Practical cache-based side channel attacks with JavaScript

Jack Cook

Last time…

Replicating the original paper

- Shusterman et al. found that many websites exhibit highly unique cache contention patterns
- While a website loads (in a separate tab/window), we can repeatedly access values from memory and measure how long it takes to retrieve them
- These can be visualized in a “memorygram”, where darker areas indicate more cache evictions over time
- The uniqueness of these “memorygrams” can be exploited -- a model trained on these can learn which website is being accessed

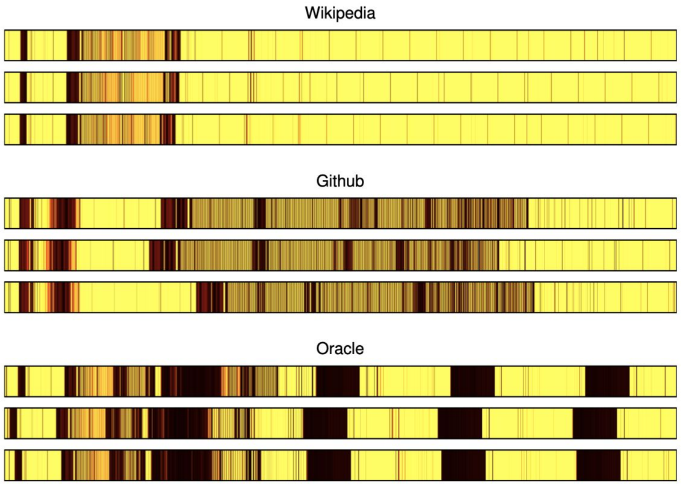
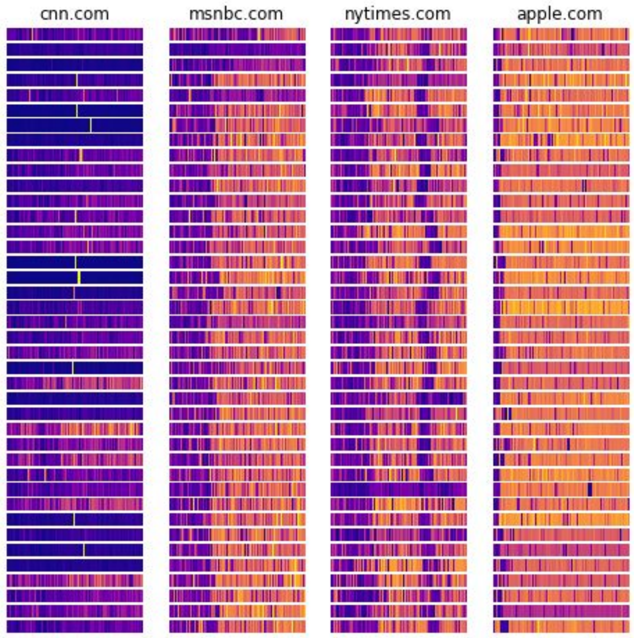


Figure 3: Examples of memorygrams. Time progresses from left to right, shade indicates the number of evictions. (Darker shades correspond to more eviction.)

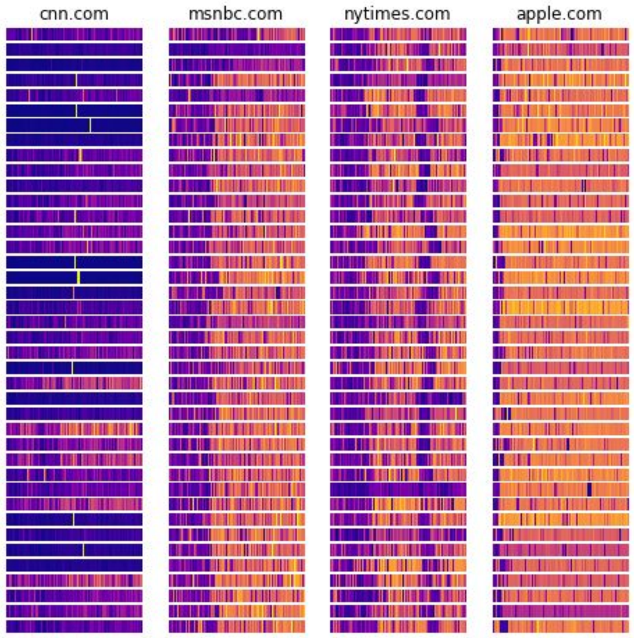
Preliminary Results

- Most success with random forest models, which can be translated into JavaScript after they're trained in Python
  - Makes the results a bit cooler -- the website you're on is displayed as soon as you open the page
- If you want to detect the user opening between a small number of (around 4) websites, my work here is done
  - 100% accuracy when distinguishing between basically any set of 4 websites that I tried
- At 10 different websites, accuracy drops to 90%
- Still collecting data to distinguish between 100 different websites, which the original paper detected with 90% accuracy



Preliminary Results (cont.)

- The original paper made each trace 30 seconds long -- I've found you can get almost all of the accuracy with about 2 seconds on a good Internet connection



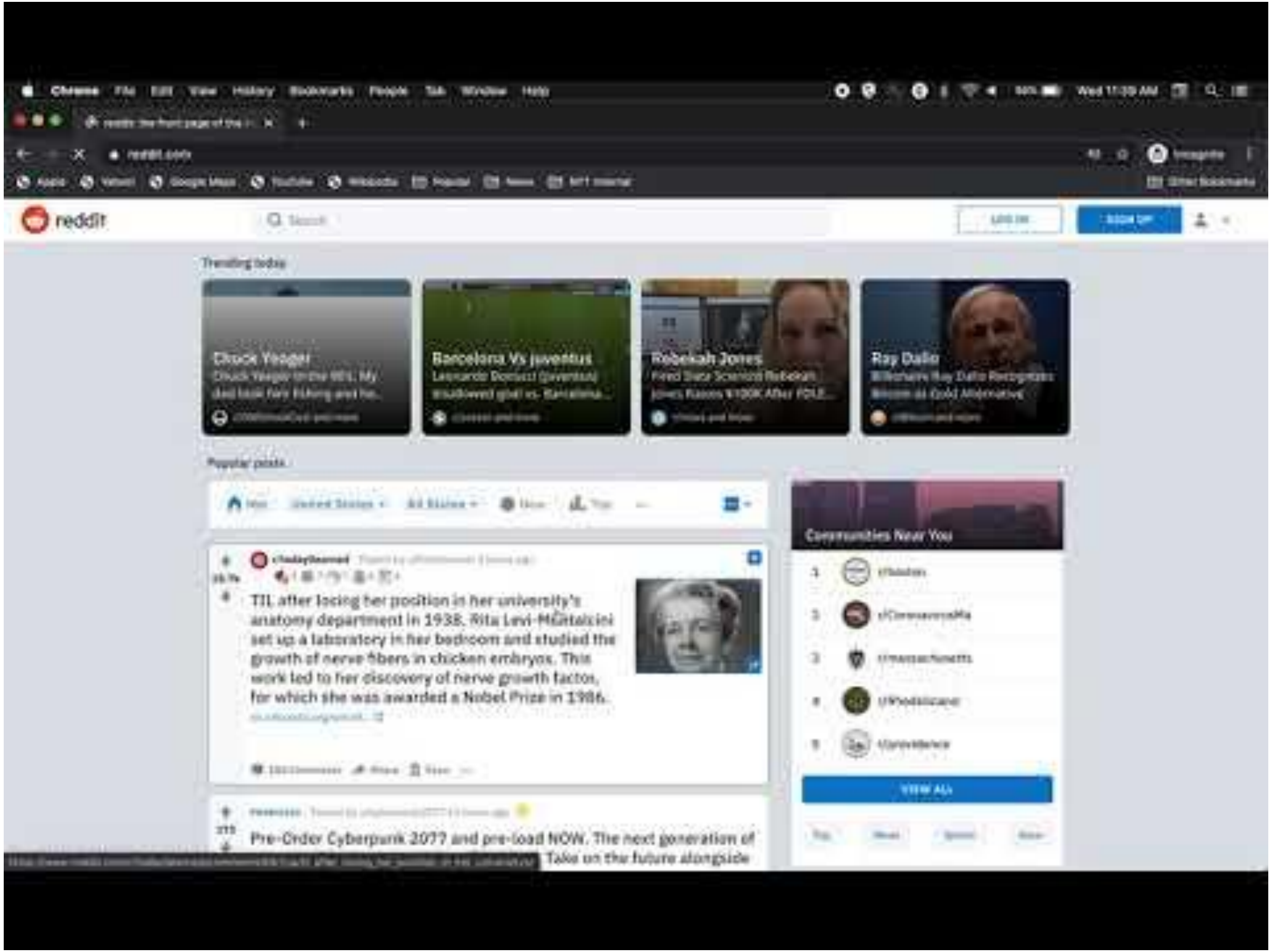
How it works

How to collect a website trace

- I found that trying to measure op/s gave better results than measuring cache contention
  - Also makes my code much easier to read
- Best results when this part was compiled to WASM
  - Interestingly, fewer op/s from WASM than JS, but results must have been more reliable

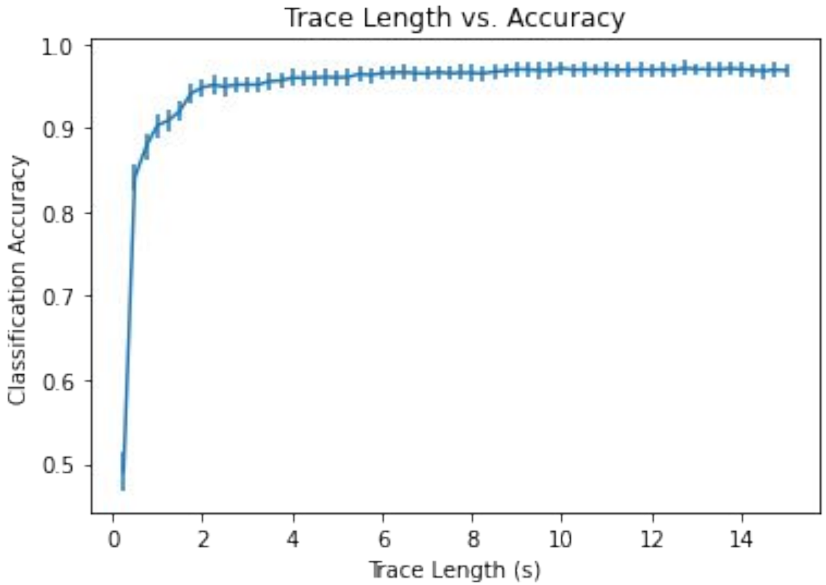
[REDACTED]

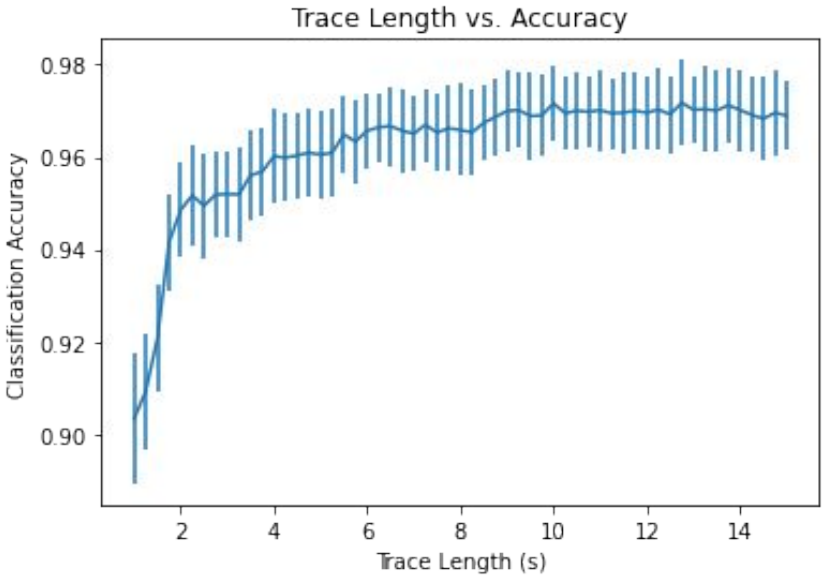
Demo (kind of)


Chrome File Edit View History Bookmarks People Tab Window Help
reddit
Trending today
Chuck Yeager
Barcelona Vs Juventus
Rebekah Jones
Ray Dalio
Popular posts
Communities Near You
TIL after losing her position in her university's anatomy department in 1938, Rita Levi-Montalcini set up a laboratory in her bedroom and studied the growth of nerve fibers in chicken embryos. This work led to her discovery of nerve growth factor, for which she was awarded a Nobel Prize in 1986.
VIEW ALL
Pre-Order Cyberpunk 2077 and pre-load NOW. The next generation of
Take on the future alongside

Results

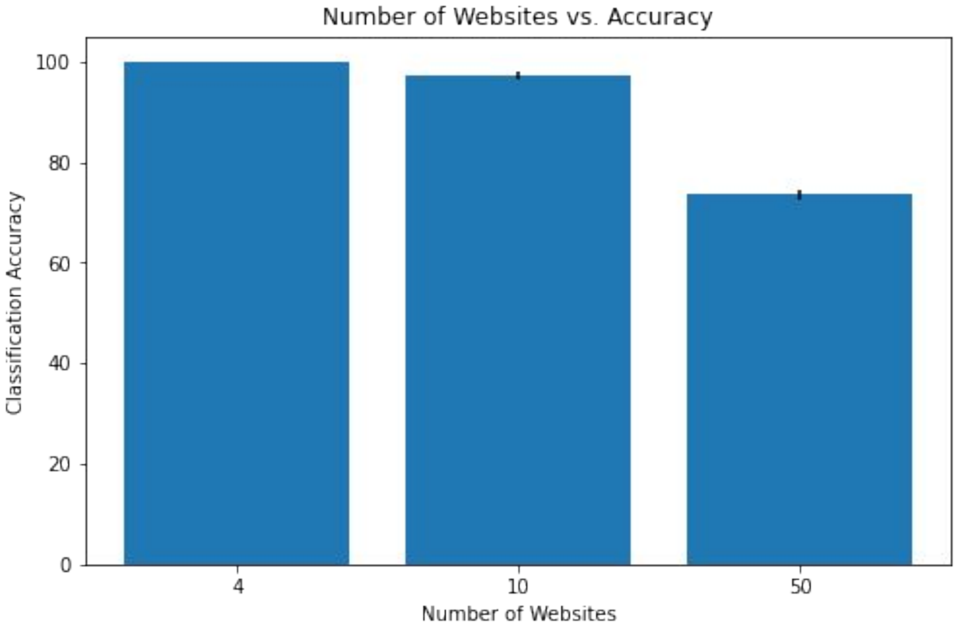
How long should traces be?





How many websites can we classify?

- Results from this morning!
- Accuracy is usually around 97% when classifying between the Alexa top 10
- When classifying between the Alexa top 50, accuracy drops to 74% (not amazing, but remember: our random choice baseline is 2%)



Can we predict traces on new computers?

- Up until now, I’ve been collecting training and testing data on my own computer
  - This gives great results, but is not representative of how this attack would probably be pulled off in the real world
- I collected testing data on my roommate’s Dell XPS 13 once, and the results were discouraging
  - Can only speculate why this is -- got unlucky? Differences due to OS? CPU?
- What if I could get data from a bunch of the same type of computer?
- Can I collect data on one MacBook Pro, and make accurate predictions on another identical MacBook Pro?

I asked all my friends to collect data

Laptops

Name	Screen ...	Year	Process...	OS Ver...	Chrome...
Allen	13	2020	2 GHz i5	Catalina	86
Angela	13	2017	2.3 GHz i5	High Sierra	87
Jennifer	13	2017	2.3 GHz i5	Catalina	87
Anna	13	2018	2.3 GHz i5	Catalina	87
Gwynnie	13	2017	2.3 GHz i5	Catalina	87
Katherine	13	2018	2.3 GHz i5	Catalina	87
Jamie	16	2019	2.3 GHz i9	Catalina	87
Hassan	16	2019	2.3 GHz i9	Big Sur	86
Julia	15	2016	2.6 GHz i7	Big Sur	87
Britney	16	2019	2.6 GHz i7	Catalina	87
Hannah	16	2019	2.6 GHz i7	Catalina	87
Kevin	15	2018	2.6 GHz i7	Catalina	87
Natalie	13	early 2015	2.7 GHz i5	Catalina	87
Eric	13	early 2015	2.7 GHz i5	High Sierra	87
Soomin	13	2018	2.7 GHz i7	Catalina	87
Ethan	15	2017	2.9 GHz i7	Catalina	87
NYT	13	2017	3.1 GHz i5	Catalina	87
Claire	13	2017	3.1 GHz i5	Catalina	87
Dad	13	2017	3.5 GHz i7	Catalina	87

Snoopy Setup Instructions

1. Make sure you have npm installed
   - If nothing comes up when you type `npm` in Terminal, install it here: https://www.npmjs.com/get-npm
2. Install selenium if you don't already have it
   - Enter `pip install selenium` into your Terminal
   - Then, open Chrome and go to chrome://version to check your Chrome version
   - Download ChromeDriver according to your Chrome version from https://sites.google.com/a/chromium.org/chromedriver/downloads
   - Move chromedriver into your path, e.g. `sudo mv ~/Downloads/chromedriver /usr/local/bin`
3. Clone the project

```
git clone https://github.com/jackcook/snoopy
```

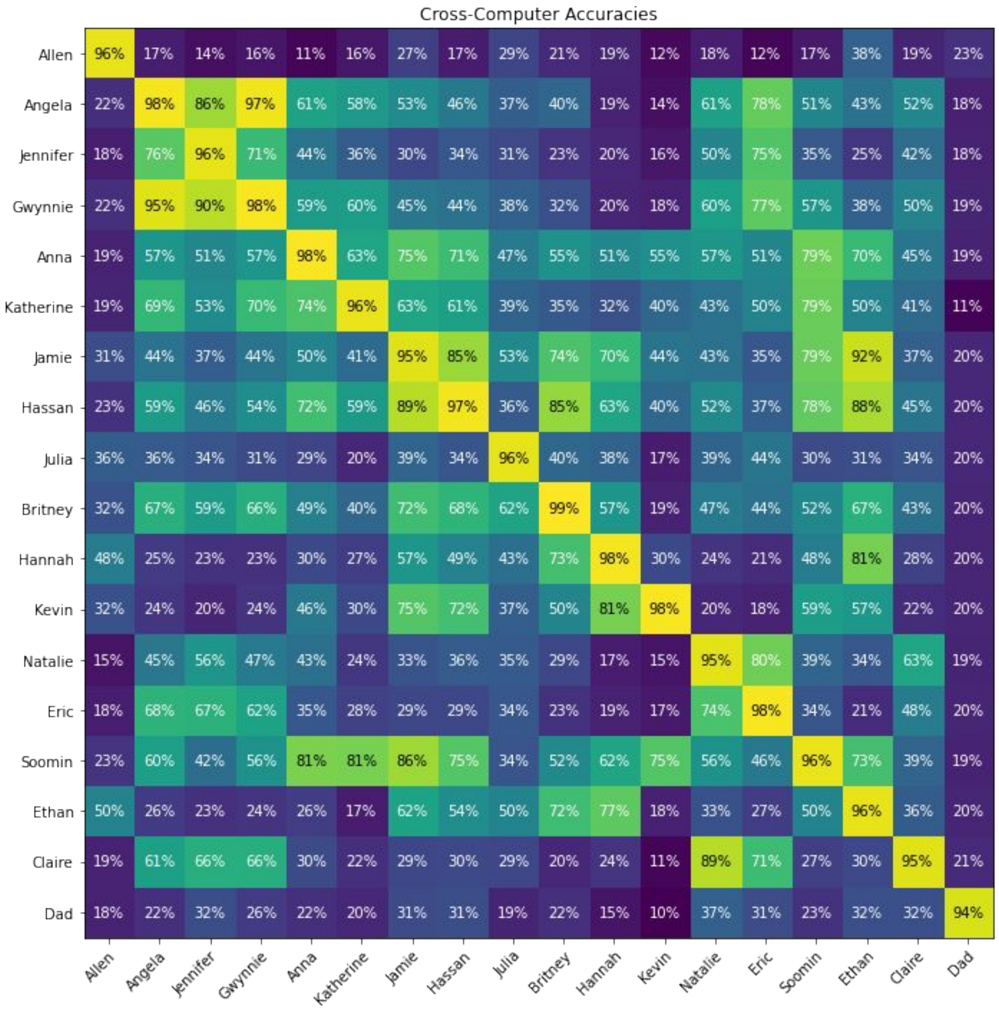
4. `cd` into the cloned directory and install dependencies

```
cd snoopy
npm install
```

5. Start the webserver

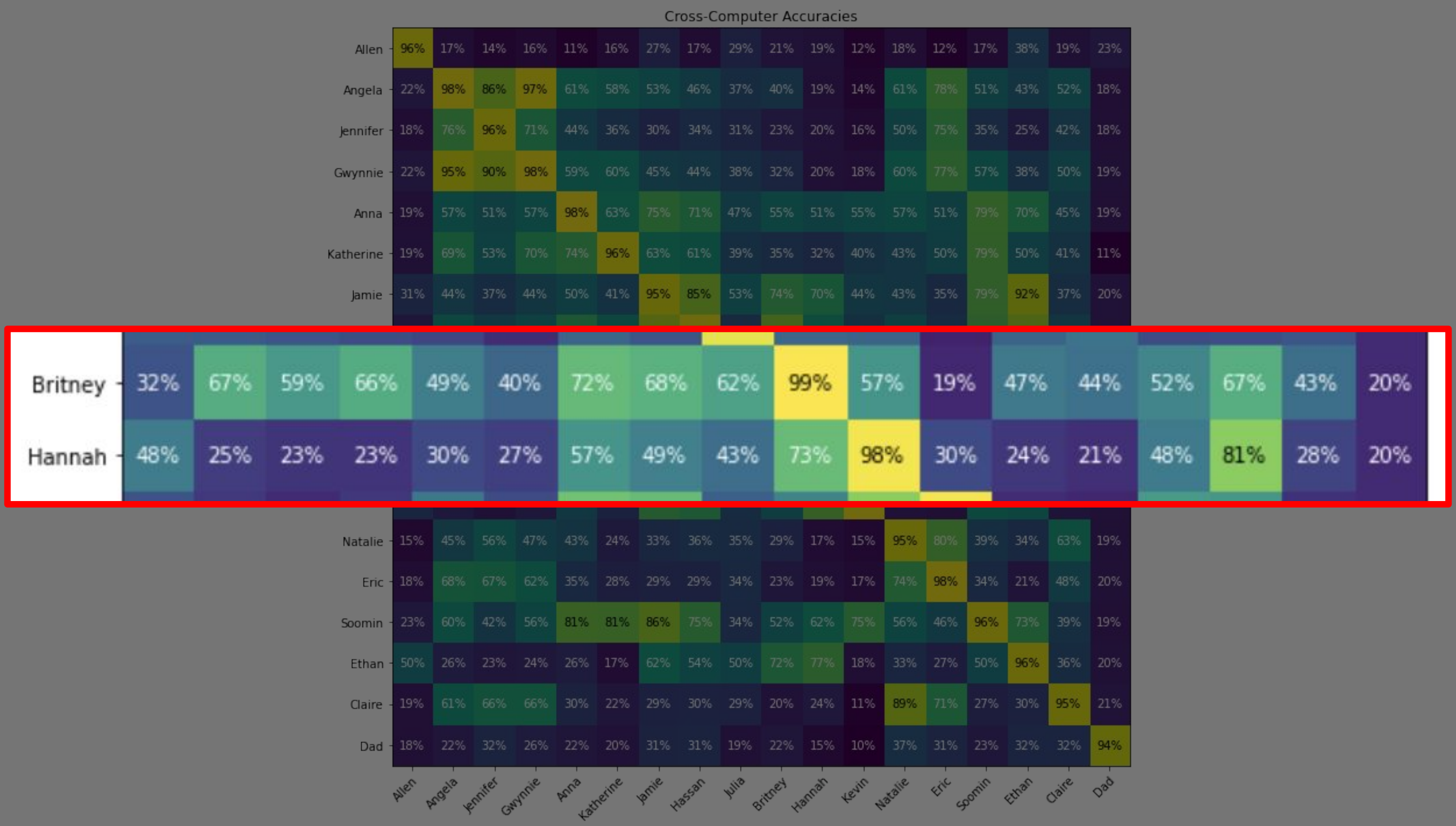
```
npm start
```

6. Make sure you're ready to collect data:
   1. Plug your laptop into its charger
   2. Disable your screen saver: In System Preferences, go to **Desktop & Screen Saver** and select "Start after: Never"

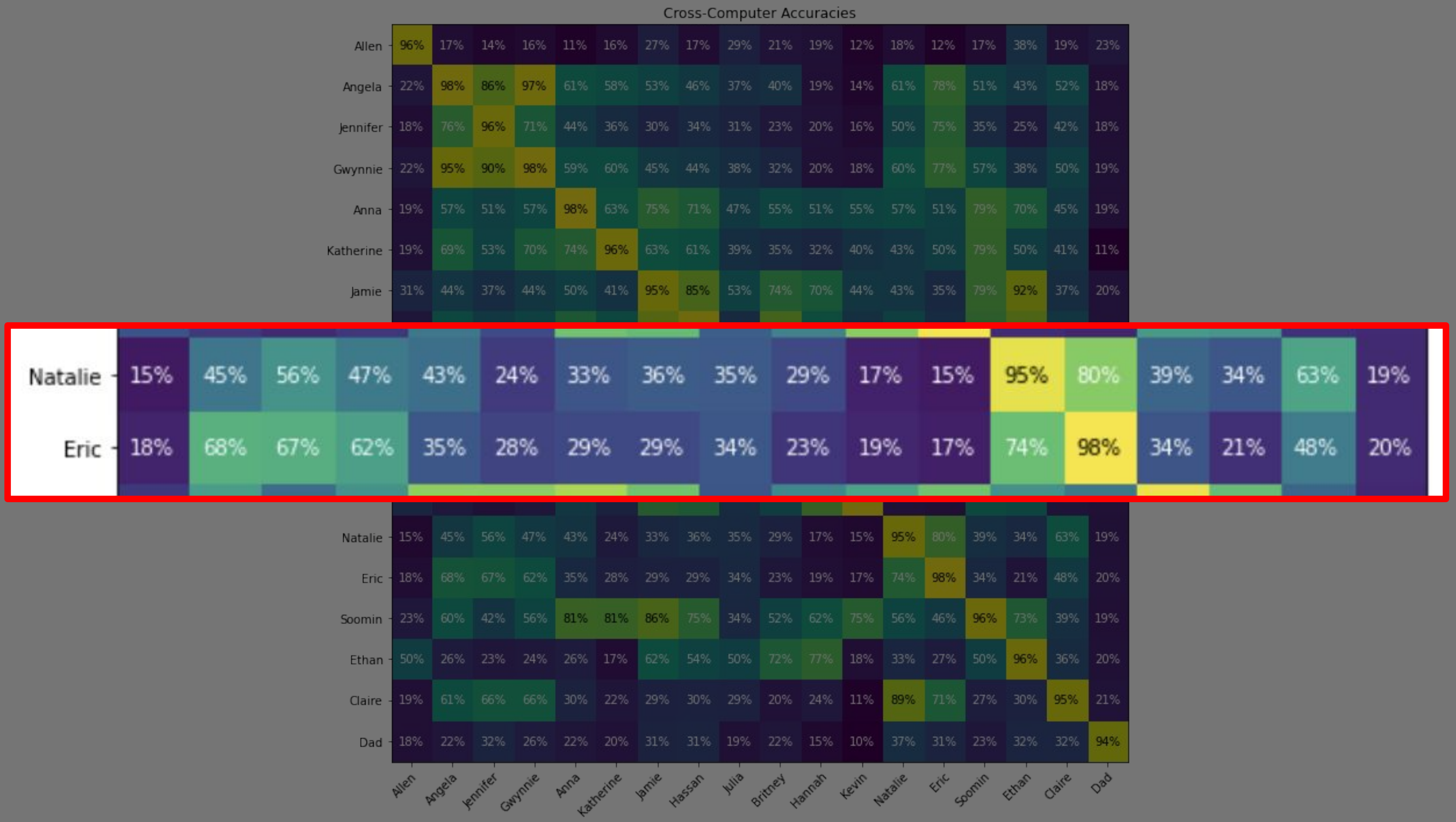

Cross-Computer Accuracies
Allen 96% 17% 14% 16% 11% 16% 27% 17% 29% 21% 19% 12% 18% 12% 17% 38% 19% 23%
Angela 22% 98% 86% 97% 61% 58% 53% 46% 37% 40% 19% 14% 61% 78% 51% 43% 52% 18%
Jennifer 18% 76% 96% 71% 44% 36% 30% 34% 31% 23% 20% 16% 50% 75% 35% 25% 42% 18%
Gwynnie 22% 95% 90% 98% 59% 60% 45% 44% 38% 32% 20% 18% 60% 77% 57% 38% 50% 19%
Anna 19% 57% 51% 57% 98% 63% 75% 71% 47% 55% 51% 55% 57% 51% 79% 70% 45% 19%
Katherine 19% 69% 53% 70% 74% 96% 63% 61% 39% 35% 32% 40% 43% 50% 79% 50% 41% 11%
Jamie 31% 44% 37% 44% 50% 41% 95% 85% 53% 74% 70% 44% 43% 35% 79% 92% 37% 20%
Hassan 23% 59% 46% 54% 72% 59% 89% 97% 36% 85% 63% 40% 52% 37% 78% 88% 45% 20%
Julia 36% 36% 34% 31% 29% 20% 39% 34% 96% 40% 38% 17% 39% 44% 30% 31% 34% 20%
Britney 32% 67% 59% 66% 49% 40% 72% 68% 62% 99% 57% 19% 47% 44% 52% 67% 43% 20%
Hannah 48% 25% 23% 23% 30% 27% 57% 49% 43% 73% 98% 30% 24% 21% 48% 81% 28% 20%
Kevin 32% 24% 20% 24% 46% 30% 75% 72% 37% 50% 81% 98% 20% 18% 59% 57% 22% 20%
Natalie 15% 45% 56% 47% 43% 24% 33% 36% 35% 29% 17% 15% 95% 80% 39% 34% 63% 19%
Eric 18% 68% 67% 62% 35% 28% 29% 29% 34% 23% 19% 17% 74% 98% 34% 21% 48% 20%
Soomin 23% 60% 42% 56% 81% 81% 86% 75% 34% 52% 62% 75% 56% 46% 96% 73% 39% 19%
Ethan 50% 26% 23% 24% 26% 17% 62% 54% 50% 72% 77% 18% 33% 27% 50% 96% 36% 20%
Claire 19% 61% 66% 66% 30% 22% 29% 30% 29% 20% 24% 11% 89% 71% 27% 30% 95% 21%
Dad 18% 22% 32% 26% 22% 20% 31% 31% 19% 22% 15% 10% 37% 31% 23% 32% 32% 94%
Allen Angela Jennifer Gwynnie Anna Katherine Jamie Hassan Julia Britney Hannah Kevin Natalie Eric Soomin Ethan Claire Dad


Cross-Computer Accuracies
Allen
Angela
Jennifer
Gwynnie
Anna
Katherine
Eric
Soomin
Ethan
Claire
Dad
Allen
Angela
Jennifer
Gwynnie
Anna
Katherine
Jamie
Hassan
Julia
Britney
Hannah
Kevin
Natalie
Eric
Soomin
Ethan
Claire
Dad
Angela 22% 98% 86% 97% 61% 58% 53% 46% 37% 40% 19% 14% 61% 78% 51% 43% 52% 18%
Jennifer 18% 76% 96% 71% 44% 36% 30% 34% 31% 23% 20% 16% 50% 75% 35% 25% 42% 18%
Gwynnie 22% 95% 90% 98% 59% 60% 45% 44% 38% 32% 20% 18% 60% 77% 57% 38% 50% 19%

Cross-Computer Accuracies
Allen
Angela
Jennifer
Gwynnie
Anna
Katherine
Jamie
Natalie
Eric
Soomin
Ethan
Claire
Dad
Allen Angela Jennifer Gwynnie Anna Katherine Jamie Hassan Julia Britney Hannah Kevin Natalie Eric Soomin Ethan Claire Dad
Jamie 31% 44% 37% 44% 50% 41% 95% 85% 53% 74% 70% 44% 43% 35% 79% 92% 37% 20%
Hassan 23% 59% 46% 54% 72% 59% 89% 97% 36% 85% 63% 40% 52% 37% 78% 88% 45% 20%

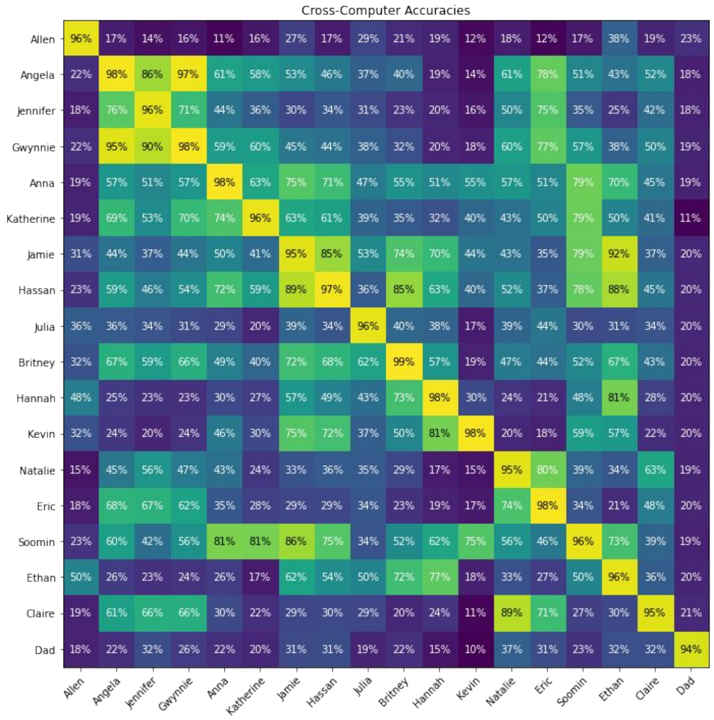


Cross-Computer Accuracies
Britney 32% 67% 59% 66% 49% 40% 72% 68% 62% 99% 57% 19% 47% 44% 52% 67% 43% 20%
Hannah 48% 25% 23% 23% 30% 27% 57% 49% 43% 73% 98% 30% 24% 21% 48% 81% 28% 20%
Allen
Angela
Jennifer
Gwynnie
Anna
Katherine
Jamie
Natalie
Eric
Soomin
Ethan
Claire
Dad
Allen Angela Jennifer Gwynnie Anna Katherine Jamie Hassan Julia Britney Hannah Kevin Natalie Eric Soomin Ethan Claire Dad


Cross-Computer Accuracies
Natalie 15% 45% 56% 47% 43% 24% 33% 36% 35% 29% 17% 15% 95% 80% 39% 34% 63% 19%
Eric 18% 68% 67% 62% 35% 28% 29% 29% 34% 23% 19% 17% 74% 98% 34% 21% 48% 20%
Allen
Angela
Jennifer
Gwynnie
Anna
Katherine
Jamie
Natalie
Eric
Soomin
Ethan
Claire
Dad
Allen Angela Jennifer Gwynnie Anna Katherine Jamie Hassan Julia Britney Hannah Kevin Natalie Eric Soomin Ethan Claire Dad

Predicting traces on new computers

- Even with training data from just one computer, I can get accuracies as high as 97% on other identical computers!
- Accuracy improved further when I combined data from multiple people with similar specs
    - Anna and Katherine have identical computers, but training on Anna's data and testing on Katherine's only gave 74% accuracy
    - When I trained on all 4 computers with 2.3 GHz i5 processors, accuracy jumped to 87%



Challenges

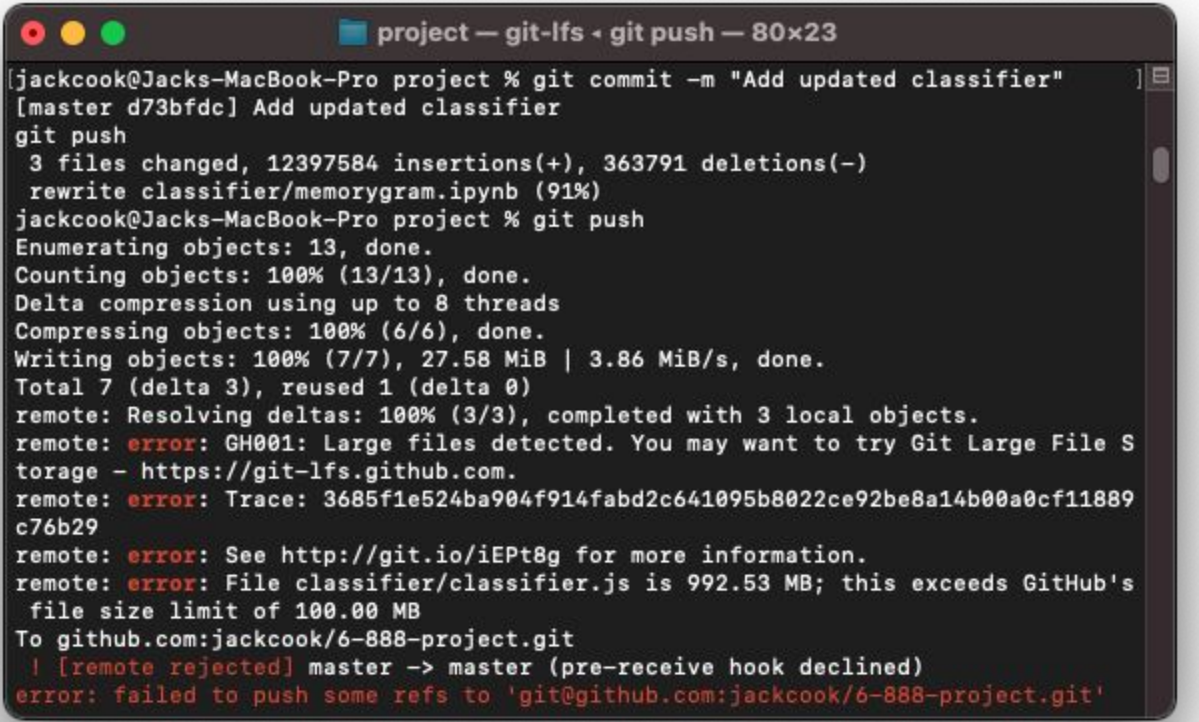
- Same problem as before: is poor prediction accuracy due to bad data, or a bad model?
- It's hard to tell whether the way I'm making traces is the best one
  - My only way to tell if something improved is to collect data for many hours
  - This meant waking up, making a small adjustment to my trace collection, letting it run all day, making another adjustment in the evening, and then running it overnight while I was asleep… and then doing this for weeks
  - Probably wouldn't have been possible if I didn't have an old laptop with me
- Data is super noisy, and this is without any programs running in the background!

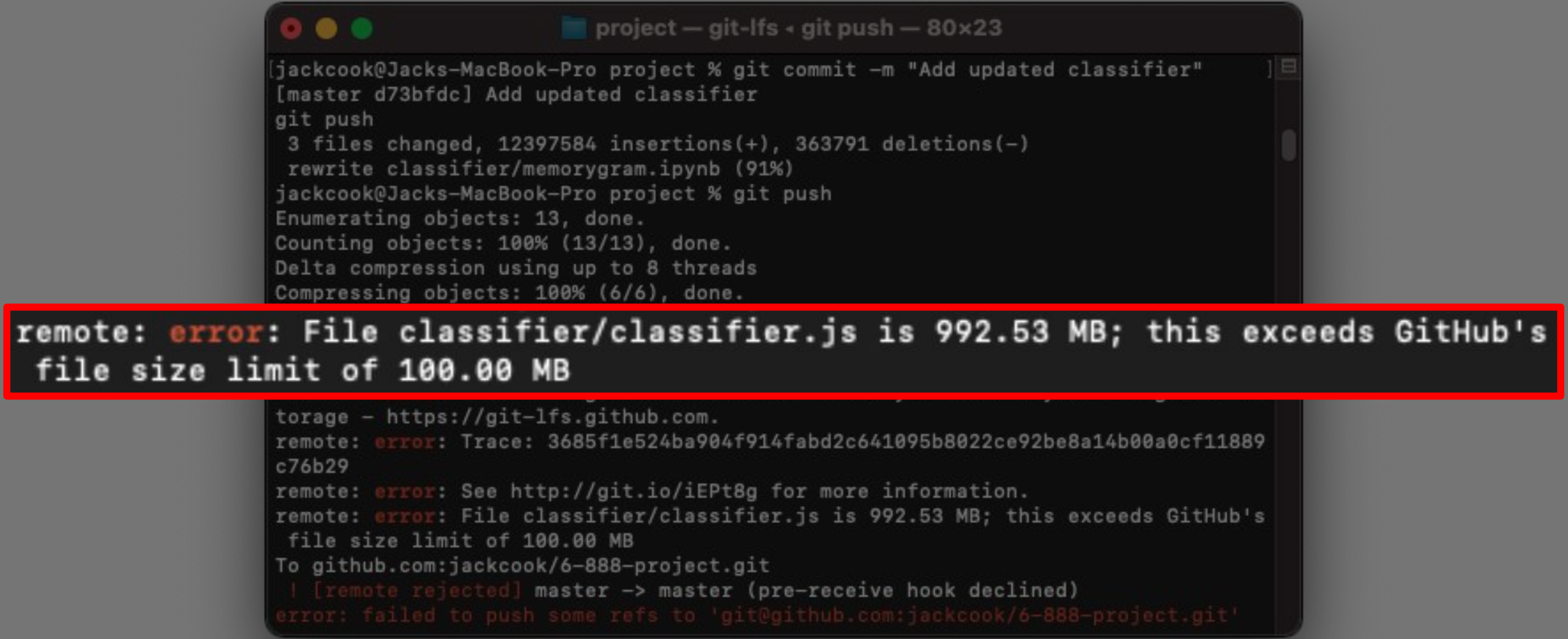
Progress!

- October 13: 87% accuracy between 4 websites
  - First working demo! Cache-based traces and a TensorFlow.js model
- October 28: 100% accuracy between 4 websites, 88% accuracy between 10 websites
  - Tweaked trace collection procedure, switched to random forest models
- November 10: 90% accuracy between 10 websites
  - Switched from cache-based traces to recording operations per second
- December 3: 94% accuracy between 10 websites
  - Fixed a bug with my selenium script, switched to extra trees classifier
- December 6: 97% accuracy between 10 websites
  - Compiled trace collection code to WebAssembly
- December 9: 74% accuracy between 50 websites
- Future: Can probably do better?

Future Work

- Investigate browsers other than Chrome
- Keep trying to find better ways to make traces
- Collect noisy data (e.g. while other applications are open) and see how much accuracy drops
- Distinguishing between websites opening and nothing happening at all (so that we don't have to hit the start button to record a trace)
- Investigate differences due to network latency
- Make the 50-way classifier smaller...


project — git-lfs ◂ git push — 80×23
jackcook@Jacks-MacBook-Pro project % git commit -m "Add updated classifier"
[master d73bfdc] Add updated classifier
git push
3 files changed, 12397584 insertions(+), 363791 deletions(-)
rewrite classifier/memorygram.ipynb (91%)
jackcook@Jacks-MacBook-Pro project % git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 27.58 MiB | 3.86 MiB/s, done.
Total 7 (delta 3), reused 1 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote: error: GH001: Large files detected. You may want to try Git Large File S
torage - https://git-lfs.github.com.
remote: error: Trace: 3685f1e524ba904f914fabd2c641095b8022ce92be8a14b00a0cf11889
c76b29
remote: error: See http://git.io/iEPt8g for more information.
remote: error: File classifier/classifier.js is 992.53 MB; this exceeds GitHub's
file size limit of 100.00 MB
To github.com:jackcook/6-888-project.git
! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'git@github.com:jackcook/6-888-project.git'


project — git-lfs ◂ git push — 80×23
jackcook@Jacks-MacBook-Pro project % git commit -m "Add updated classifier"
[master d73bfdc] Add updated classifier
git push
3 files changed, 12397584 insertions(+), 363791 deletions(-)
rewrite classifier/memorygram.ipynb (91%)
jackcook@Jacks-MacBook-Pro project % git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
remote: error: File classifier/classifier.js is 992.53 MB; this exceeds GitHub's file size limit of 100.00 MB
torage – https://git-lfs.github.com.
remote: error: Trace: 3685f1e524ba904f914fabd2c641095b8022ce92be8a14b00a0cf11889c76b29
remote: error: See http://git.io/iEPt8g for more information.
remote: error: File classifier/classifier.js is 992.53 MB; this exceeds GitHub's file size limit of 100.00 MB
To github.com:jackcook/6-888-project.git
! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'git@github.com:jackcook/6-888-project.git'

Questions?