An analysis of censorship in Chinese open source projects

Jeffrey Knockel, Masashi Crete-Nishihata, and Lotus Ruan

Nature of Chinese censorship

- “Decentralized”, “fragmented”
- “Anaconda in the Chandelier” (Link)
- “Intermediary liability”, “self-discipline” (MacKinnon)
- “Atomization and personalization of censorship” (Bandurski)

Previous work

- Chat apps (Knockel et al. 2011, Crandall et al. 2013, Hardy 2013, Ruan et al. 2016)
- Live streaming services (Knockel et al. 2015, Crete-Nishihata et al. 2016)
- Blogs (MacKinnon 2010)
- Microblogs (Bamman et al. 2012, Miller 2017)
- Search engines (Villeneuve 2006)
- Online games (Knockel et al. 2017)

Our work

- Previous work: *companies*
- Our work: *individuals*
- How far down does pressure to censor go?
- > 1,000 blacklists
- > 200,000 unique keywords

Open source projects

- Open source obviates reverse engineering
- Look for censorship blacklists in Github projects
- GitHub has 24 million users, 67 million repos
- One third of Chinese developers use it

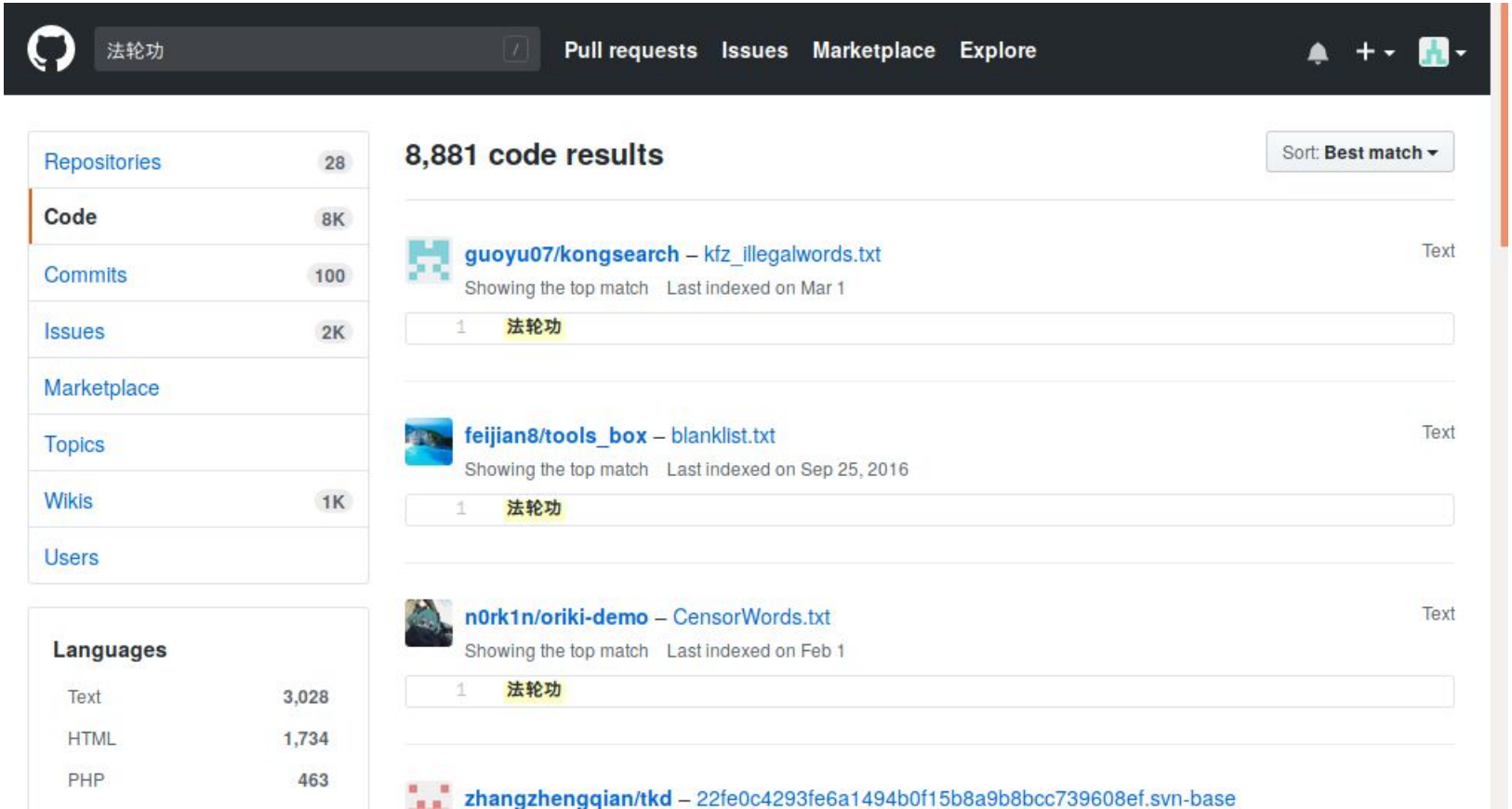
GitHub and China

- 2013: MITM attack
  - Fake SSL cert
  - Harvest passwords
- 2015: DDoS
  - “Great Cannon”
  - Injected javascript attack code
- China has no domestic alternative

How do we find blacklists?

1. Download files containing sensitive words
2. Extract lists of strings
3. Classify whether list is Chinese blacklist

Step 1: Downloading files



Ethical considerations

- No full-site code search in GitHub API (computationally expensive)
- Scraping search forbidden by `robots.txt`
- Received approval from GitHub support

Constructing search queries

- Files must contain one of 21,934 keywords previously found in blacklists
- File names must contain one of these strings:

badword, banned, blacklist, censor, dirty, filter, forbid, forbidden, illegal, keyword, profanity, sensitive

Navigation

Project description

Release history

Download files

Project links

Homepage

Statistics

GitHub statistics:

Stars: 994

Forks: 139

Open issues/PRs: 42

Project description

Chardet: The Universal Character Encoding Detector

Detects

- ASCII, UTF-8, UTF-16 (2 variants), UTF-32 (4 variants)
- Big5, GB2312, EUC-TW, HZ-GB-2312, ISO-2022-CN (Traditional and Simplified Chinese)
- EUC-JP, SHIFT_JIS, CP932, ISO-2022-JP (Japanese)
- EUC-KR, ISO-2022-KR (Korean)
- KOI8-R, MacCyrillic, IBM855, IBM866, ISO-8859-5, windows-1251 (Cyrillic)
- ISO-8859-5, windows-1251 (Bulgarian)
- ISO-8859-1, windows-1252 (Western European languages)
- ISO-8859-7, windows-1253 (Greek)
- ISO-8859-8, windows-1255 (Visual and Logical Hebrew)
- TIS-620 (Thai)

Note

Step 2: Extracting lists

- Built a list extractor
- Extracts lists from multiple structured data formats
- Used files containing “法轮” to determine which formats
- (From previous research, very likely to be on lists)
- Wanted extractor generalized, not overfitted
- < 350 lines of python

Formats

- XML
- JSON
- CSV
- C-like code
- Symbol-delimited plain text
- Newline-delimited plain text

XML

<keywords>法轮功 | june4 | fuck</keywords>

XML

<list>
<keyword>法轮功</keyword>
<keyword>june4</keyword>
<keyword>fuck</keyword>
</list>

XML

```
<list>
<keyword text="法轮功" />
<keyword text="june4" />
<keyword text="fuck" />
</list>
```

Flattening

All text values (for XML: text nodes and attribute values) put into list with path.

- (element, “list”) / (element, “keyword”) / (attribute “text”): “法轮功”
- (element, “list”) / (element, “keyword”) / (attribute “text”): “june4”
- (element, “list”) / (element, “keyword”) / (attribute “text”): “fuck”

XML

```
<list>
<description>Blacklist</description>
<keyword text="法轮功" action="delete" />
<keyword text="june4" action="delete" />
<keyword text="fuck" action="delete" />
</list>
```

Flattening

- (element, “list”) / (element, “description”): “Blacklist”
- (element, “list”) / (element, “keyword”) / (attribute “text”): “法轮功”
- (element, “list”) / (element, “keyword”) / (attribute “action”): “delete”
- (element, “list”) / (element, “keyword”) / (attribute “text”): “june4”
- (element, “list”) / (element, “keyword”) / (attribute “action”): “delete”
- (element, “list”) / (element, “keyword”) / (attribute “text”): “fuck”
- (element, “list”) / (element, “keyword”) / (attribute “action”): “delete”

Flattening

- (element, “list”) / (element, “description”): “Blacklist”
- (element, “list”) / (element, “keyword”) / (attribute “text”): “法轮功”
- (element, “list”) / (element, “keyword”) / (attribute “action”): “delete”
- (element, “list”) / (element, “keyword”) / (attribute “text”): “june4”
- (element, “list”) / (element, “keyword”) / (attribute “action”): “delete”
- (element, “list”) / (element, “keyword”) / (attribute “text”): “fuck”
- (element, “list”) / (element, “keyword”) / (attribute “action”): “delete”

1. [“Blacklist”]
2. [“法轮功”, “june4”, “fuck”]
3. [“delete”, “delete”, “delete”]

Requirements for all lists

- At least 20 elements
- At least one Chinese character

JSON


```
{
    list: "法轮功 | june4 | fuck",
}
```

JSON

```
{
    list: ["法轮功", "june4", "fuck"],
}
```

JSON

```
{
    list: {
        "法轮功": "***",
        "june4": "*****",
        "fuck": "****",
    }
}
```

JSON

```
{
    list: [
        {text: "法轮功", …},
        {text: "june4", …},
        {text: "fuck", …},
    ]
}
```

JSON

```
{
    list: {
        {"法轮功": "***", …},
        {"june4": "*****", …},
        {"fuck": "****", …},
    }
}
```

CSV

keyword, substitution
法轮功, ***
june4, *****
fuck, ****

C-like code

```
var blacklist = "法轮功 | june4 | fuck";
```

C-like code

```
var blacklist = ["法轮功", "june4", "fuck"];
```

C-like code

Arrays / lists / tuples

- {…, …, …}
- […, …, …]
- (…, …, …)

C-like code

Strings

- "…\"…\"…"
- '…\'…\'…'

Symbol-delimited

法轮功 | june4 | fuck

Newline-delimited

法轮功

june4

fuck

- Constraints:
- No line indented 3 or more spaces
- Average length ≤ 15

Step 3: Classifying lists

1. Naive approach
2. Machine learning approach

Naive approach

A list is a Chinese blacklist iff it contains any string containing the substring “法轮”

Evaluate the approach by manually verifying each positively classified list

Machine learning approach

- One-class support vector machine (SVM)
- Training only requires positive labels (blacklists)
- No representative sampling of all lists of strings on GitHub that are not Chinese blacklists

Machine learning approach

1. Convert any traditional characters to simplified
2. Split string into Chinese and English words

历史de伤口 → “历史” “de” “伤口”

3. Each list is a vector of the number of counts of each word

Machine learning approach

- Used lists from previous research (chat, live streaming, etc.)
- Singular value decomposition to *d* dimensions
- One-class SVM → no cross-validation :(
- To evaluate *d*, use Chinese blacklists from naive approach as positive labels
- Pick best *d*, ignoring degenerate cases
- Manually verify Chinese blacklists

Results

- Downloaded 648,323 files
- Extracted 45,986 lists with at least 20 elements and one Chinese character

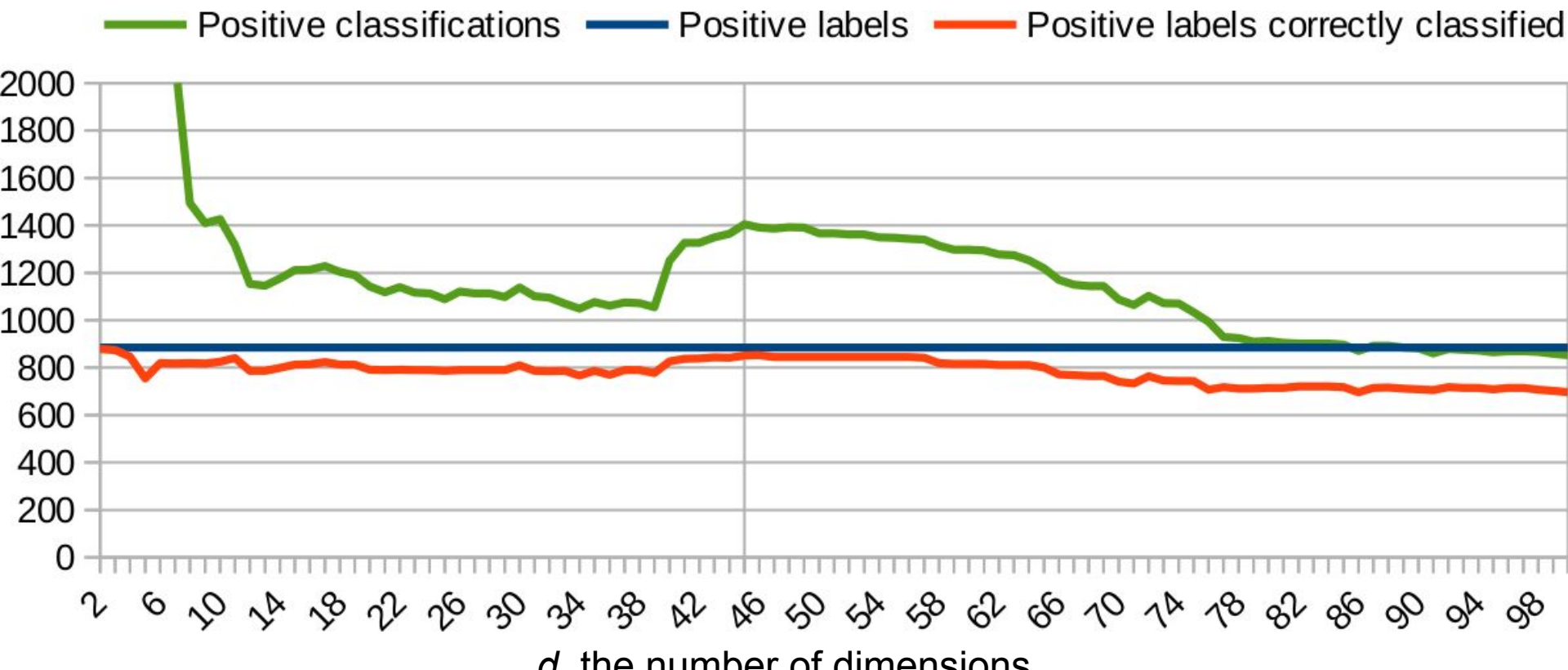
Naive approach

- 915 potential Chinese blacklists
- 838 true Chinese blacklists
- 884 after inspecting files for more than one list

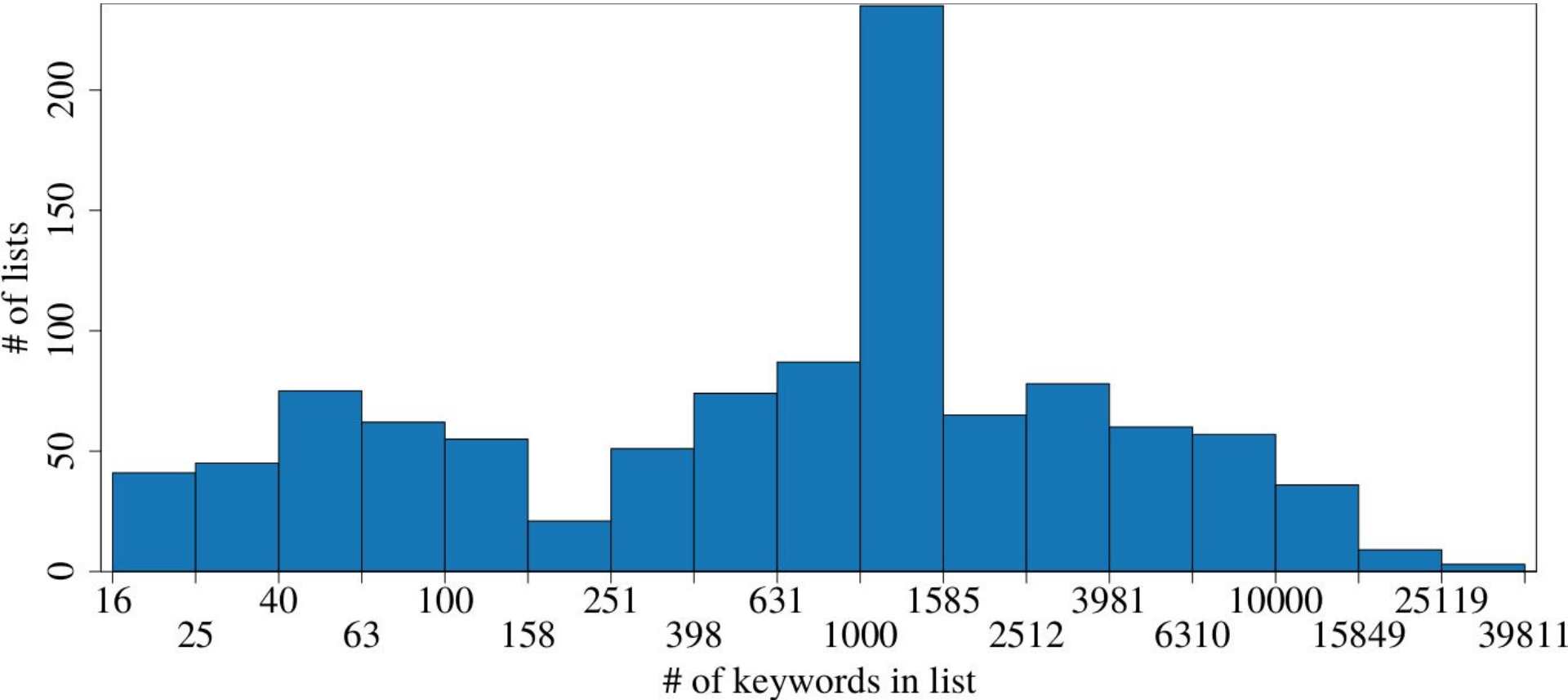
Machine learning approach

- Positively classified 1,391 lists
- 1,054 true Chinese blacklists
- 215,007 unique keywords

Machine learning approach



Blacklist lengths



Keyword characterization

Top 1–10			Top 11–20		
n	Keyword	Translation	n	Keyword	Translation
703	鸡巴	dick	621	台独	Taiwanese independence
689	法轮	Falun	620	阴唇	labia
665	李洪志	Li Hongzhi	618	真善忍	truthfulness, tolerance
640	阴道	vagina	616	疆独	Xinjiang independence
638	阴茎	penis	616	做爱	making love
635	藏独	Tibetan independence	611	口交	blowjob
633	龟头	glans	604	法轮功	Falun Gong
629	淫水	kinky	597	性交	sex
626	肛交	anal sex	596	共匪	CCP bandit
622	小穴	small hole	593	江泽民	Jiang Zemin

- So now we have this giant, diverse dataset
- We’ve aggregated it together
- We see some the popular topics
- We must have finally gotten to the bottom of what the Chinese government wants to censor

Keyword characterization

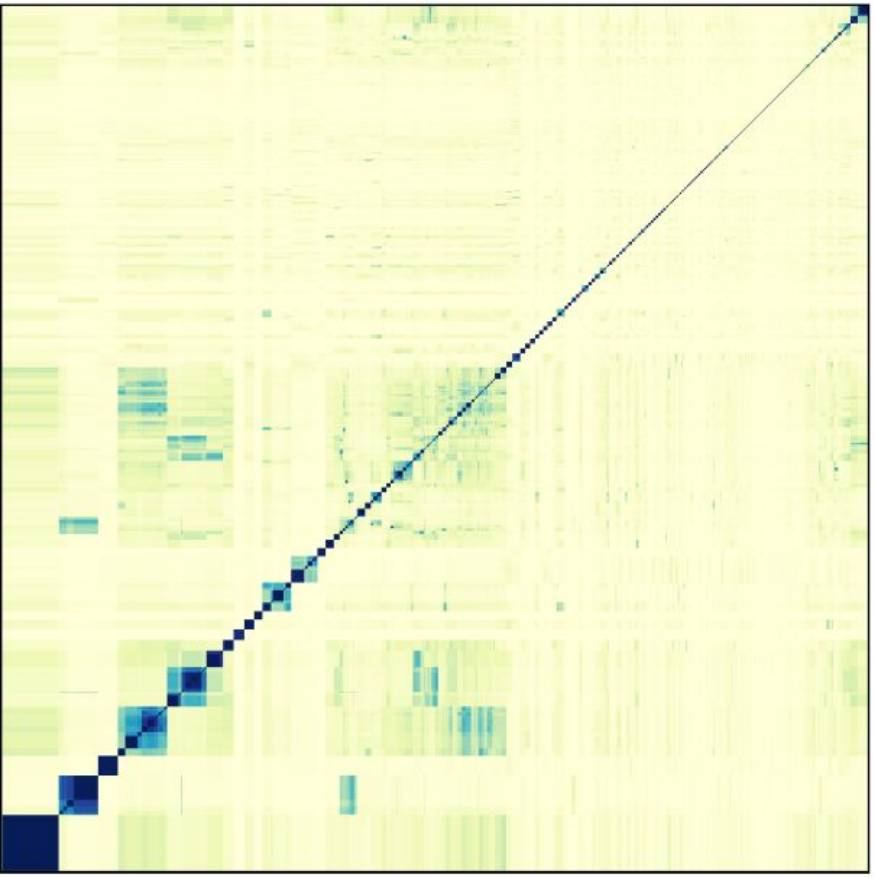
This is how the Chinese government censors sex, Falun Gong, and independence movements.

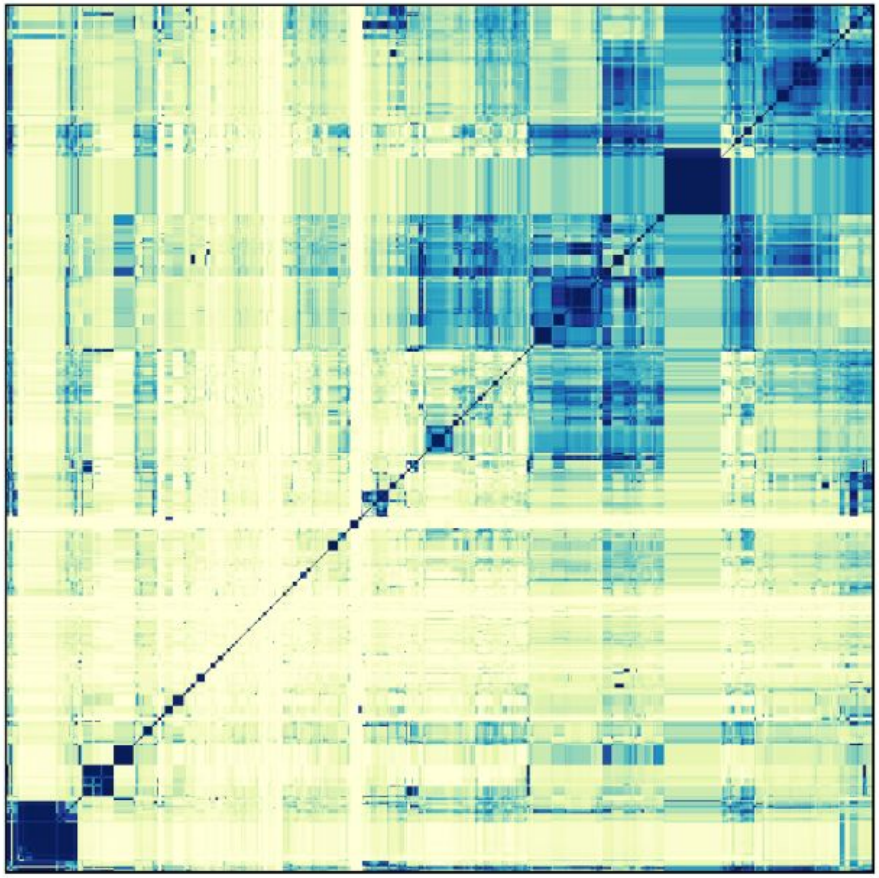
Keyword characterization

~~This is how the Chinese government censors sex, Falun Gong, and independence movements.~~

Let’s take a closer look at this data.

Keyword characterization





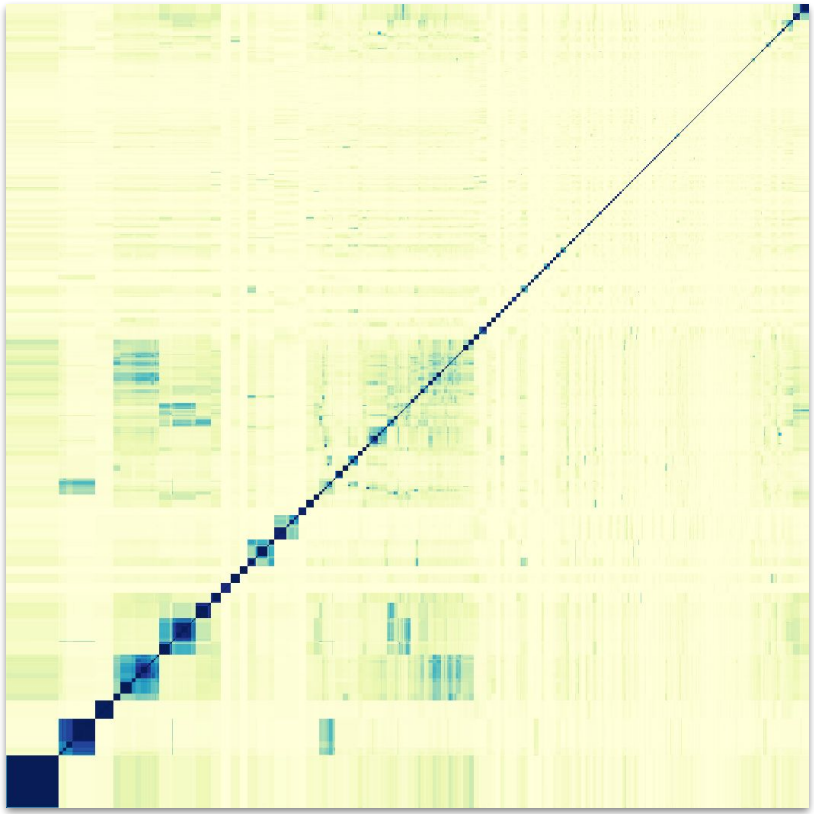
Keyword characterization

~~This is how the Chinese government censors sex, Falun Gong, and independence movements.~~

- Aggregating data covers a multitude of sins
- Implementation is not monolithic
- Observing developers, not Chinese government

Little overlap between lists

- Consistent with previous research of *companies*
- Where did so many disparate lists come from?
- Over half of lists have over one thousand keywords
- Developers individually curating them?



Conclusion

- You can find blacklists in open source code
- We found > 1,000 Chinese blacklists
- > 200,000 unique keywords
- Largest dataset to date

Future work

- Where do all of these lists come from?
- Why do developers include blacklists?
- What kinds of projects are using these lists?
- What proportion of Chinese open source projects use blacklists?
- Can GitHub metadata (commits, followers, forks, etc.) tell us more about how lists are updated or shared?

- Some can be addressed by our data
- Some can be answered by designing an interview

Questions?