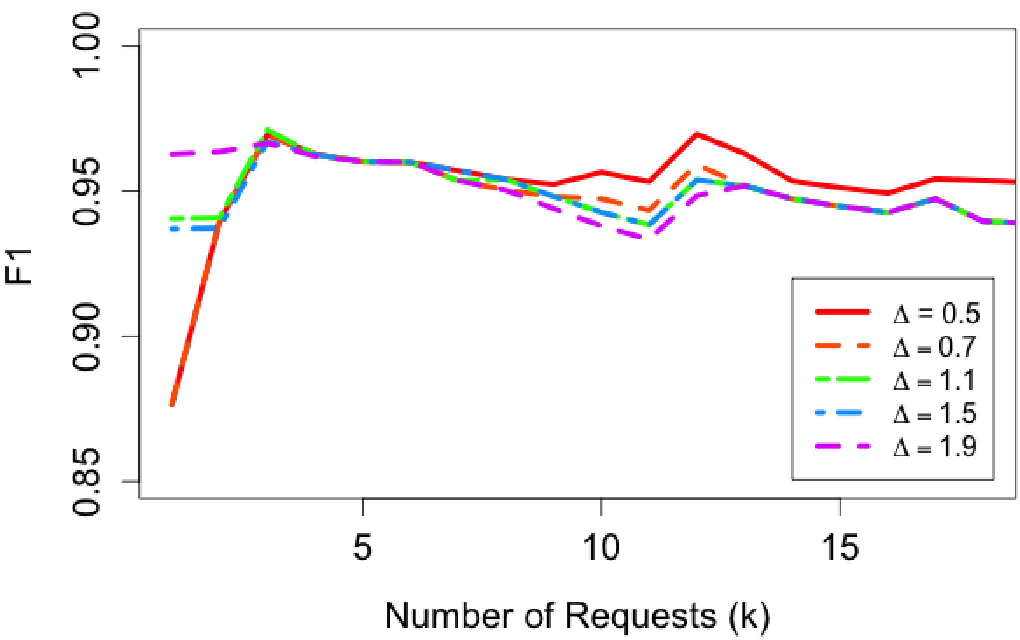


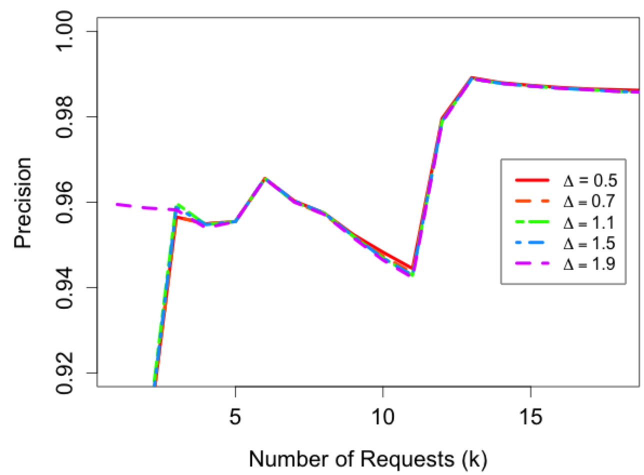


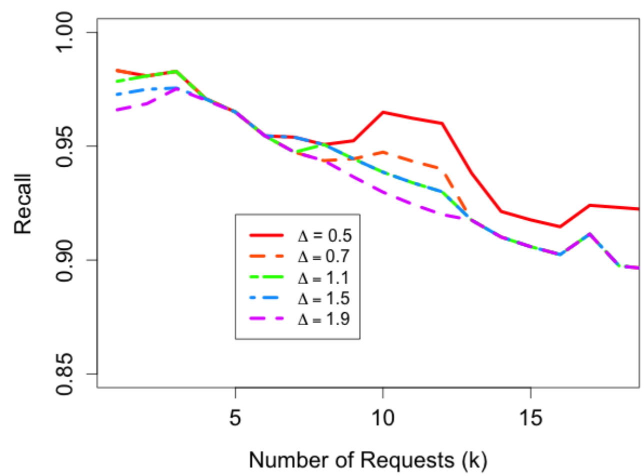
Real-time detection performance

KNO.E.SIS

- E-commerce Server
  - Very strong results (F1 ~ 0.95 for k > 5)
  - Decreasing accuracy for larger k
    - For many requests, robots start to look like humans
  - Balanced by very low FP rate

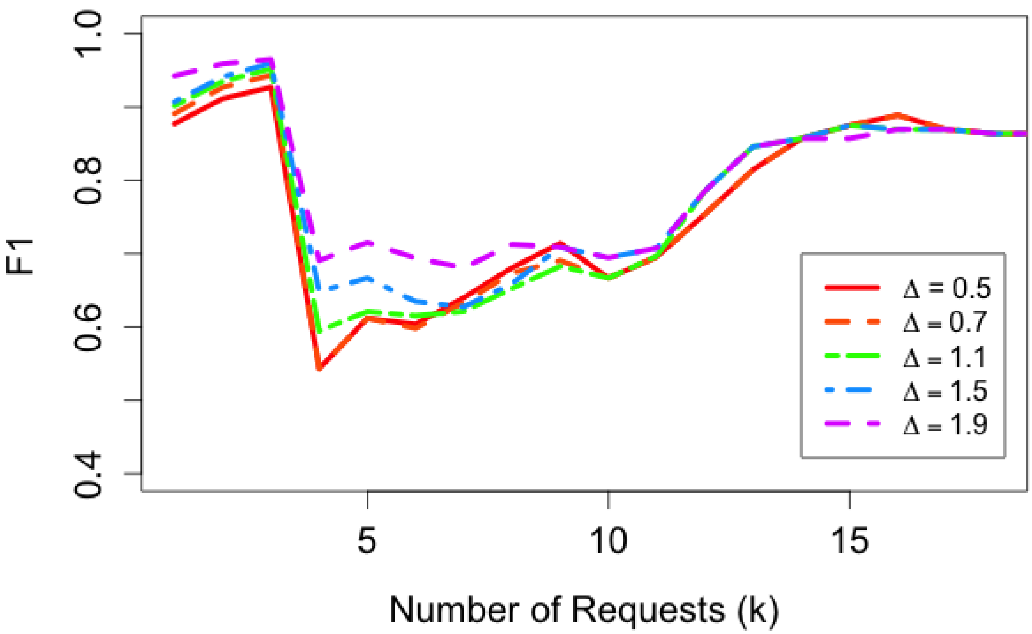


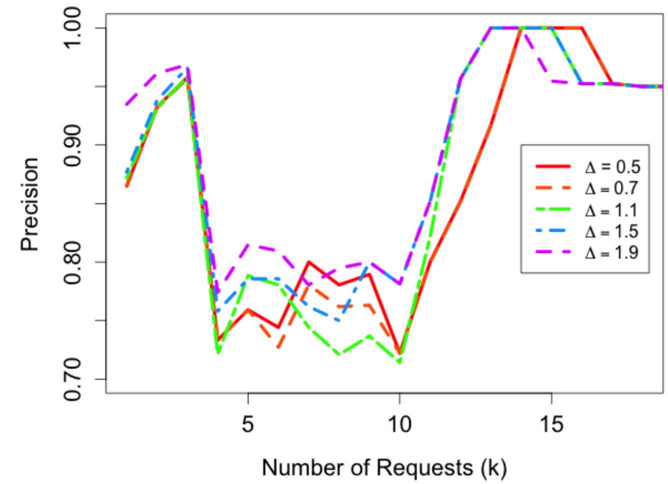


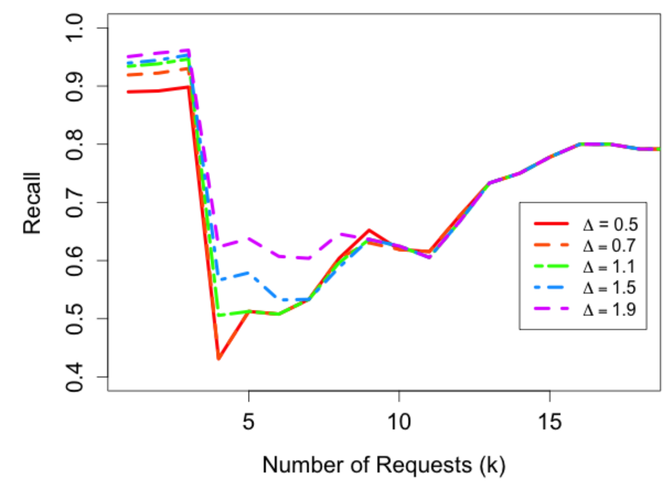


Real-time detection performance

- Digital Archive Server
  - Great results (F1 > 0.8 for k > 12)
  - Drop in FP rate for k > 12
  - Accuracy enhanced at k > 12
    - May be due to Web site structure: static home, log in pages







Robot detection

- Summary
  - Offline detection
    - Across a variety of distinct datasets, strong performance (Approx. F1 > 0.9; ~ 0.73 for Academic Web server)
    - Improvement over state-of-the-art
  - Real-time detection
    - Very strong real-time capability, depending on domain (F1 > 0.75; ~ 0.95 for E-commerce)
    - Decision can be made within a small number of requests (k > 12)
    - Despite strict settings of Δ, low percentage of sessions go unclassified
  - Variation in results across web server domains!
    - Interactions between site structure or content? Can this be incorporated in a resource request pattern model?

Outline

- Introduction and motivation
- Analysis of Web robot traffic:
  - Robot detection
  - Performance Optimization: Predictive Caching
- Future research

Web Caching

- Web server / cluster *caching* is a primary means to provide low latency, reduce network bottlenecks
- Caches store some resources in a smaller, faster, more expensive level of memory (RAM or controller vs. HDD)

- Very limited size, but very fast access
  - Cache hit:
    - Low-latency response
  - Cache miss:
    - High-latency response due to disk I/O; increases cluster bandwidth; ages Web server
- Caching *polices* dictate how and when resources are loaded into a cache

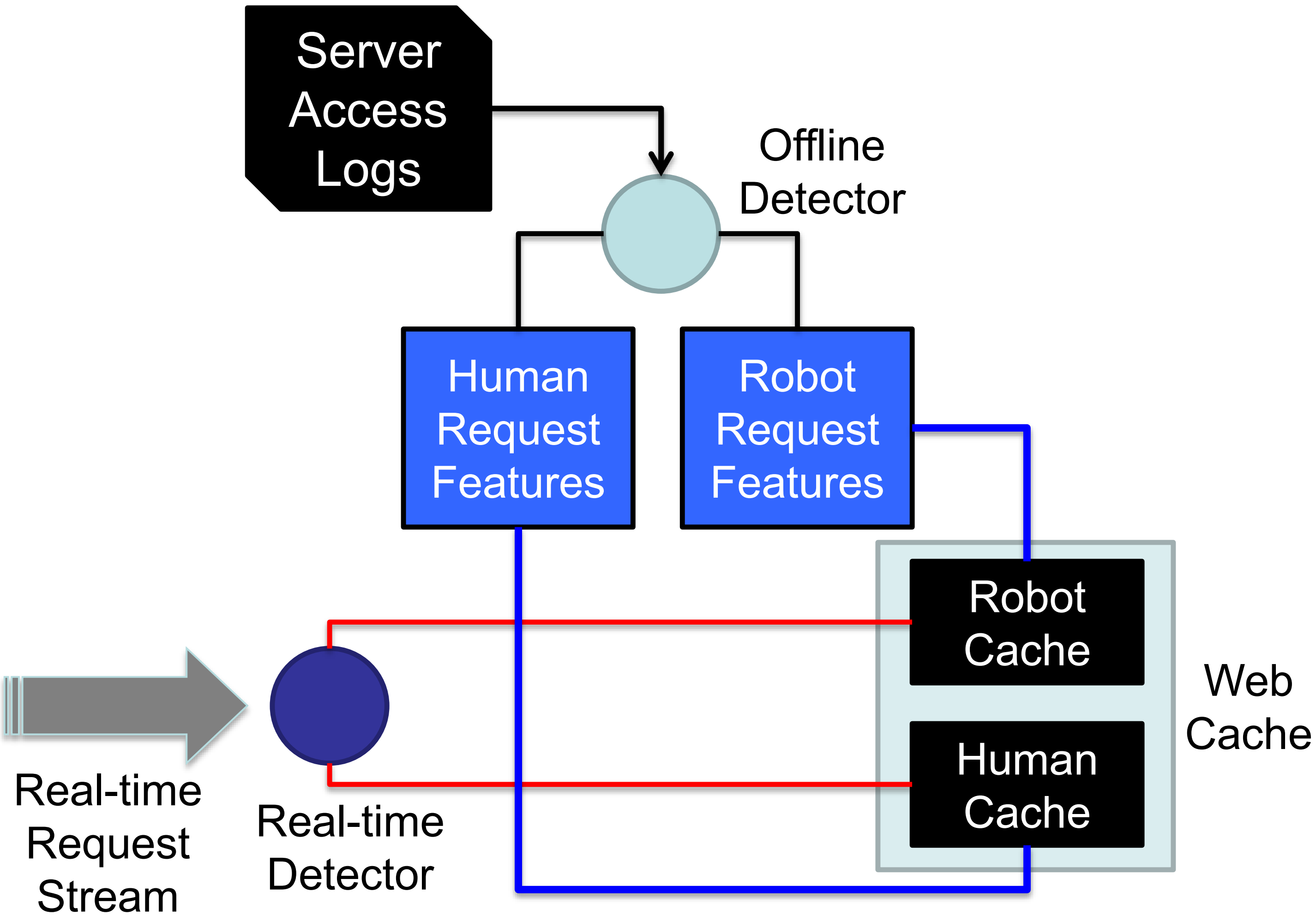
Web caching polices

- Numerous polices exist, built around simple heuristics:
  - Least-recently-used (LRU): keep resources recently accessed in the cache [repeated requests]
  - Log-size: Store as many resources as we can
  - Popularity: Keep frequently requested resources

- Can we service robot requests with such rules? Robots…
  - Do not send repeated requests for same resource
  - May specifically target resources of a given size
  - Could favor different resources compared to humans

- Different behaviors → Handle with separate caches
  - Leverage our offline and real-time detector

Proposed Caching Architecture



Predictive robot caching policy

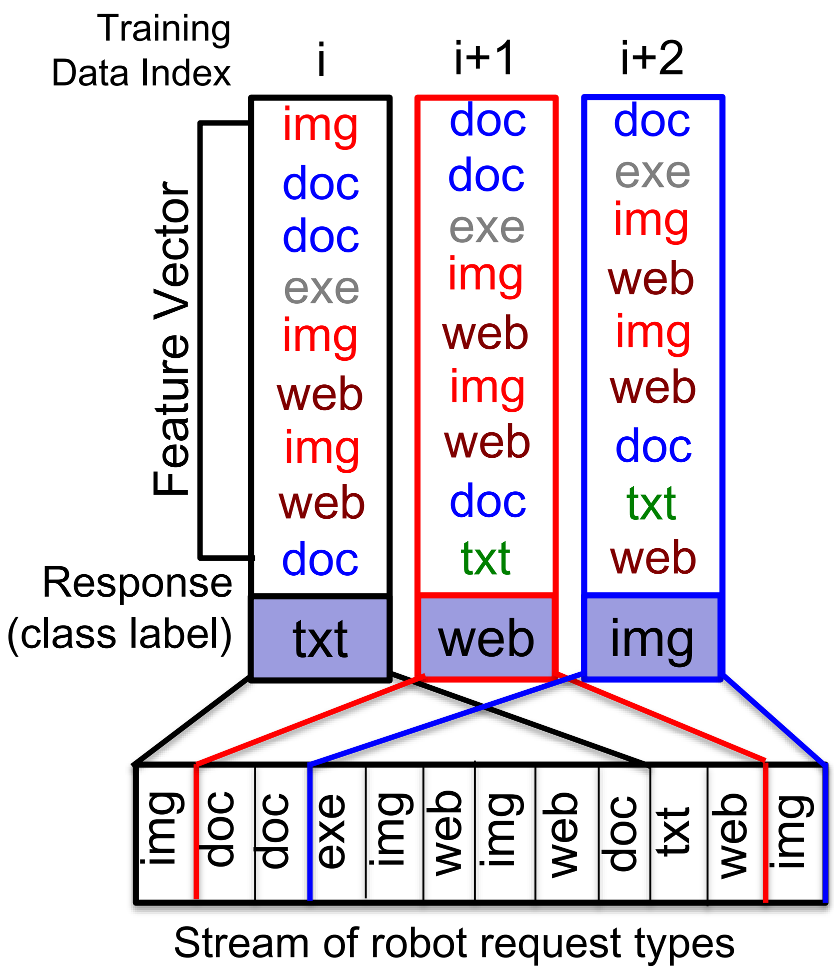
- Intuition:
  - Detection demonstrated that the *type* of the next robot request is predictable
  - Resource-based classification finds robots to favor a small number of resource types, captured in request *sequences*
  - Characterizing robot resource popularity: power-law distribution

- Idea:
  - Extract sequences of request *types* from robot sessions
  - Predict *type* of the next resource
  - Select resources to admit into cache based on frequency of requests within predicted type

Learning request sequences

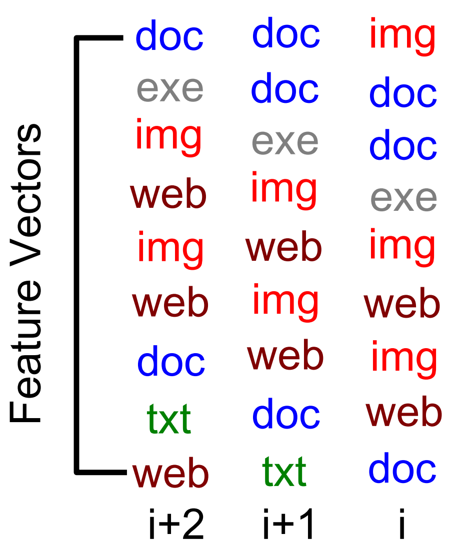
- Request sequence: types of last *n* consecutive requests made in a robot session

- Prediction task: given the order and types of last n-1 requests, predict type of *n*th request



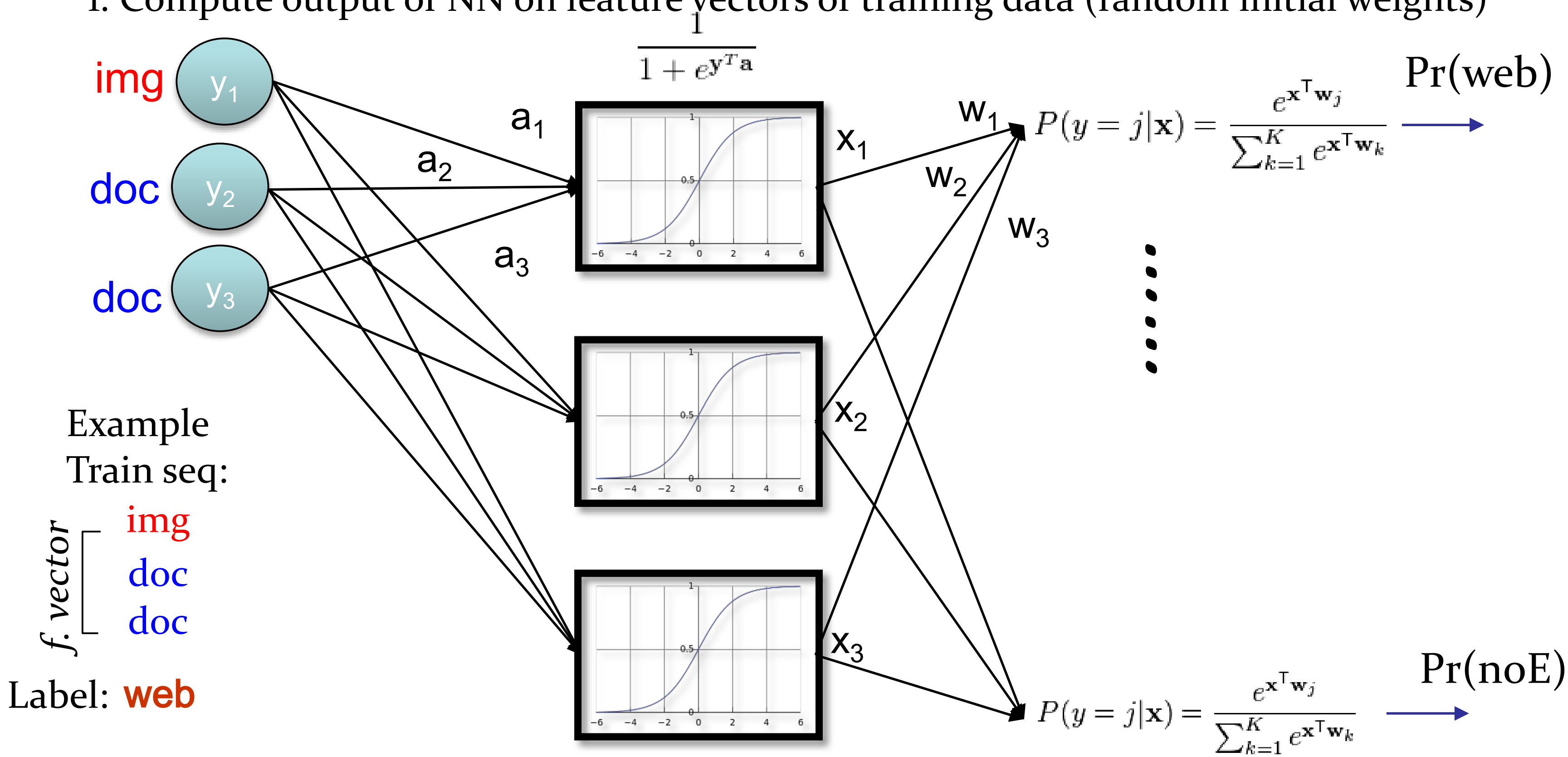
Choosing a classifier

- NN, SVN, Mult. Log-regression:
  - Only learns *features* of a request sequence (*i* has 3 doc, 2 web, 3 img, 1 exe; 2 img-web subsequence)
  - Does not correlate features across training data
- Nth-order Markov based models:
  - Learns *ordering* of sequences (*i has img* in pos. 1, *i+1* has *doc* in pos. 1)
  - High-order needed to capture rich features
- *Elman Neural Network* learns using both features and ordering
  - Learns sequence features like a NN
  - Uses layer of *context* nodes that integrates previously seen sequences throughout training process



Neural network training

1. Compute output of NN on feature vectors of training data (random initial weights)



Output = [Pr(web), Pr(txt), Pr(img), Pr(doc), Pr(av), Pr(prog), Pr(com), Pr(mal), Pr(noe)]

Truth = [1, 0, 0, 0, 0, 0, 0, 0, 0]

Repeat for all training samples

Neural network training

- Define an error function that measures difference from Truth to Output

$$J(w) = -\sum_{i=1}^{n} \sum_{k=1}^{c} t_{ik} \ln(z_{ik})$$

t_{ik}: target output of training sample *i* at index *k*

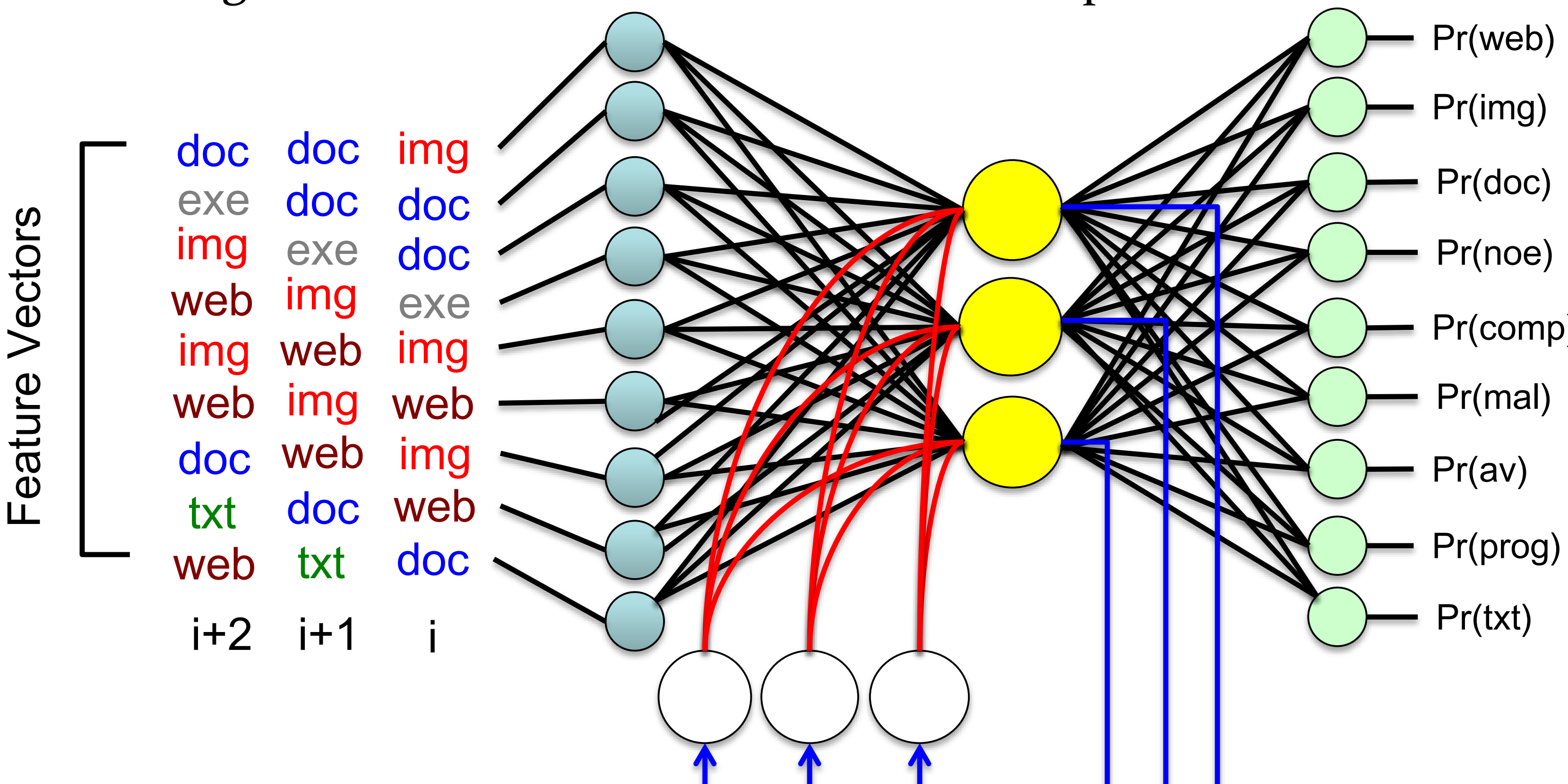
z_{ik}: predicted output of training sample *i* at index *k*

w: network weights learned through training

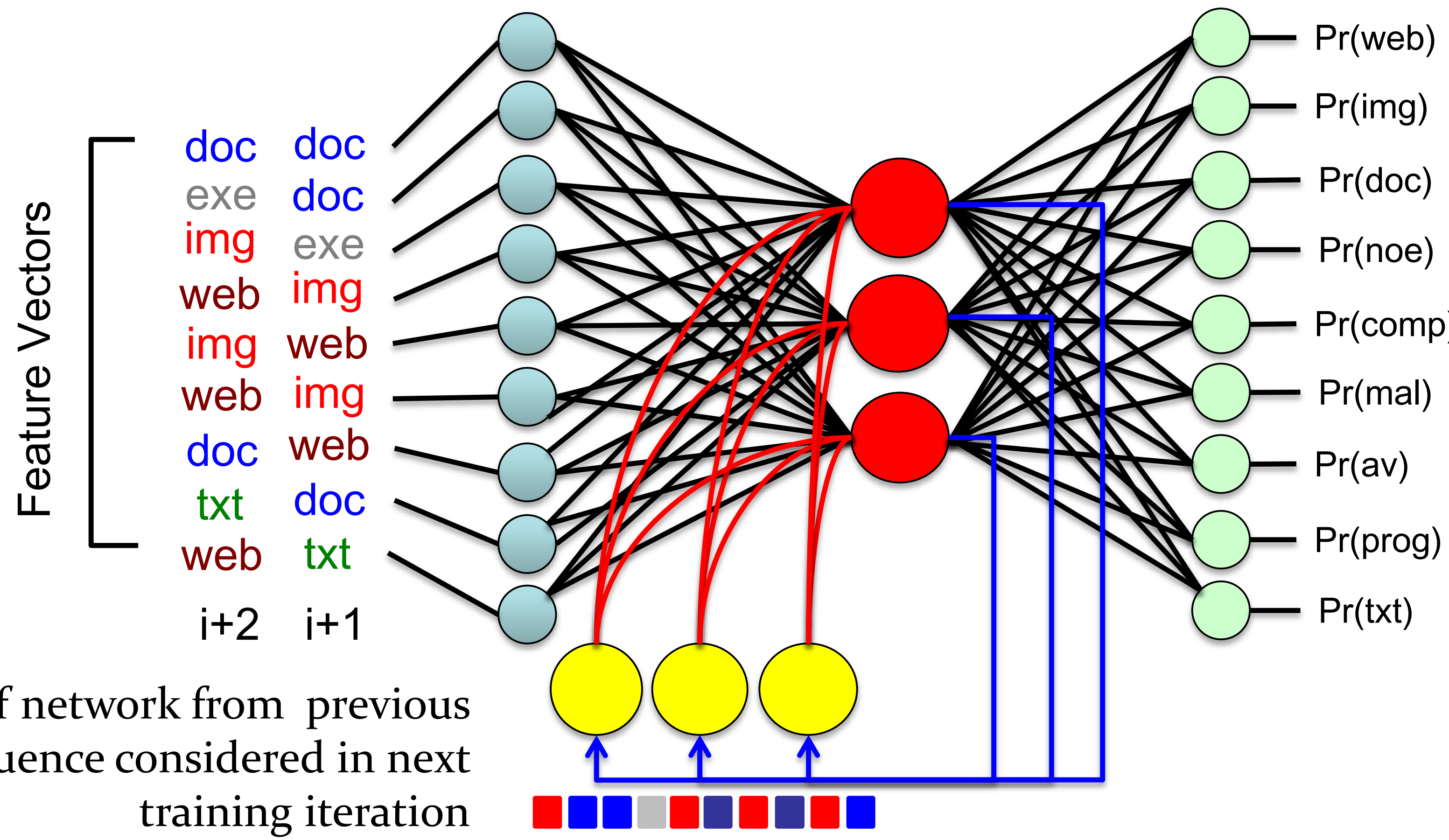
- Minimize J w.r.t. each weight w by simultaneously minimizing all partial derivatives $\partial J/\partial$w
  - Use stochastic gradient descent to approximate computationally
- Run network with new weights w, compute new J, re-optimize w…
  - Repeat until convergence: $|J(w_{i-1}) - J(w_i)| < \delta$

Elman neural network training

- Elman NN Twist: hidden units save state to context units
- Weight from hidden to context = 1
- Weights from context to hidden: additional parameters

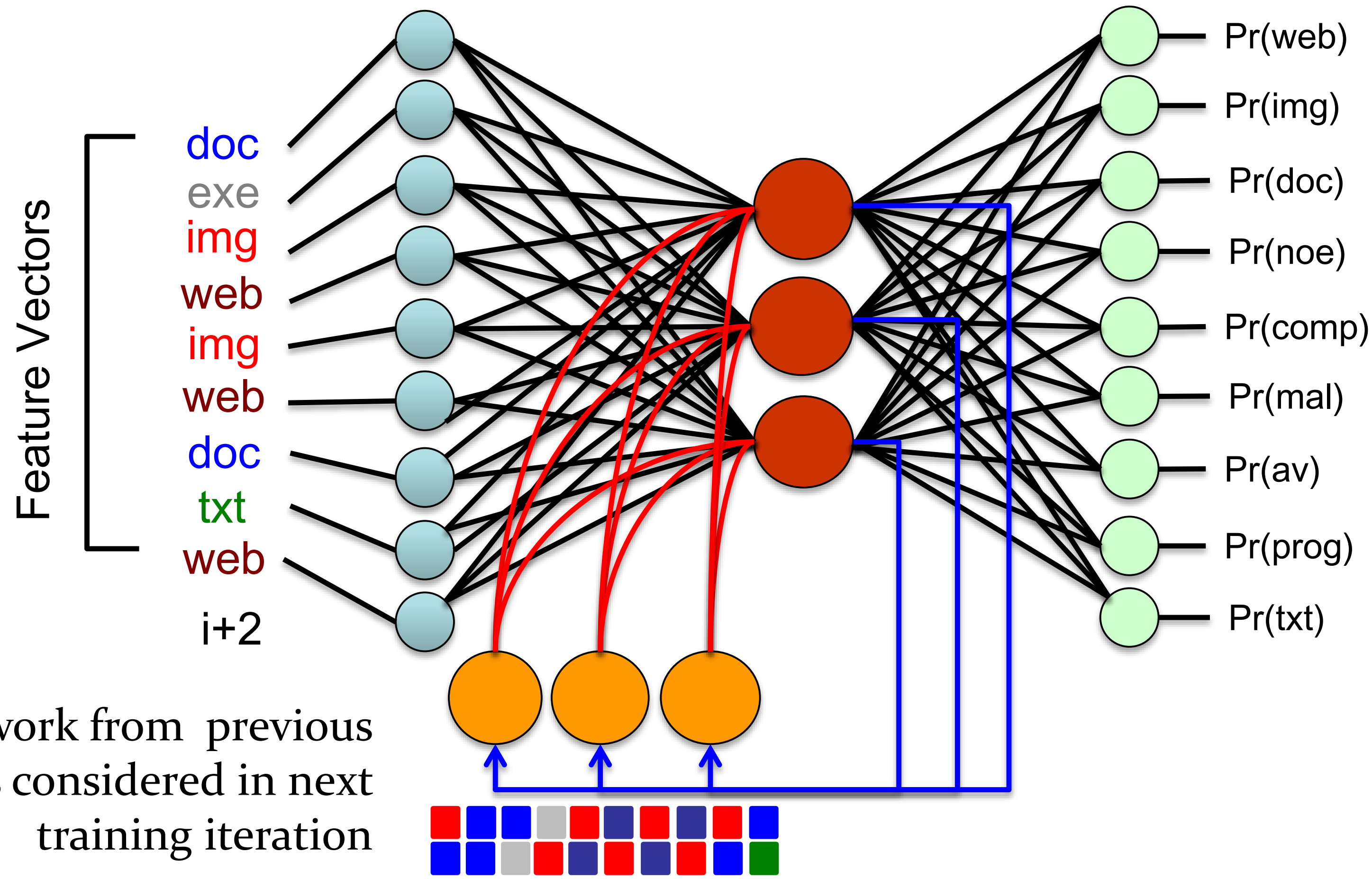


Elman neural network training



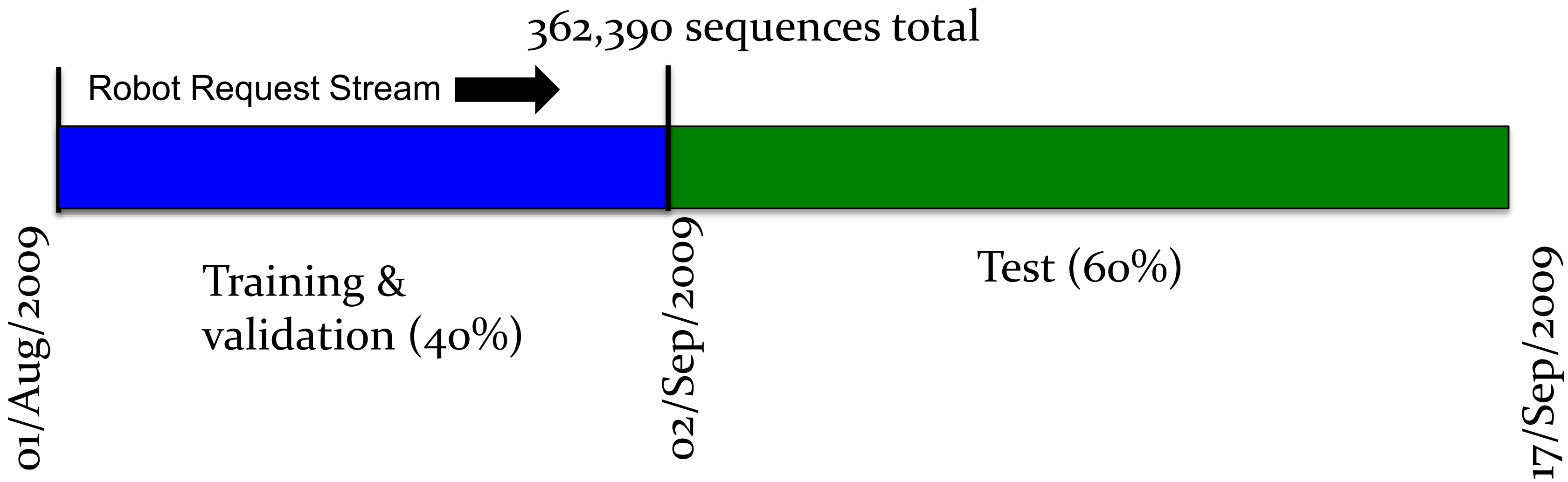
State of network from previous sequence considered in next training iteration

Elman neural network training



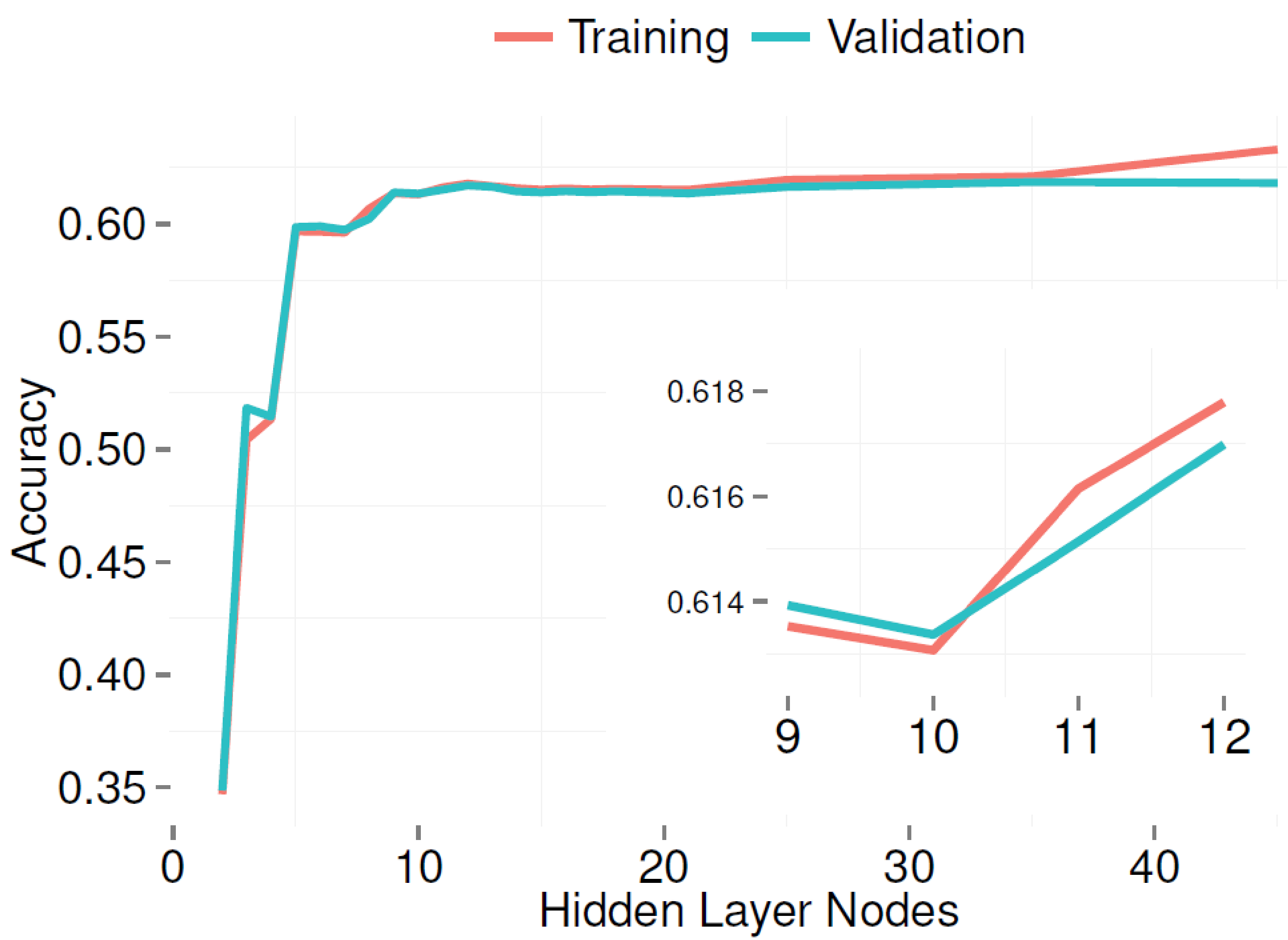
State of network from previous sequences considered in next training iteration

Network training and validation



- Sequences of size k=10
- First 40% of requests used to find best # of hidden units for ENN
  - 10-fold cross-validation
- Evaluate ENN prediction accuracy on rest of data; compare results against many other multinomial predictors

Fitting neural network size



Comparison of classifiers

- We compare the classification accuracy of ENN against other typical multinomial classifiers
  - DTMC (learning only by sequence order):
  - Multinomial Logistic Regression (learning only features):
  - Random guess (Correct 1/9 times)

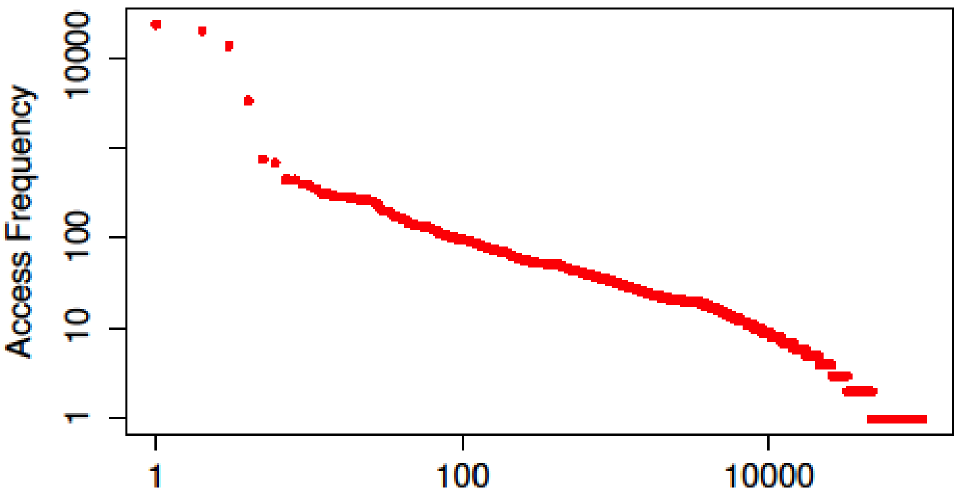
Model	Accuracy	Gain-RG	Gain-MLR	Gain-DTMC
RG	0.111	-	-	-
MLR	0.338	67.16%	-	-
DTMC	0.392	71.68%	16.0%	-
ENN	0.647	82.84%	47.8%	39.4%

- Order in request sequences may be a stronger predictor compared to features

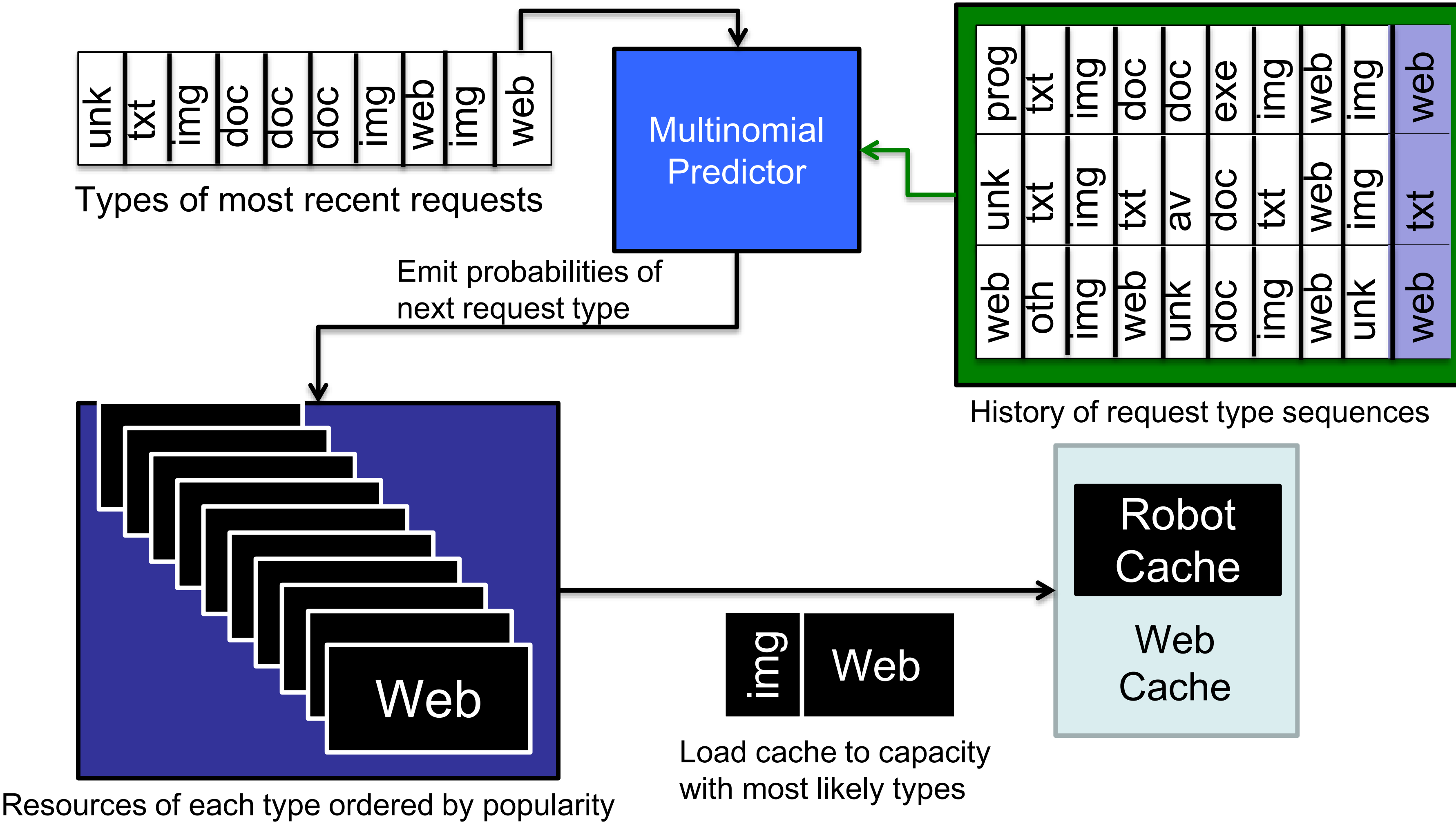
Robot caching policy

- After predicting request type, admit the most frequently requested resources *within that type* into the cache
  - *Power-law* popularity in robot requests: most frequently requested resources are fetched much more often than others

- ❑ If all resources of a type fit in cache, load popular resources of the 2nd most likely type
- ❑ Repeat until cache is at capacity



Robot caching policy



Caching Performance

- Compared performance (hit-ratio) of our predictive policy over robot traffic versus suite of baseline polices
  - Log-size: Store smallest resources; maximize # of resources in cache
  - LRU: Store most recently requested resources, evicting oldest resources
  - Popularity: Evict resources requested least frequently
  - Hyper-G: Evict resources requested least frequently, break ties using LRU
- Popularity-based caches generally used in practice

Caching Performance

Policy	1MB	2MB	3MB	4MB	5MB	8MB	12MB	20MB	40MB
Log-size	.055	.056	.057	.057	.057	.058	.058	.059	.059
LRU	.111	.126	.136	.141	.145	.153	.159	.165	.175
Hyper-G	.174	.178	.172	.180	.176	.188	.189	.212	.236
Pop	**.192**	.204	.206	.205	.205	.205	.223	.224	.282
ENN	.185	**.199**	**.212**	**.220**	**.228**	**.258**	**.284**	**.335**	**.425**
ENN-Gain	-3.4%	-2.5%	3.78%	6.82%	10.1%	20.5%	21.5%	33.1%	33.6%

- Note that improvement in hit-ratio grows just logarithmically with cache size
  - Small % improvement → equivalent to using a worse policy with an exponentially (cost-prohibitive) larger cache
- ENN performance grows even stronger with larger cache size

Outline

- Introduction and motivation
- Analysis of Web robot traffic:
  - Robot detection
  - Performance Optimization: Predictive Caching
- Future research

Future Research

- Automated robot classification
  - Taxonomy of robot times for finer-grained detection
- Workload generation
  - Methods that generate representative streams of intertwined robot and human traffic
- Predictive caching
  - Extension of preliminary results
  - Implementation of real caching algorithm

Very exciting work going on here!

Thank you for your attention!

Questions?