Support Operations Engineering:
Scaling Developer Products to the Millions

Junade Ali - **@IcyApril**


NO NO NO NO NO, MY FRIEND
Service
Rules of Customer Service
SOUTHPARK.CC.COM

CLOUDFLARE

Challenges

- Used by 20+ million web properties
  - Free, self-service, and Enterprise service levels
  - Pro-bono enterprise-grade protection to at-risk Public Interest Groups
- Ever growing customer support requests
  - ~15,000 customer support tickets per month
  - Complex and varied web hosting environments
  - Everyone from florists to Fortune 1000 companies
  - 24x7 TSE coverage

First Support Operations (SOPS) service

- HelperBot Stateless
  - A diagnostics API
- Exposed in many contexts
  - Internal service-to-service
  - API Gateway
  - Customer communication webhooks
- Uses many data sources & active tests

Get additional help

×

Enter Domain and issue | Check diagnostics | File a support ticket | Review ticket confirmation

To help Cloudflare detect common issues and try to resolve your problem quickly, we'll run some diagnostics on your domain. Please wait.
We have found that:

- [redacted] has mixed content errors. Mixed content disrupts content delivery over HTTPS because some of the resources requested are served over HTTP. If this error occurs, a green lock does not appear when serving HTTPS traffic. Review How do I fix the SSL Mixed Content Error Message? before filing a Cloudflare Support ticket.
- The origin web server for [redacted] is responding slowly to initial requests for uncached HTML content. To remove this performance bottleneck, look at your server's ability to return HTML faster and/or implement HTML caching in Cloudflare. You may benefit from caching static HTML and anonymous page views. Learn how to do this in: WordPress, Drupal and Magento.

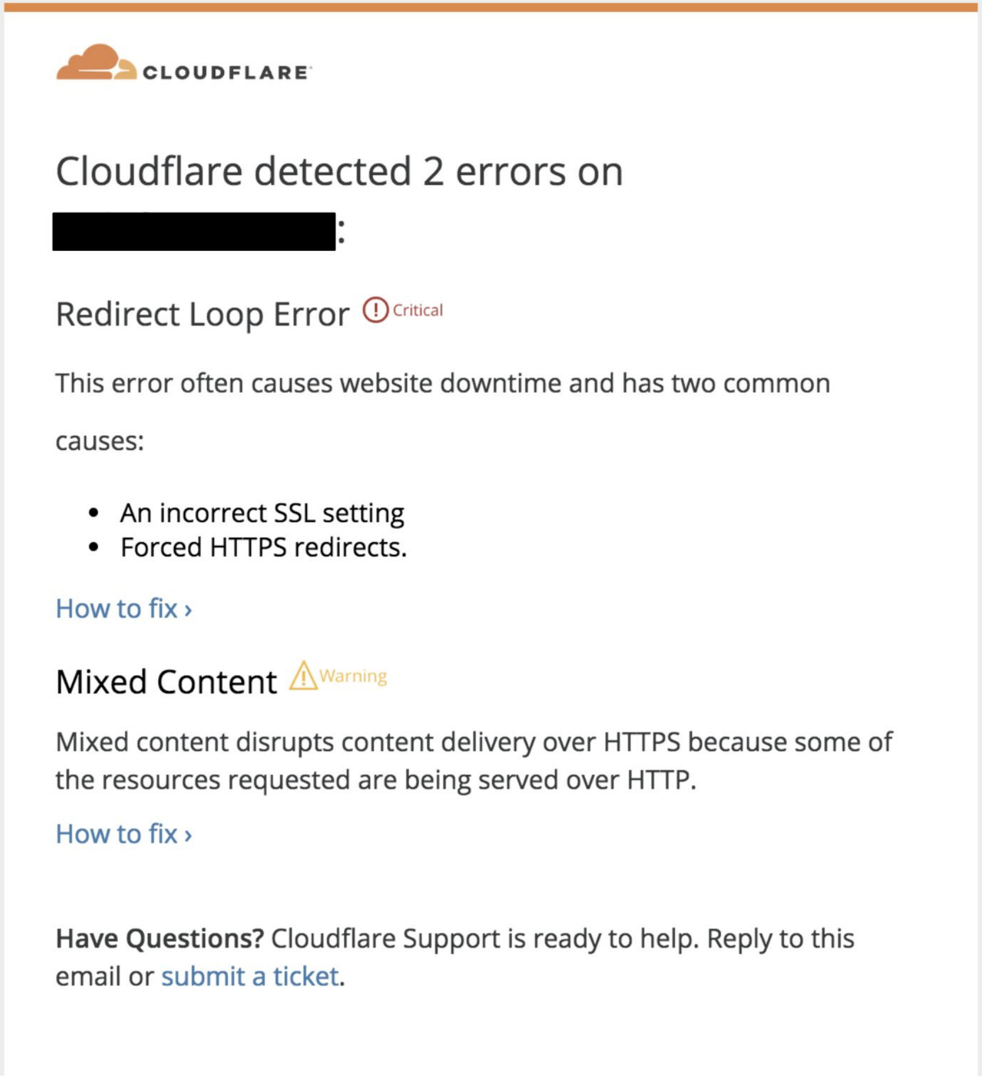
If the recommendations above didn't help resolve your problem, click **Next** to file a Cloudflare Support ticket.

Previous | Cancel | Next

CLOUDFLARE

Campaign Metrics

- Chrome 68 Release
- 91,895 daily tests
- 1 month of human manual testing


CLOUDFLARE

Cloudflare detected 2 errors on [redacted]:

Redirect Loop Error Critical

This error often causes website downtime and has two common causes:

- An incorrect SSL setting
- Forced HTTPS redirects.

How to fix ›

Mixed Content Warning

Mixed content disrupts content delivery over HTTPS because some of the resources requested are being served over HTTP.

How to fix ›

Have Questions? Cloudflare Support is ready to help. Reply to this email or submit a ticket.

The Need for Automation

- Customer Tooling > Agent Tooling
- Tooling != Automation
- Automation > Customer Tooling

NLP is far from perfect…

- State of the Art NLP wasn’t suitable
  - ~70-80% accuracy
  - ~50% for best commercial POC
- Tolerances for false positives vary
  - Free or paid?
  - General question or sensitive issue?


Let´s talk
about it!

Scope for Failure

NLP Pipeline

1. NER
2. Multi-Classifier
3. **Over-Engineering***
4. **Formal Contracts***

* applied depending on risk sensitivity

False Positive Rate:

- Multi-Classifier: **21%**
- Over-Engineering: **1-2%**
- Formal Contracts: **0%**

CLOUDFLARE

Novel Safety Engineering Approaches

- Baseline
  - Failure is tolerable due to majority benefit
  - I.e. Low risk & free user wait time for response
- Binary Classifier
  - Higher risk, but not sensitive
- Formally Defined Safety Checks
  - Sensitive requests
  - May require customer validation actions

2019 6th International Conference on Control, Decision and
Information Technologies (CoDIT'19) | Paris, France / April 23-26, 2019

String similarity algorithms for a ticket classification system

Malgorzata Pikies[1] and Junade Ali[2]

***Abstract*— Fuzzy string matching allows for close, but not exactly, matching strings to be compared and extracted from bodies of text. As such, they are useful in systems which automatically extract and process documents. We summarise and compare various existing algorithms for achieving string similarity measures: Longest Common Subsequence (LCS), Dice coefficient, Cosine Similarity, Levenshtein distance and Damerau distance. Based on previously classified customer support enquiries (tickets), we considered the effectiveness of different algorithms and configurations to automatically identify keywords of interest (such as error phrases, product names and warning messages) in instances where such key phrases are misspelled, copied incorrectly or are otherwise differently formed. An optimal algorithm selection is made based on novel studies of the aforementioned similarity measures on text strings tokenised into characters. Such analysis also allowed for an optimum similarity threshold to be identified for various categories of enquiries, to reduce mismatched strings whilst allowing optimal coverage of the correctly matched key phrases. This led to a 15% improvement in the ratio of false positives to true positive classifications over the existing approach used by a customer support system.**

I. Introduction

Maintaining a high customer satisfaction benchmark is one of the main priorities of every company, and a customer support team is often the primary frontier for customers to contact a business. In order to provide the best customer service, agents have to prioritise tickets, reply quickly and accurately. With a growing customer base, the average waiting time for a reply can elongate. Classifiers based on string matching algorithms can shorten a ticket response time, hence help with agents' performance and reduce costs of the business. In practice it can be accomplished by using an automatic classification system linked with a database of replies to the most frequent enquiries or run technical diagnostics based on the error information provided by a customer. A system like that can immediately (subject to the text processing time) reply to tickets, which can reduce workload of customer support agents.

The purpose of this paper is an introduction to mechanisms behind the chosen string similarity algorithms. Given two strings (sequences of characters) X and Y, the difficulty of finding a quantity to measure the relationship between them comes down to two things. One is finding the correct similarity function $S(X,Y)$ and the second is finding a threshold $t_{S/D}$. Based on these values, two strings can be classified as

- similar: $S(X,Y) \geq t_S$ or $D(X,Y) \leq t_D$,
- different: $S(X,Y) < t_S$ or $D(X,Y) > t_D$,

where $D(X,Y)$ is a string dissimilarity function.

II. Current knowledge

There are numerous examples of fuzzy string matching or string similarity algorithms being used in customer support environments for extracting relevant information. [1] proposed an automated labelling system for bug trackers and customer support. They describe their recurrent neural network solution, where the text is tokenised into vectors of words and sentences. [2] describes using a Natural Language Processing (NLP) based tool for a keyword extraction and mentions usage of the Levenshtein distance for word matching, yet the study focuses on the enhancement of the Machine Learning (ML) tagger with a Twitter model using previous customer service interactions. [3] uses word and character embeddings with neural models. They compare different linking methods with the fuzzy string matching, which computes the Levenshtein Distance between their queries using support tickets. Despite using approaches for string similarity, there has been very little existing work comparing string similarity techniques or their configuration parameters when used in customer support automation.

String classification systems has been studied to label and understand a variety of text strings. Prior to any string analysis one has to decide on the text tokenisation. A string of text can be divided into items, such as words, phrases, letters etc. Items can be used to create *n*-grams. A set of all strings of an integer length n, in a finite alphabet Σ is denoted by Σ^n. An *n*-gram (sometimes called a shingle or a *q*-gram) based on letters is simply an any string from Σ^n [5]. In practice, a sequence of *n*-grams is created from a text of interest (see Tab. I for examples). Once strings are divided into substrings, the measurement of their similarity is possible.

To the best of our knowledge, there exist a gap in the literature that we want to fill. This paper is the first one to compare performance of different string similarity algorithms for a keyword extraction using test strings tokenised into characters.

TABLE I

n-GRAM EXAMPLES FOR A STRING "ALAMAKOTA" TOKENISED INTO LETTERS.

Name	*n*	*n*-grams
unigram	1	(a, l, a, m, a, k, o, t, a)
bigram	2	(al, la, am, ma, ak, ko, ot, ta)
trigram	3	(ala, lam, ama, mak, ako, kot, ota)

[1]Malgorzata Pikies is with Cloudflare, London, United Kingdom malgorzata@cloudflare.com
[2]Junade Ali is with Cloudflare, London, United Kingdom junade@cloudflare.com

Algoritm	n	TP, [%]	FP, [%]	FP/TP
Cosine	1	100.0	99.0	0.99
	2	89.0	21.0	0.24
	3	81.5	19.0	0.23
Dice	1	100.0	98.0	0.98
	2	89.5	21.5	0.24
	3	88.0	20.0	0.23
Damerau		92.5	26.5	0.29
LCS		93.5	32.0	0.34
Levenshtein		92.5	26.5	0.29

$$s(X,Y) = \frac{\vec{U}(X) \cdot \vec{V}(Y)}{|\vec{U}(X)||\vec{V}(Y)|} = cos\theta,$$

https://ieeexplore.ieee.org/abstract/document/8820497/

Over-Engineering for Safety

- Binary Classification
  - Cascading failure to reduce false positives
  - Non-sensitive requests by paying users
  - Convolutional Neural Network
- Use of Diagnostics
  - Corresponding failed diagnostics is also tolerable

Cascading Failure can be a good thing…

Table 4: True (TP) and false (FP) positive matches, and their ratio. 'DNS' and 'Crypto' tickets matched for the 'DNS' category.

Algoritm	Keyword classification			Binary classification		
	TP, [%]	FP, [%]	FP/TP	TP, [%]	FP, [%]	FP/TP
Cosine	80.12	22.59	0.2820	25.54	1.72	0.0674
Dice	80.41	22.67	0.2820	25.54	1.72	0.0674
Damerau	81.09	25.03	0.3087	25.40	1.77	0.0697
LCS	82.46	28.21	0.3421	25.83	2.02	0.0780
Levenshtein	80.99	25.07	0.3095	25.44	1.77	0.0697

Table 5: True (TP) and false (FP) positive matches, and their ratio. 'Errors' and 'DNS' tickets matched for the 'Errors' category.

Algoritm	Keyword classification			Binary classification		
	TP, [%]	FP, [%]	FP/TP	TP, [%]	FP, [%]	FP/TP
Cosine	53.26	1.95	0.0366	36.82	0.49	0.0132
Dice	53.04	1.95	0.0368	36.71	0.49	0.0133
Damerau	52.72	2.24	0.0425	36.71	0.49	0.0133
LCS	53.15	3.7	0.0697	36.82	0.49	0.0132
Levenshtein	52.83	2.34	0.0443	36.71	0.49	0.0133

Formally Defined Run-Time Contracts

How?

1. Contracts + data stored
2. Customer validation
3. Contracts revalidated
4. Downstream APIs revalidate

Failure cases halt processing and remove data fields to prevent software errors.

Expected failures linked to JIRAs, unexpected to Sentry/PagerDuty.


```
{
  "contracts": {
    "[redacted]_enabled": true,
    "has_active_zones": true,
    "has_ent_zones": true,
    "json_valid_schema_generated_date": true,
    "json_valid_schema_[redacted]_codes": true,
    "json_valid_schema_support_ticket": true,
    "json_valid_schema_user_id": true,
    "using_[redacted]": false,
    "valid_action_parameter": true,
    "valid_json": true,
    "valid_user_exists": true
  },
  "result": {
    "failed_zone_ids": [],
    "[redacted]_permissible": true,
    "[redacted]_success": false,
    "total_zones": 2,
    "zones_with_token": 2
  }
}
```

Data Matters

- Simplified taxonomy
  - Encourages greater accuracy
- Classification to fill in the gaps
  - Used to add additional dimensions to reporting
- Make everything self-serve
  - Attach repeat configuration change items to JIRAs

Error 525 and Error 520

Yesterday 12:45 pm

Cloudflare Support Team Yesterday 12:49 pm (assign)

Helperbot

Automated test failed!

Zones Detected:


```
[{"user_id": [redacted], "zone_id": [redacted], "zone_name":
"[redacted]", "zone_status": "V"}]
```


Helperbot Test

Name: `server_errors_metrics`
Return Code: `5xx_errors_high`
Data:


```
{"percent_4xx": 0.0, "percent_5xx": 58.0, "raw": {"200": 8, "301": 5,
"304": 1, "520": 3, "525": 16}, "total_req": 33}
```


Zone Tested:


```
{"user_id": [redacted], "zone_id": [redacted], "zone_name":
"[redacted]", "zone_status": "V"}
```


Stateless Helperbot: Run test again

Cloudflare Support Team Yesterday 12:49 pm (assign)

Hi there,

Thanks for writing to Cloudflare Support.

Sorry to hear you are experiencing some difficulties here.

We have run some automated tests and we can see that there was a 525 error when accessing ankitbanerjee.in.

A 525 error indicates that Cloudflare is not able to complete a SSL/TLS handshake with your web server. If you are seeing this error these are the common causes and the steps you can take to resolve the issue.

- Your origin server does not have a certificate installed.
- The cipher suitesthat Cloudflare accepts and the cipher suites that the origin server supports do not match.
- If you are only intermitently seeing 525's this suggests the TCP connection between Cloudflare and your origin is being reset during the SSL handshake causing the error.

Here is what we recommend in order to ensure all requests from Cloudflare are accepted by your server over HTTPS

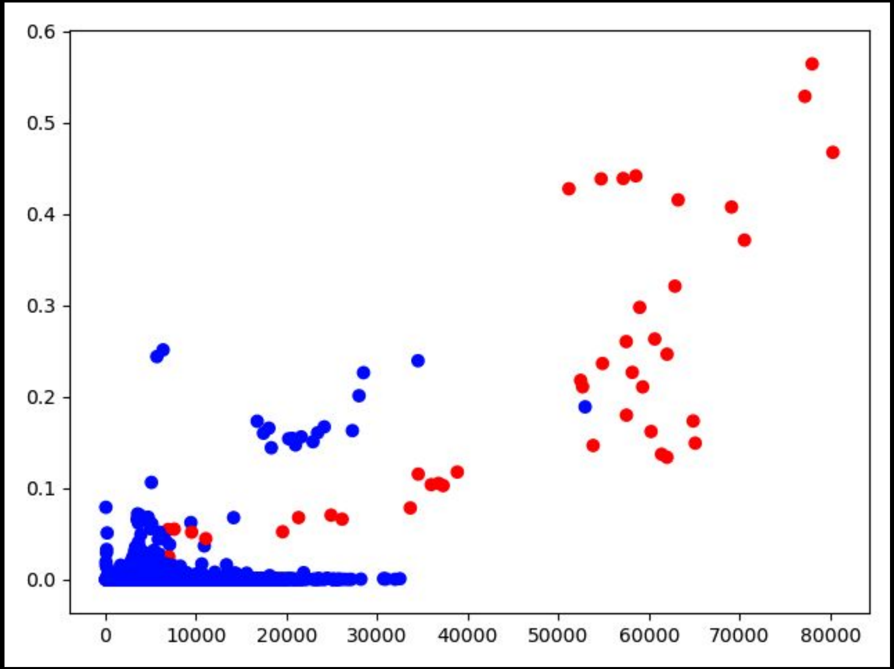
- Pause pause Cloudflare or update your local hosts file to point directly at your server IP to test that your server is presenting a SSL certificate. If you do not have a certificate installed on your server you can generate one using our Origin CA certificates. This are free certificates for the purpose of encrypting the connection between Cloudflare and your web server, so that you do not need to purchase a certificate.
- Review the cipher suites your server is using to ensure they match what is supported by Cloudflare.
- Check your server's error logs from the timestamps you see 525s to ensure there

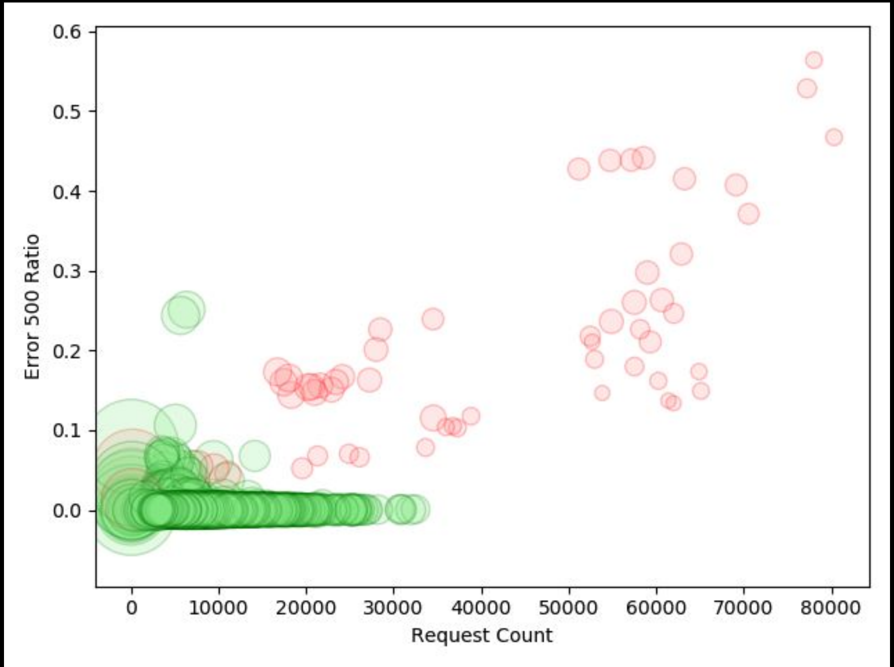
CLOUDFLARE

Next-Gen Security Operations Centre

- Proactive messaging for self-serve users
- Can same be applied to a SOC?
    - Active testing
    - Analysis of passive traffic data flow

Multi-Dimensional Visibility

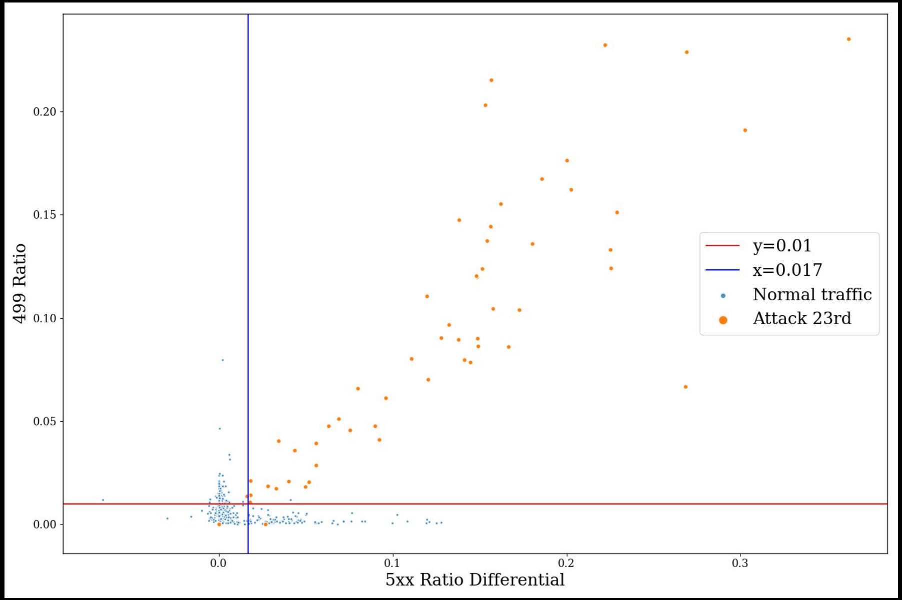


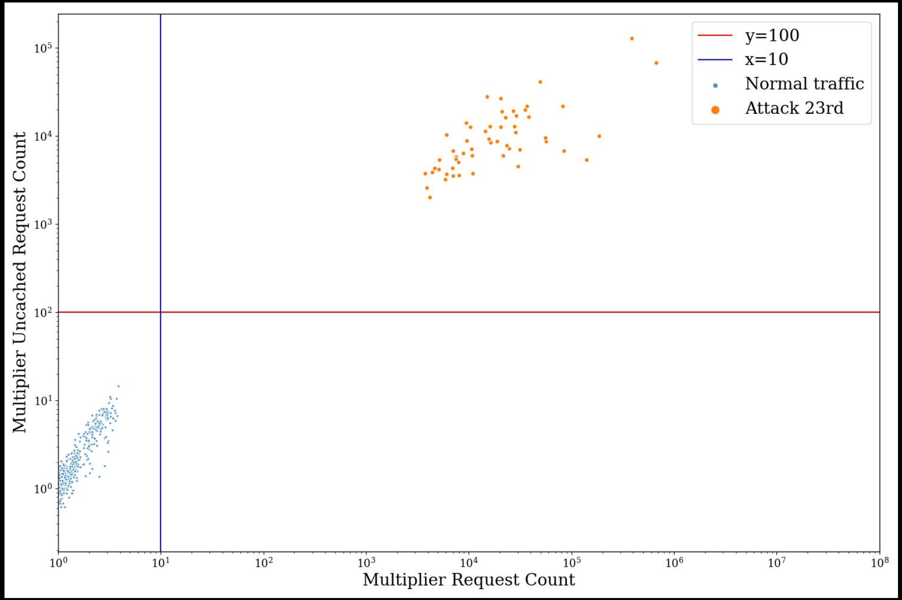


Colour	= path ratio < 0.554 in red
Scatter size	= UA ratio*2500

CLOUDFLARE

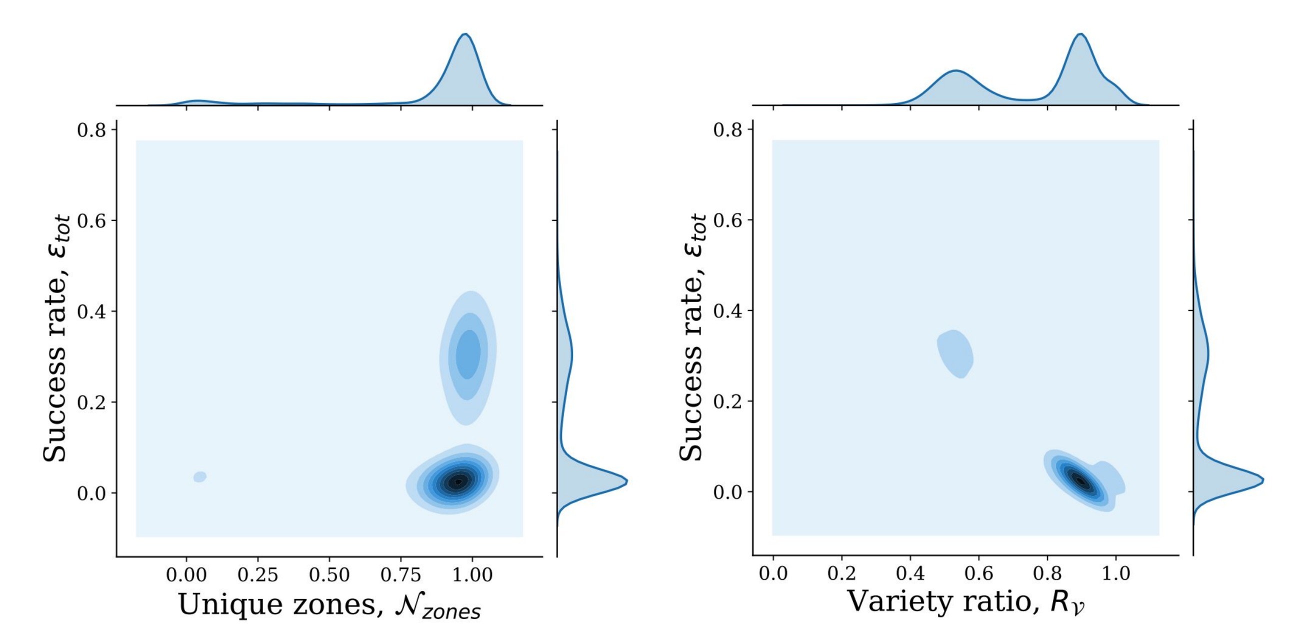
Additional properties for disambiguation





CLOUDFLARE

Intelligent Threat Fingerprinting



E_{tot} - success rate of brute force attack

R_V - abnormal HTTP status (429, 5xx, etc)

$\mathcal{N}_{zones}$ - normalised sites attacked

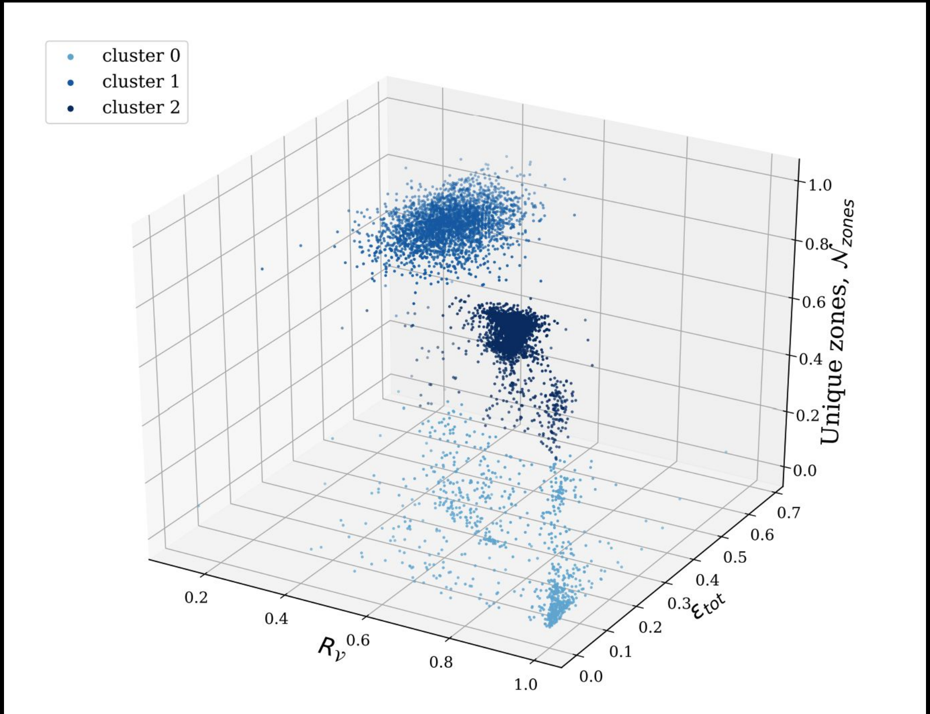
CLOUDFLARE

Intelligent Threat Fingerprinting

On these 3 aggregate properties, unsupervised clusterization is able to correlate to fingerprint of attack.

E.g. Cluster 1 (highest success):

- median success rate of 30.5%
- 99.5% req from same UA
- 99.45% same country



CLOUDFLARE

Current state

- HelperBot formed of 6 services
  - From chatbot to SOC anomaly detection
  - 10 ancillary SOPS services
- Metrics
  - TSF: 57.3% deflection (excl. email tickets)
  - HelperBot: ~60% free ticket automation
  - ~78% without human interaction
- Plenty more to do
  - 24% of *all* tickets automated
  - 35% planned EOY ‘19, 50% in ‘20
  - Groundwork laid to drive ever greater automation

SOPS Principles

- Favour automation over tooling
- Question the fundamentals
- Context-Sensitive Safety
- Be diligently data driven
- Build services as an asset

Thank you!

Get in contact:
@IcyApril
junade@cloudflare.com