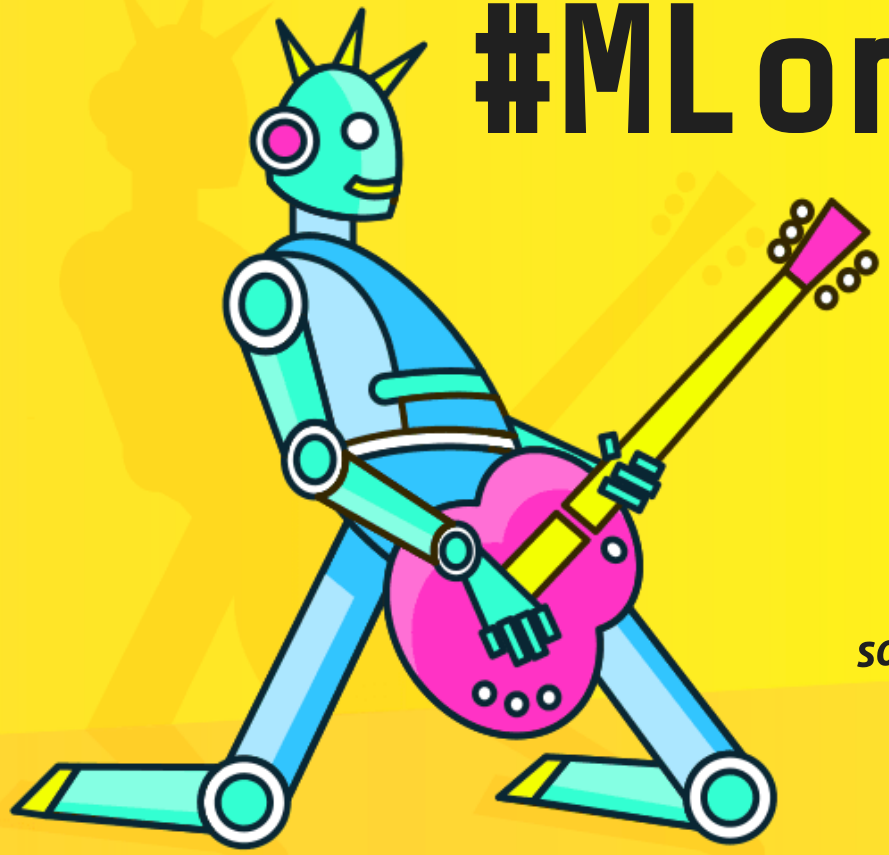
Machine Learning for Code

Egor Bulychev

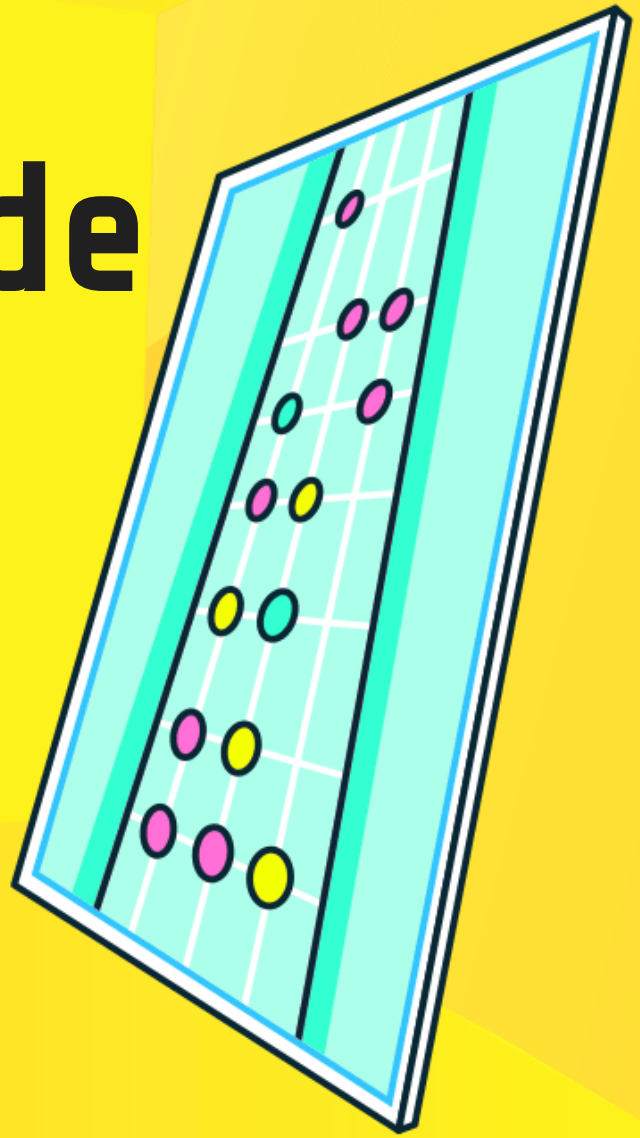
UseData Conf 2019

Профессиональная конференция для специалистов по машинному обучению и анализу данных

#MLonCode

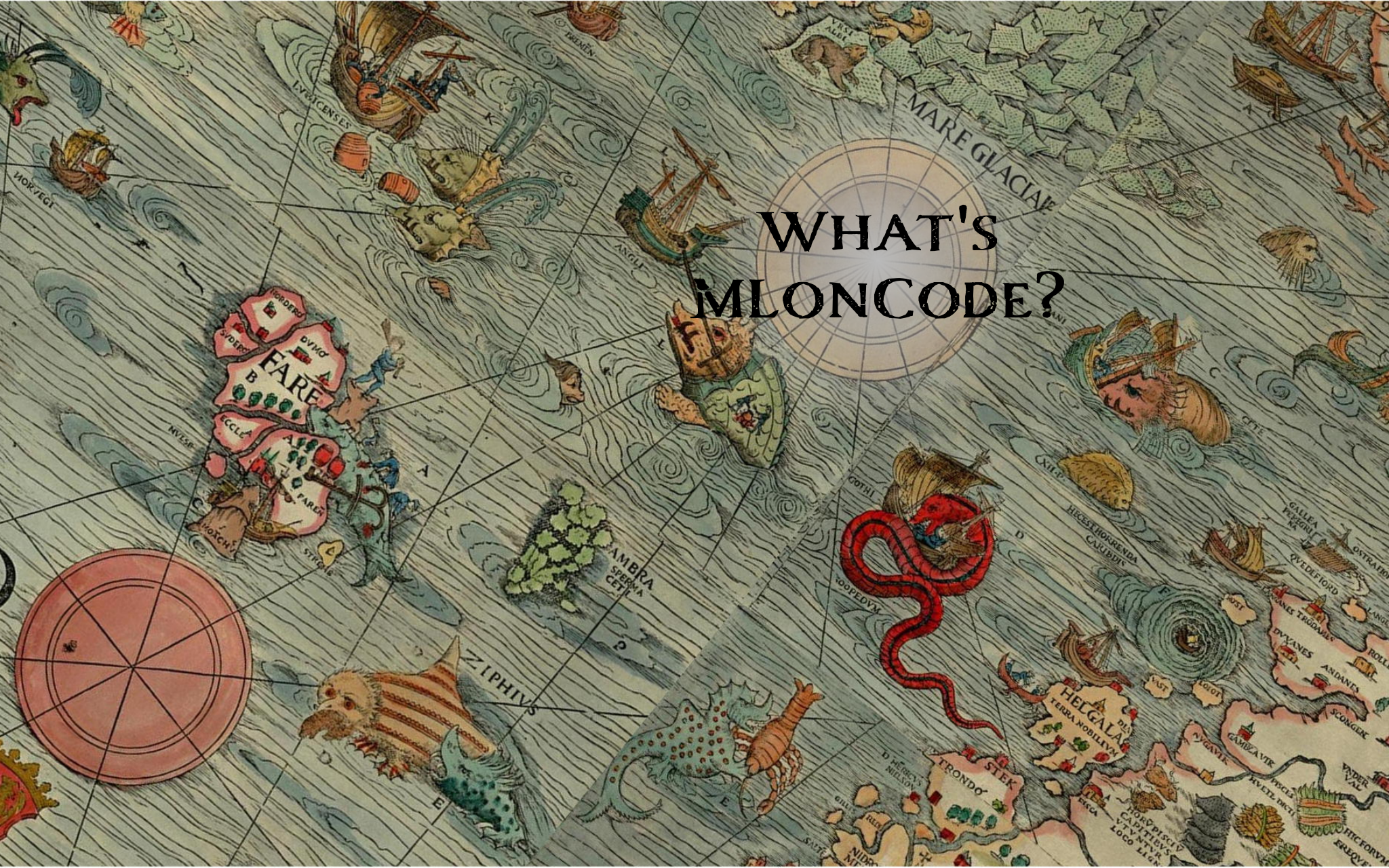


source{d}



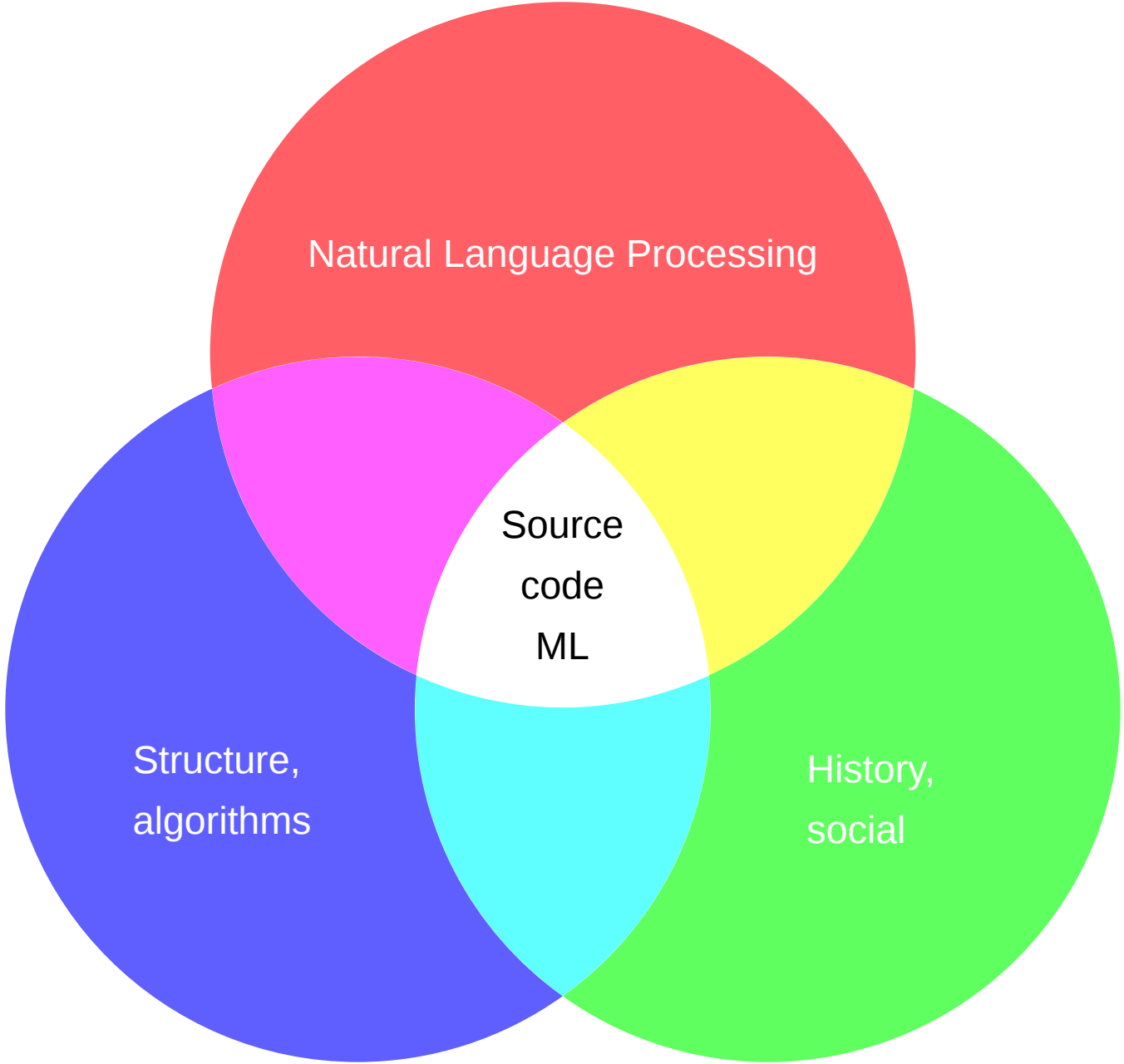
Plan

- What's MLonCode?
- Why MLonCode hard?
- Basics
- Similar repository search
- Code review
- Developer similarity


What's
MLonCode?
MARE GLACIALE
FARE
HELGALA
TERRA NOBILIVM
TRONDO
SIEK
ZIPHIVS
AMBRA
SPERMA
CETI
GOTHI
ANGLI
NORVEGI

Stereotypes

- Program Synthesis
- Program Translation
- Code Suggestion and Completion
- Program Repair and Bug Detection


Natural Language Processing
Source
code
ML
Structure,
algorithms
History,
social


Why MLonCode
hard?

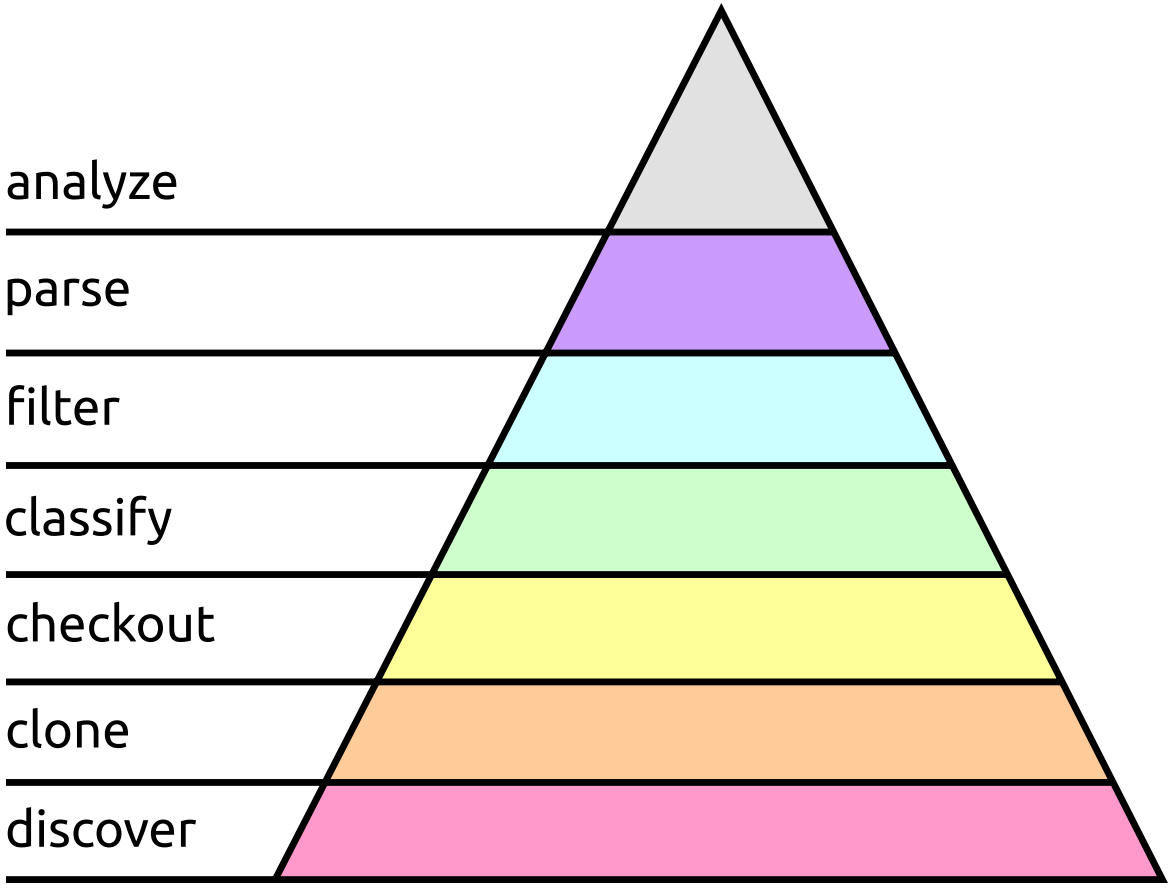

Data

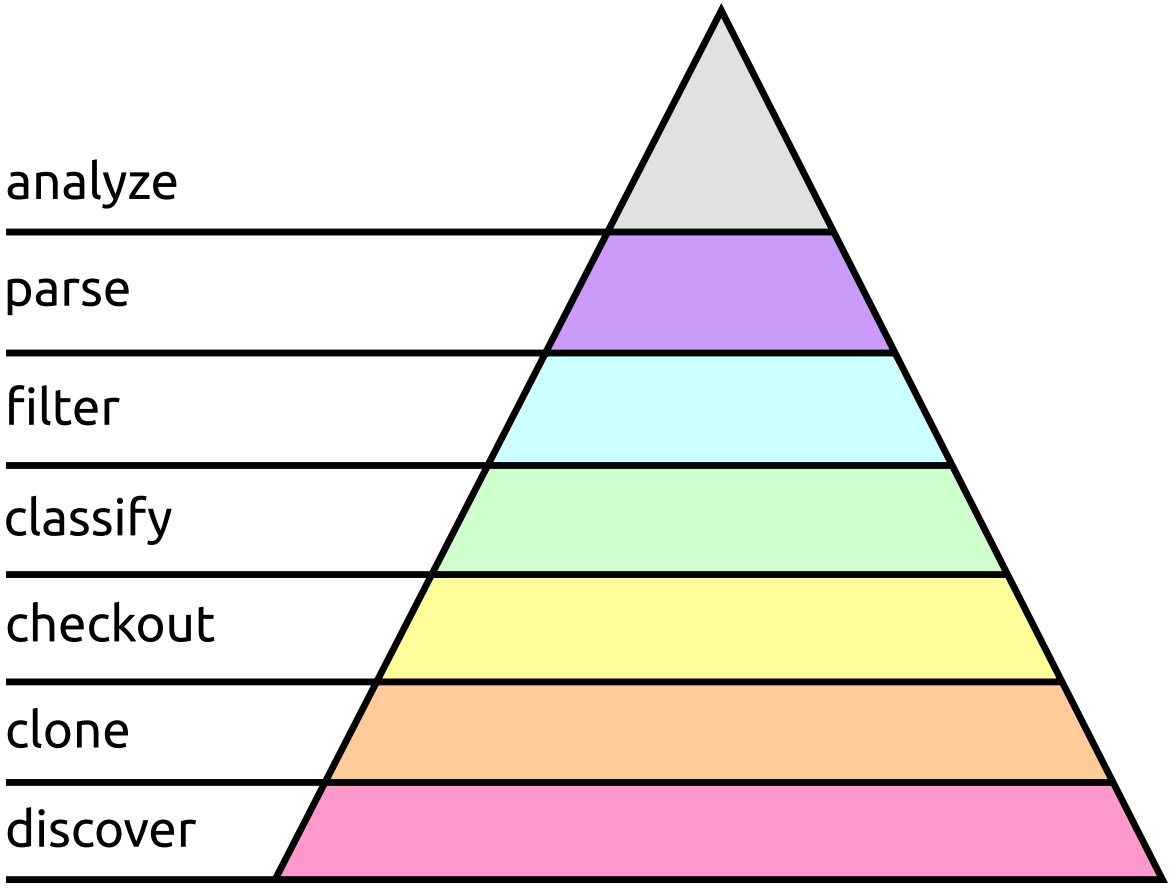
Common datasets

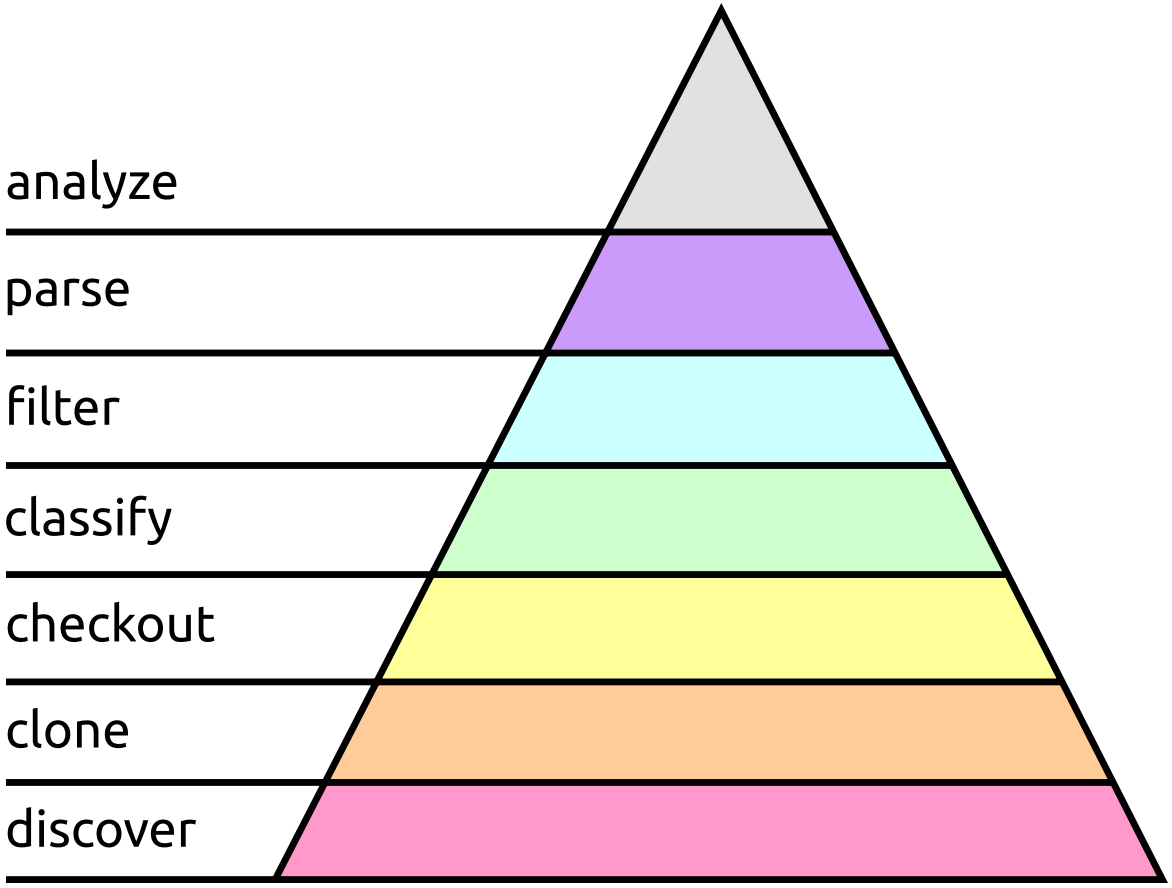
-
-
-

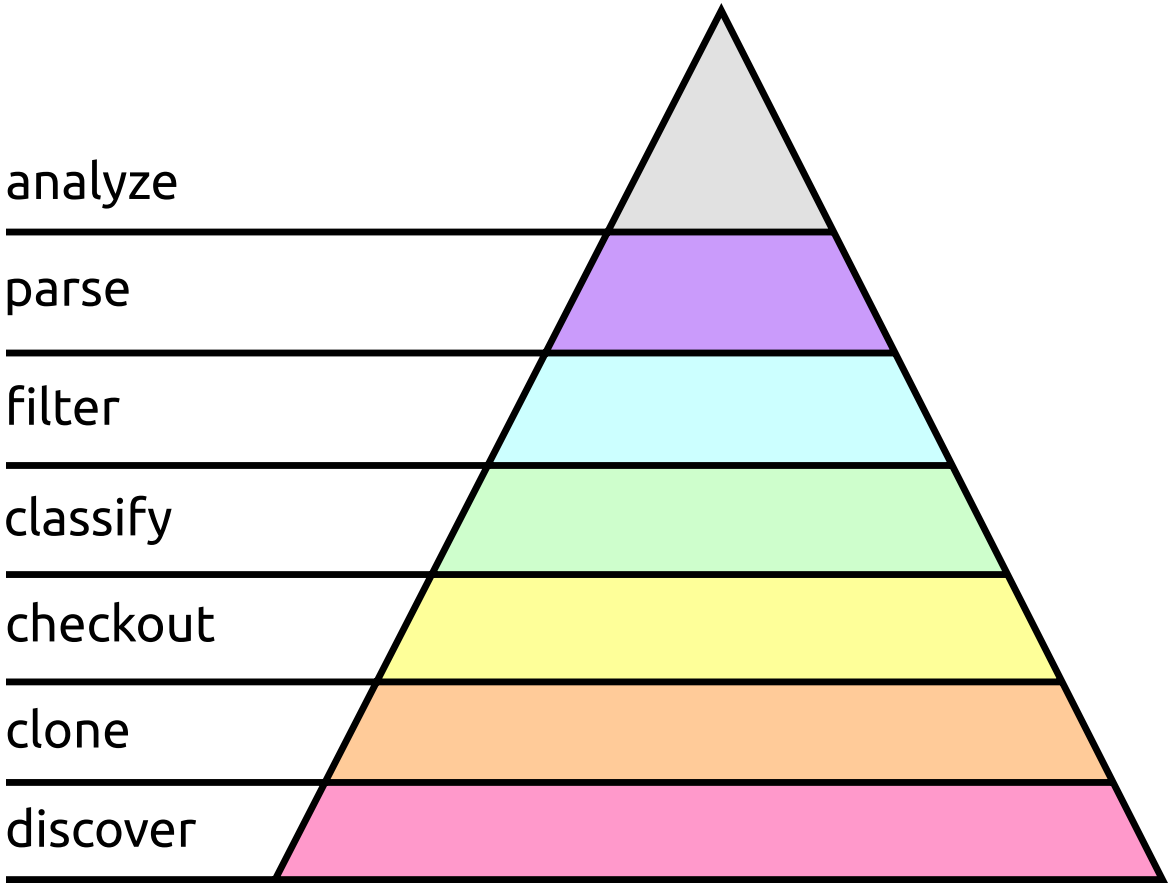
- Size: 6 TB
- Number of repositories: 260k+ (out of 96 M+) from GitHub
- Number of files: 136M+ files
- LOC: ~28 billions

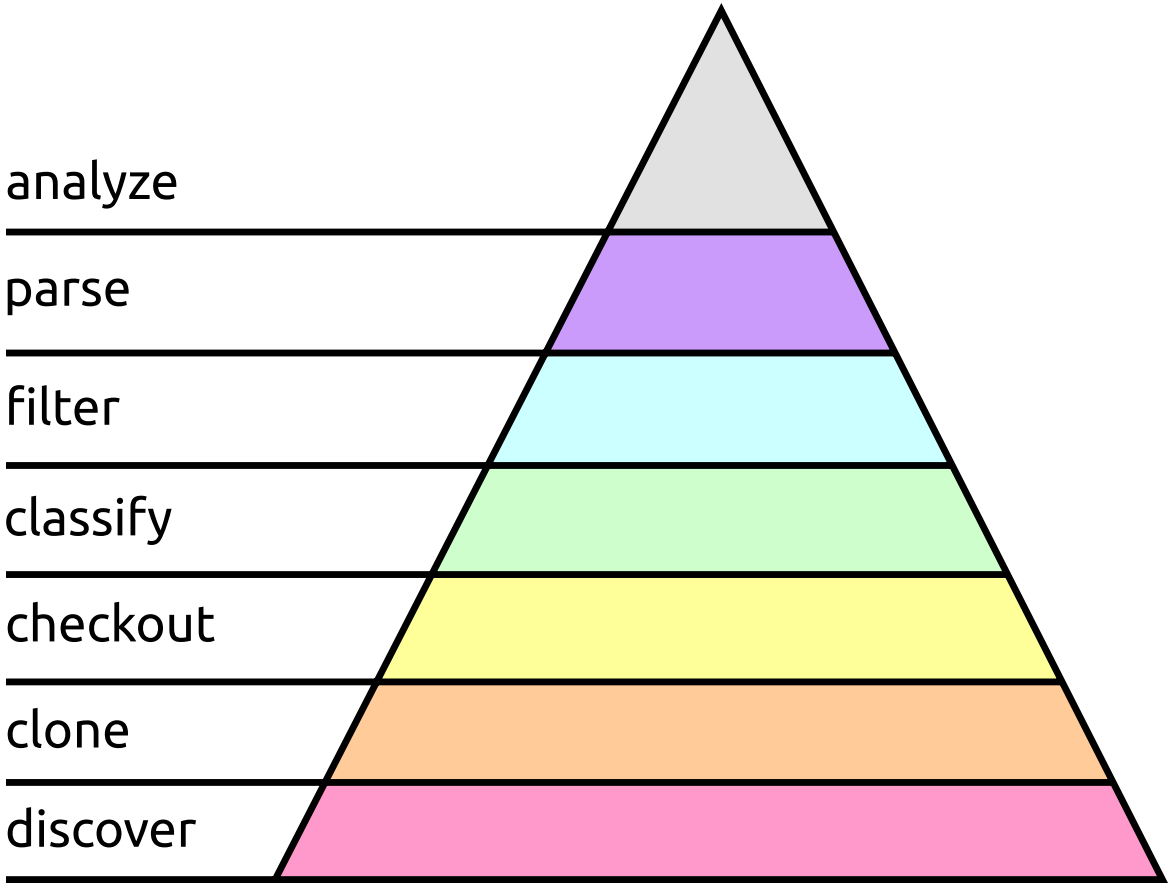
Tools

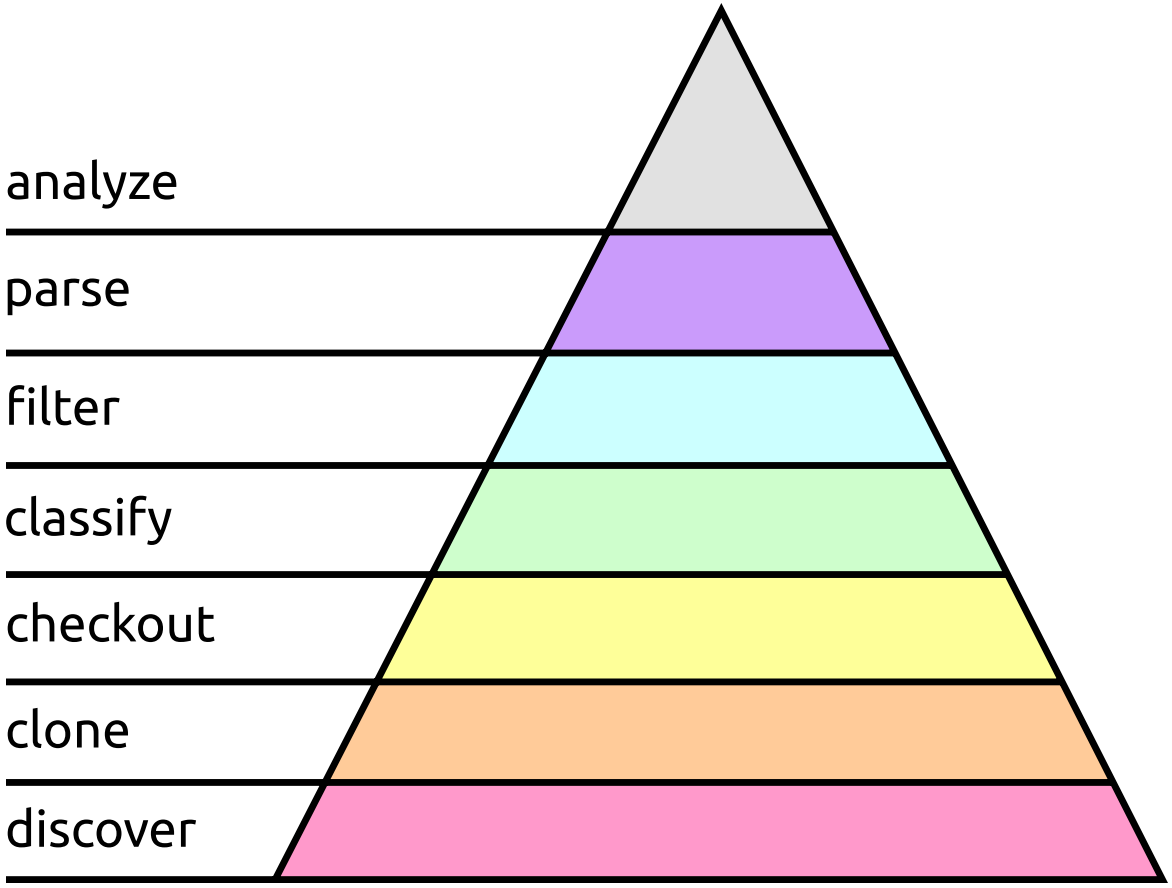

analyze
parse
filter
classify
checkout
clone
discover

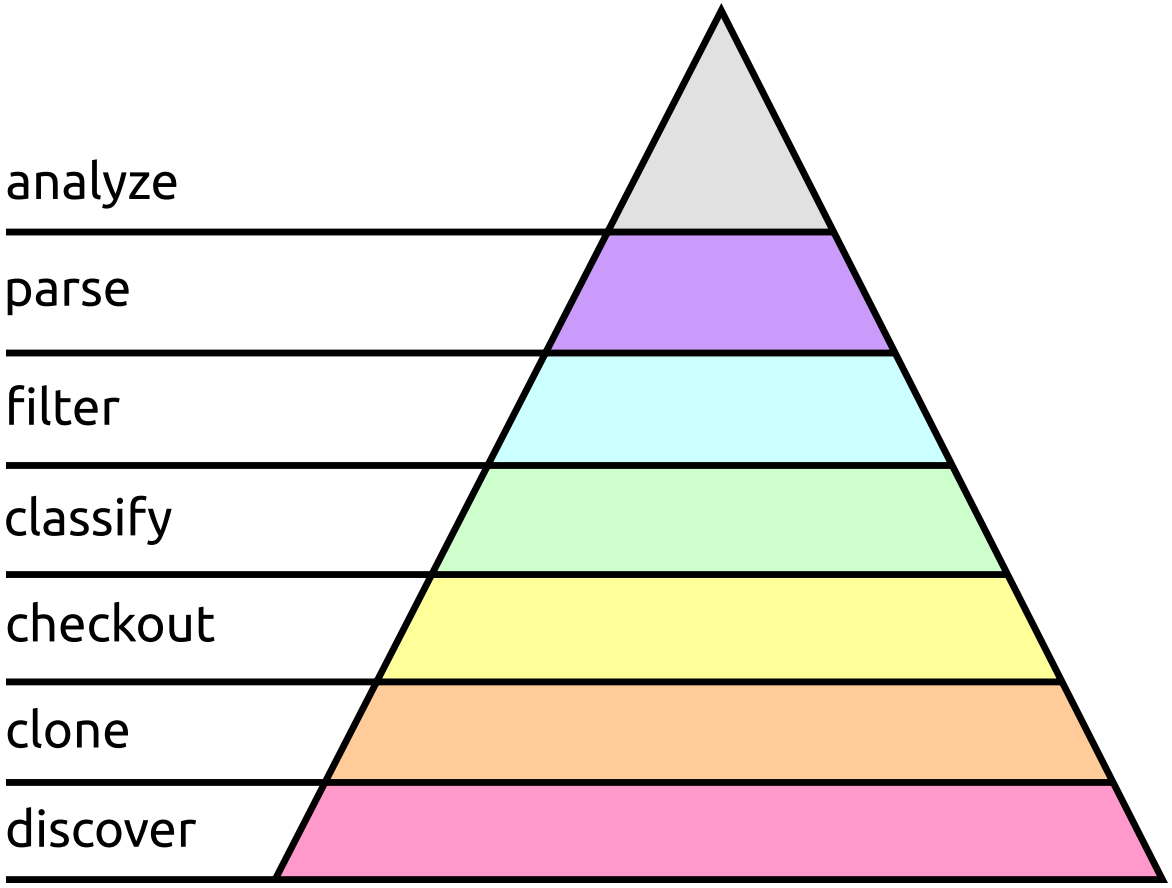

analyze
parse
filter
classify
checkout
clone
discover


analyze
parse
filter
classify
checkout
clone
discover


analyze
parse
filter
classify
checkout
clone
discover


analyze
parse
filter
classify
checkout
clone
discover


analyze
parse
filter
classify
checkout
clone
discover


analyze
parse
filter
classify
checkout
clone
discover

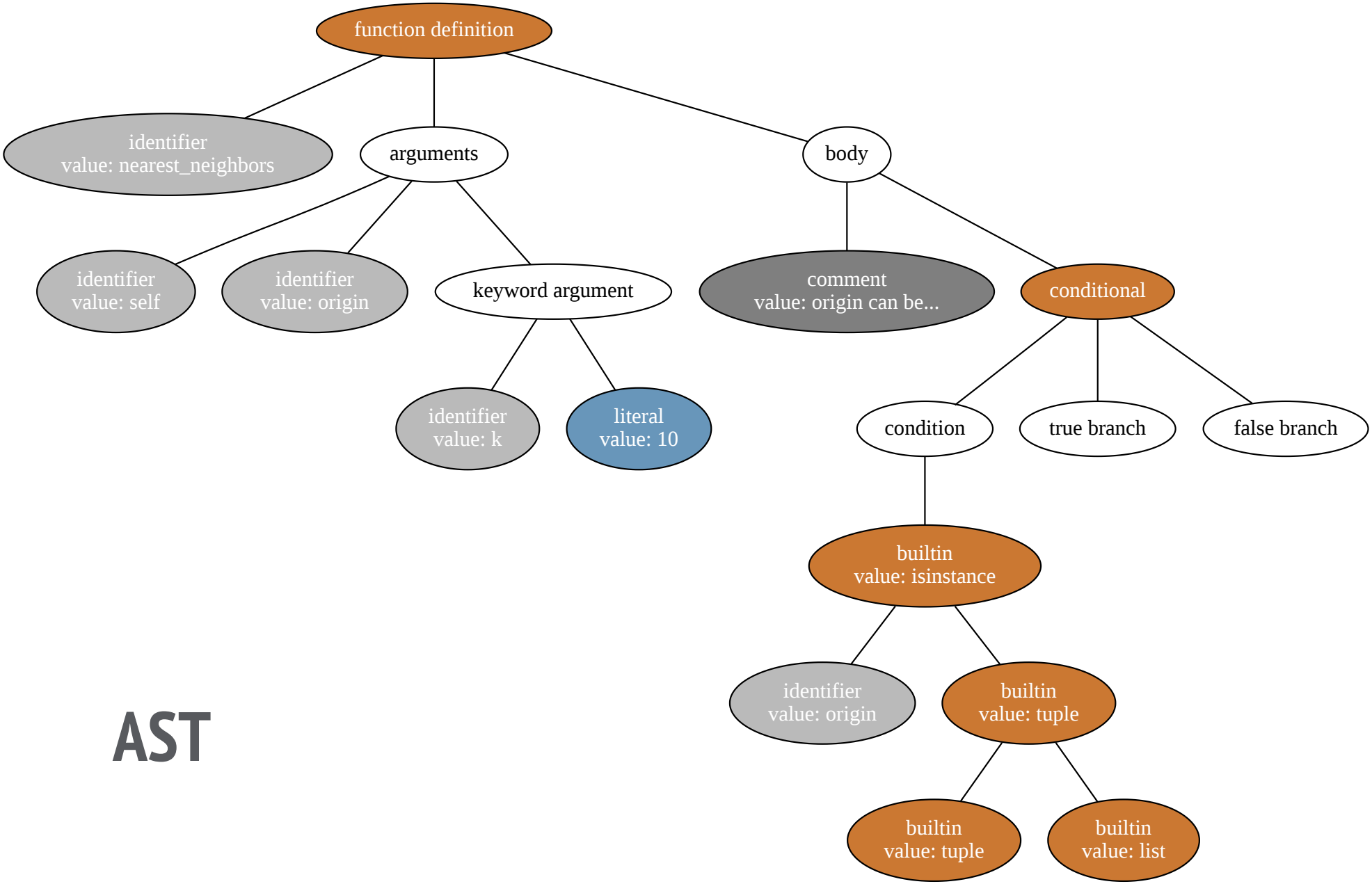
Parsing

```
def nearest_neighbors(self, origin, k=10,
                      skipped_stop=0.99, throw=True):
    # origin can be either a text query or an id
    if isinstance(origin, (tuple, list):
         words, weights = origin
         weights = numpy.array(weights, dtype=numpy.float32)
         index = None
         avg = self._get_centroid(words, weights, force=True)
    else:
         index = origin
         words, weights = self._get_vocabulary(index)
         avg = self._get_centroid_by_index(index)
    if avg is None:
         raise ValueError(
             "Too little vocabulary for %s: %d" % (index, len(words)))
```

Parsing

```
01. def nearest_neighbors(self, origin, k=10,
02.                       skipped_stop=0.99, throw=True):
03.     # origin can be either a text query or an id
04.     if isinstance(origin, (tuple, list):
05.         words, weights = origin
06.         weights = numpy.array(weights, dtype=numpy.float32)
07.         index = None
08.         avg = self._get_centroid(words, weights, force=True)
09.     else:
10.         index = origin
11.         words, weights = self._get_vocabulary(index)
12.         avg = self._get_centroid_by_index(index)
13.     if avg is None:
14.         raise ValueError(
15.             "Too little vocabulary for %s: %d" % (index, len(words)))
```

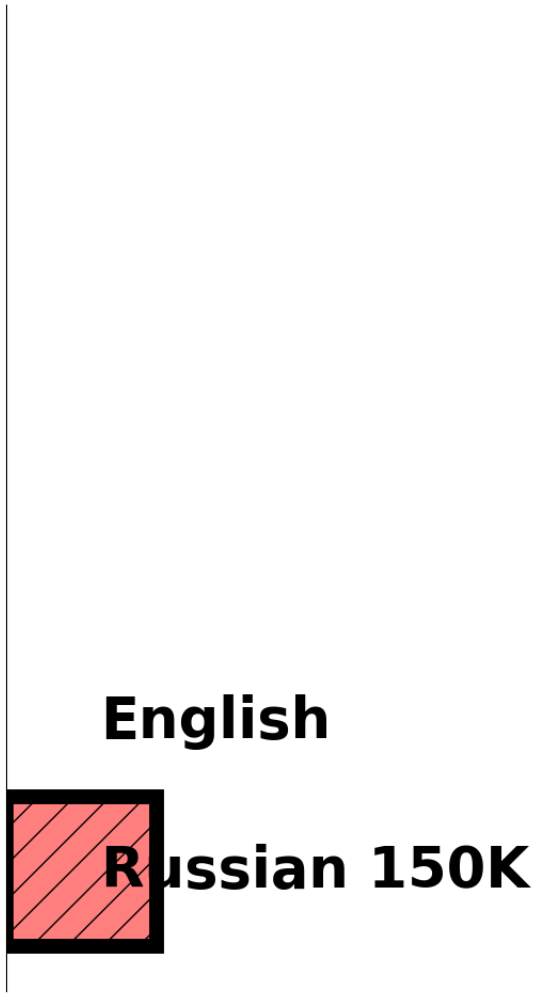
- keywords
- identifiers
- literals
- strings
- comments
- reserved


function definition
identifier
value: nearest_neighbors
arguments
body
identifier
value: self
identifier
value: origin
keyword argument
comment
value: origin can be...
conditional
identifier
value: k
literal
value: 10
condition
true branch
false branch
builtin
value: isinstance
identifier
value: origin
builtin
value: tuple
builtin
value: tuple
builtin
value: list
AST

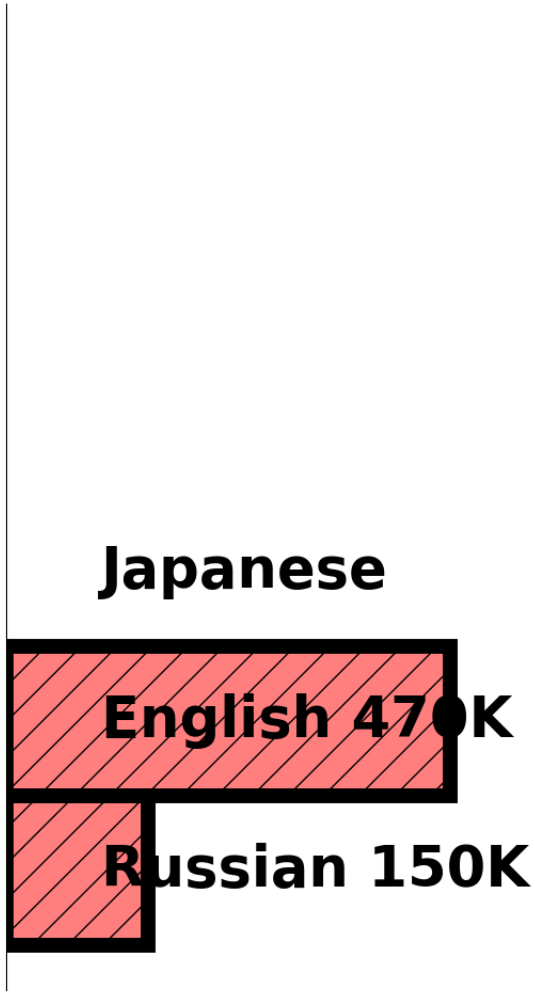
#words

Russian

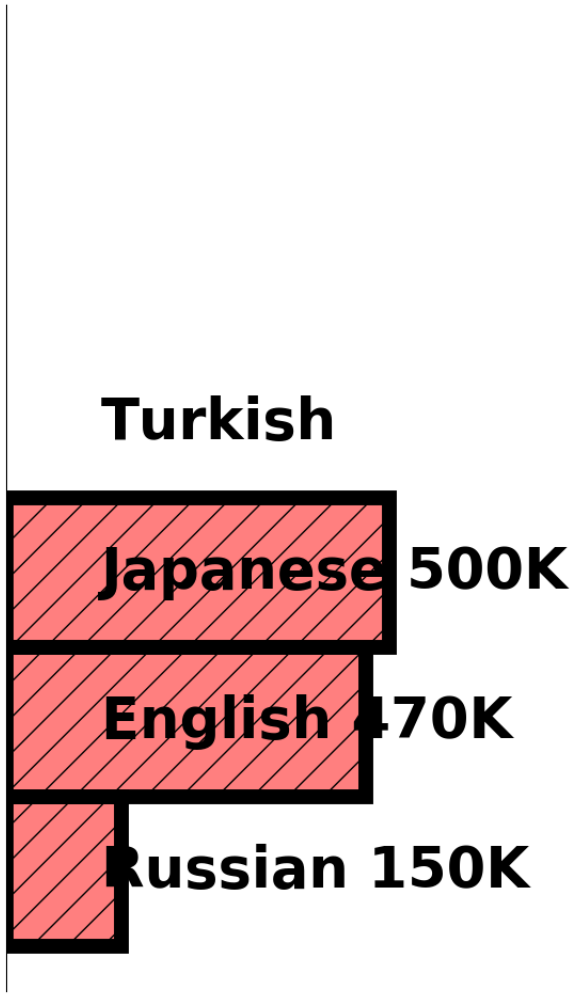
#words



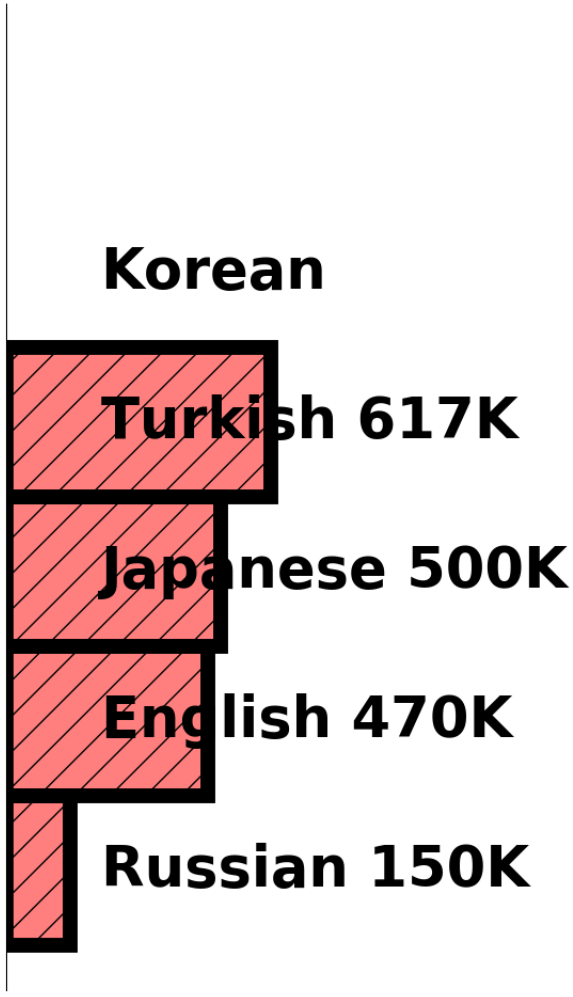
#words

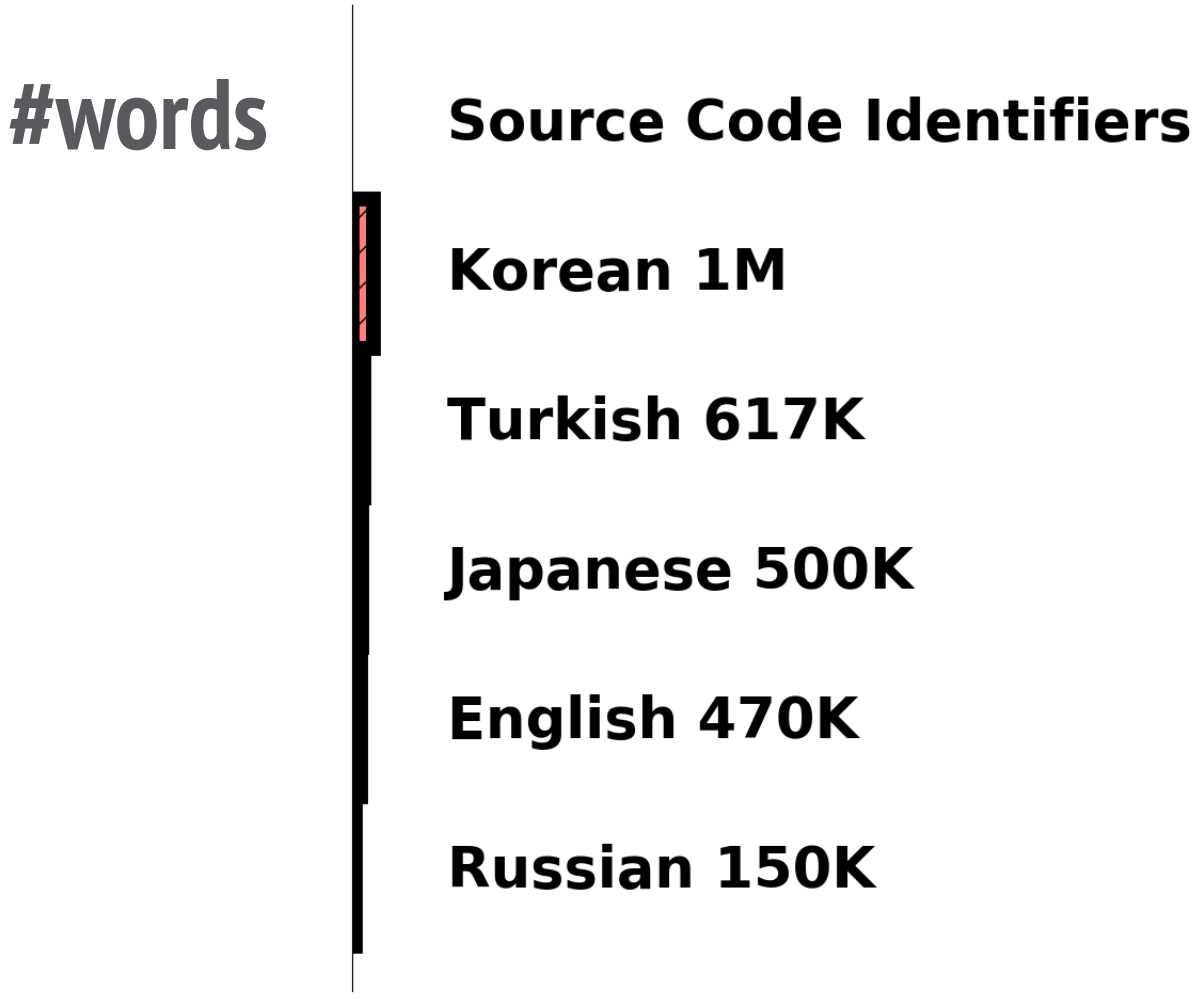


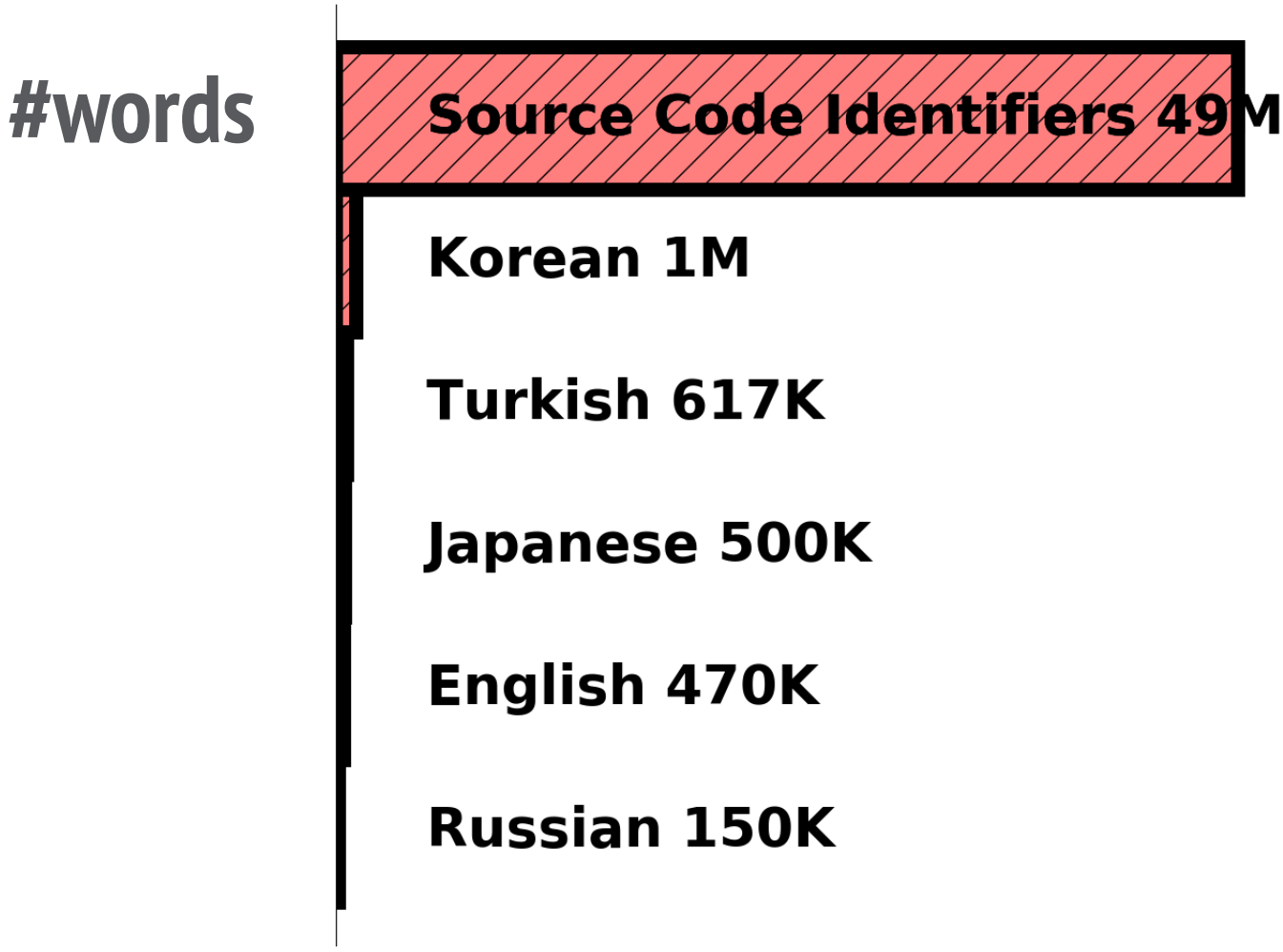
#words



#words


Korean
Turkish 617K
Japanese 500K
English 470K
Russian 150K


#words
Source Code Identifiers
Korean 1M
Turkish 617K
Japanese 500K
English 470K
Russian 150K


#words
Source Code Identifiers 49M
Korean 1M
Turkish 617K
Japanese 500K
English 470K
Russian 150K

Why so many identifiers?

```
01. _tcp_socket_connect -> [tcp, socket, connect]
02. AuthenticationError -> [authentication, error]
03. set_visible -> [set, visible]
```


Code naturalness

```
01. class ???:
02.     def connect(self, dbname, user, password, host, port):
03.         # ...
04.     def query(self, sql):
05.         # ...
06.     def close(self):
07.         # ...
```

```
01. class Database:
02.     def connect(self, dbname, user, password, host, port):
03.         # ...
04.     def query(self, sql):
05.         # ...
06.     def close(self):
07.         # ...
```

Take away

- Identifier ~ combinations of words
- Raw code is text
- Parsed code is AST
- Non-linear structure of AST
- Code naturalness
- 370 programming languages

Basics

Identifier splitter

Approaches

- Heuristic
- Machine Learning

Heuristic - split by

- underscore `set_name -> [set, name]`
- changing of cases `SetName -> [set, name]`
- etc

Result: 49M -> 3M, but...

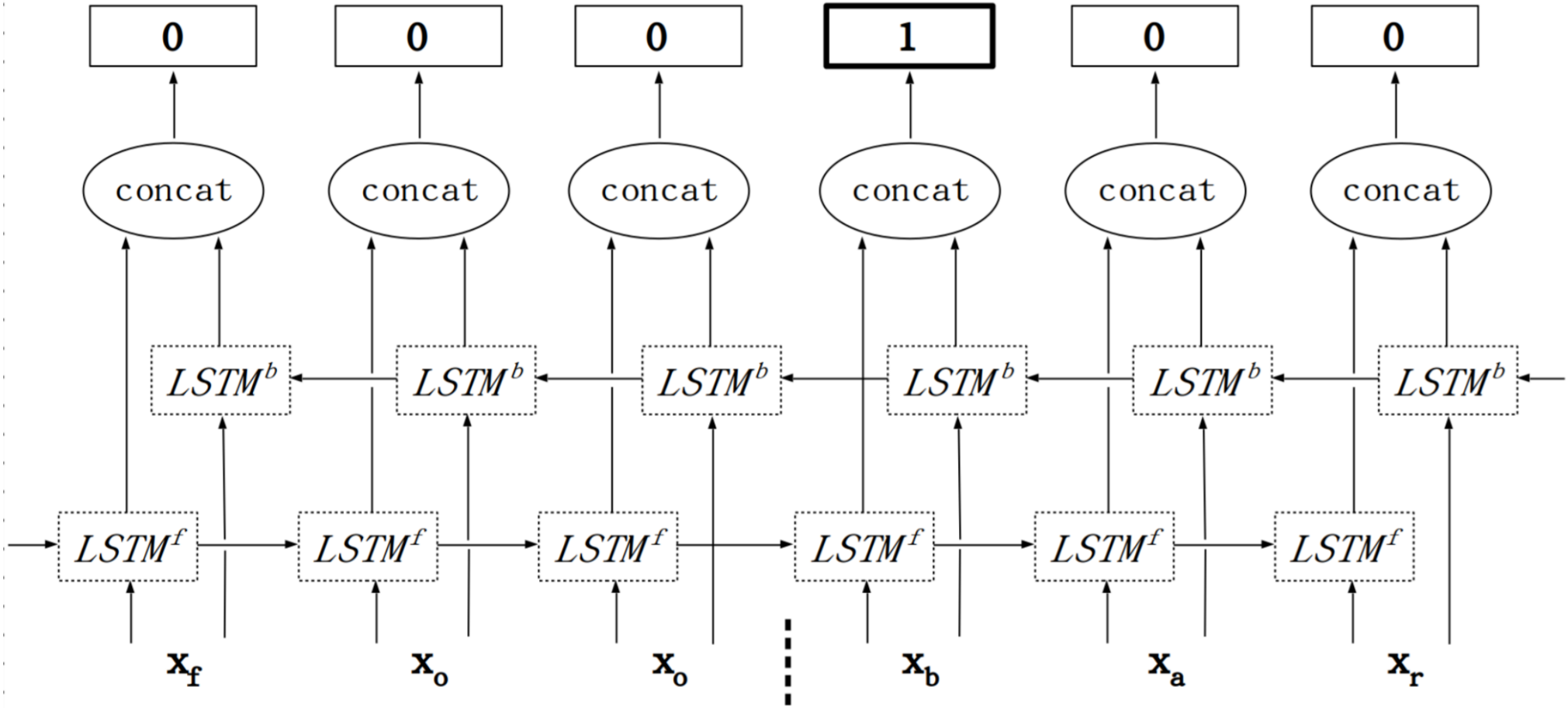
- `BUFFERFLAG_CODECCONFIG -> [bufferflag, codecconfig]`
- `metamodelength -> [metamodelength]`

- - 49M of identifiers
- Preprocessing
- Train NN

FooBar -> X, y

- f
- o
- o
- b
- a
- r

- 0
- 0
- 0
- 1
- 0
- 0


0
0
0
1
0
0
concat
concat
concat
concat
concat
concat
LSTM^b
LSTM^b
LSTM^b
LSTM^b
LSTM^b
LSTM^b
LSTM^f
LSTM^f
LSTM^f
LSTM^f
LSTM^f
LSTM^f
x_f
x_o
x_o
x_b
x_a
x_r

BiLSTM

Result: 49M -> 1M

- `BUFFERFLAG_CODECCONFIG -> [buffer, flag, codec, config]`
- `metamodelength -> [meta, mode, length]`

Identifier embeddings

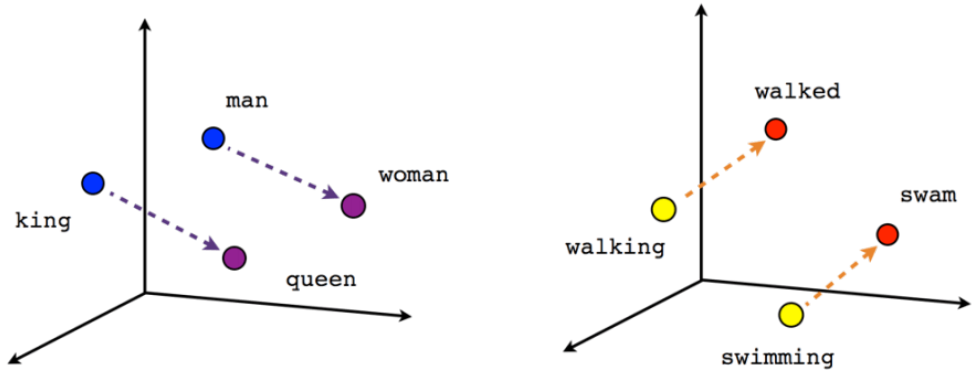
Nearest to “foo”

- afoo
- myfoo
- mfoo
- dofoo
- qux
- baz
- wibble
- quux
- testing

Analogies

“send” - “receive” + “pop” = “push”

“database” - “query” + “tune” = “settings”



$V_1 \Leftrightarrow$ `"foo"`

$V_2 \Leftrightarrow$ `"bar"`

$V_3 \Leftrightarrow$ `"integrate"`

$$distance(V_1, V_2) < distance(V_1, V_3)$$

$$distance(V_i, V_j) = \arccos \frac{V_i \cdot V_j}{\|V_i\| \|V_j\|}$$

Scalar product / Norm

How to estimate

$$V_i \cdot V_j \, ?$$

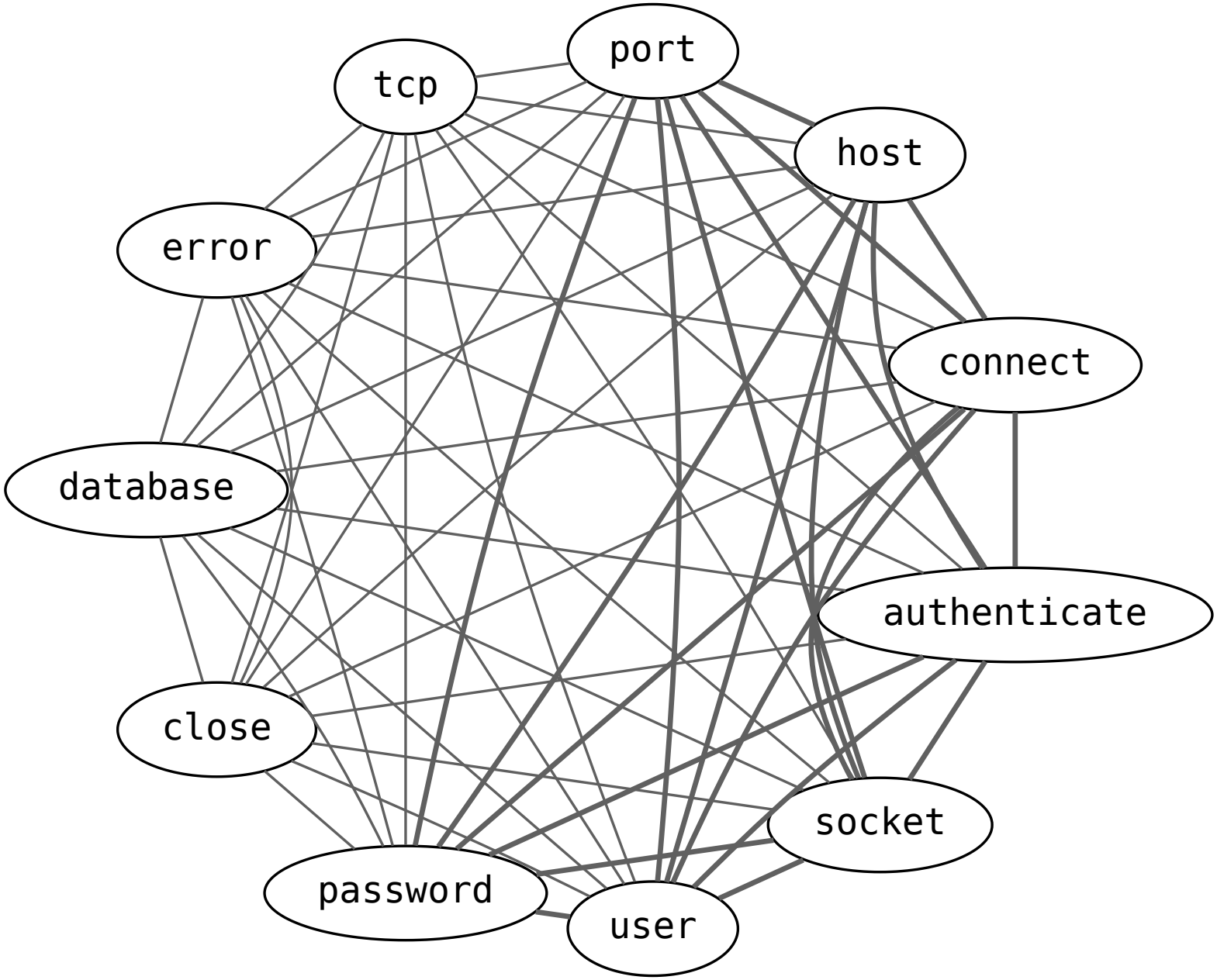
```
class Database:
    def connect(self, user, password, host, port):
        self._tcp_socket_connect(host, port)
        try:
            self._authenticate(user, password)
        except AuthenticationError as e:
            self.socket.close()
            raise e from None
```

Splitting and normalization

```
01. _tcp_socket_connect -> [tcp, socket, connect]
02. AuthenticationError -> [authentication, error]
03. authentication, authenticate -> authenticate
```

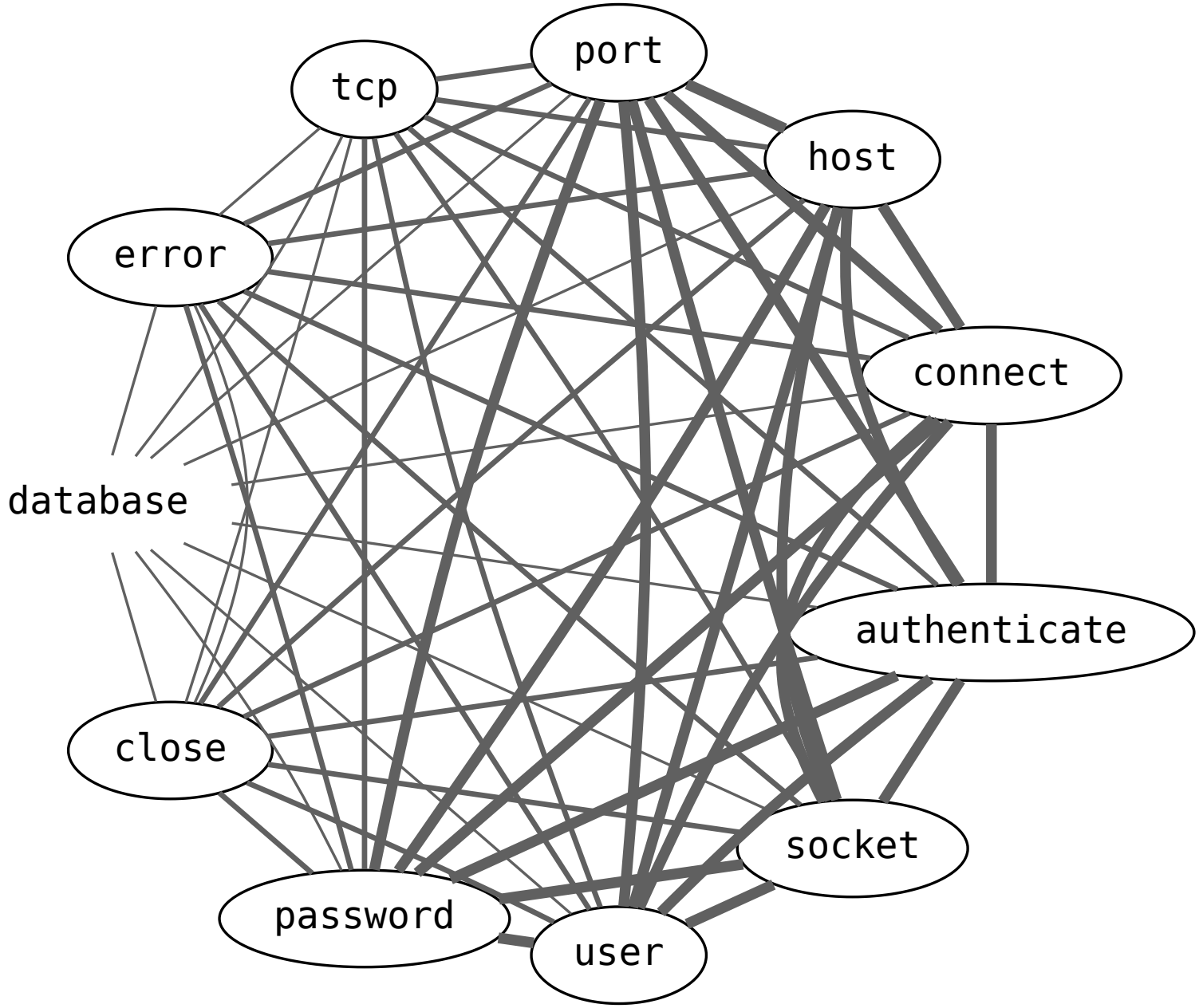
```
01. class Database:
02.     def connect(self, user, password, host, port):
03.         self._tcp_socket_connect(host, port)
04.         try:
05.             self._authenticate(user, password)
06.         except AuthenticationError as e:
07.             self.socket.close()
08.             raise e from None
```

```
>>> database, connect², user², password², host², port²,
    tcp, socket², authenticate², error, close
```

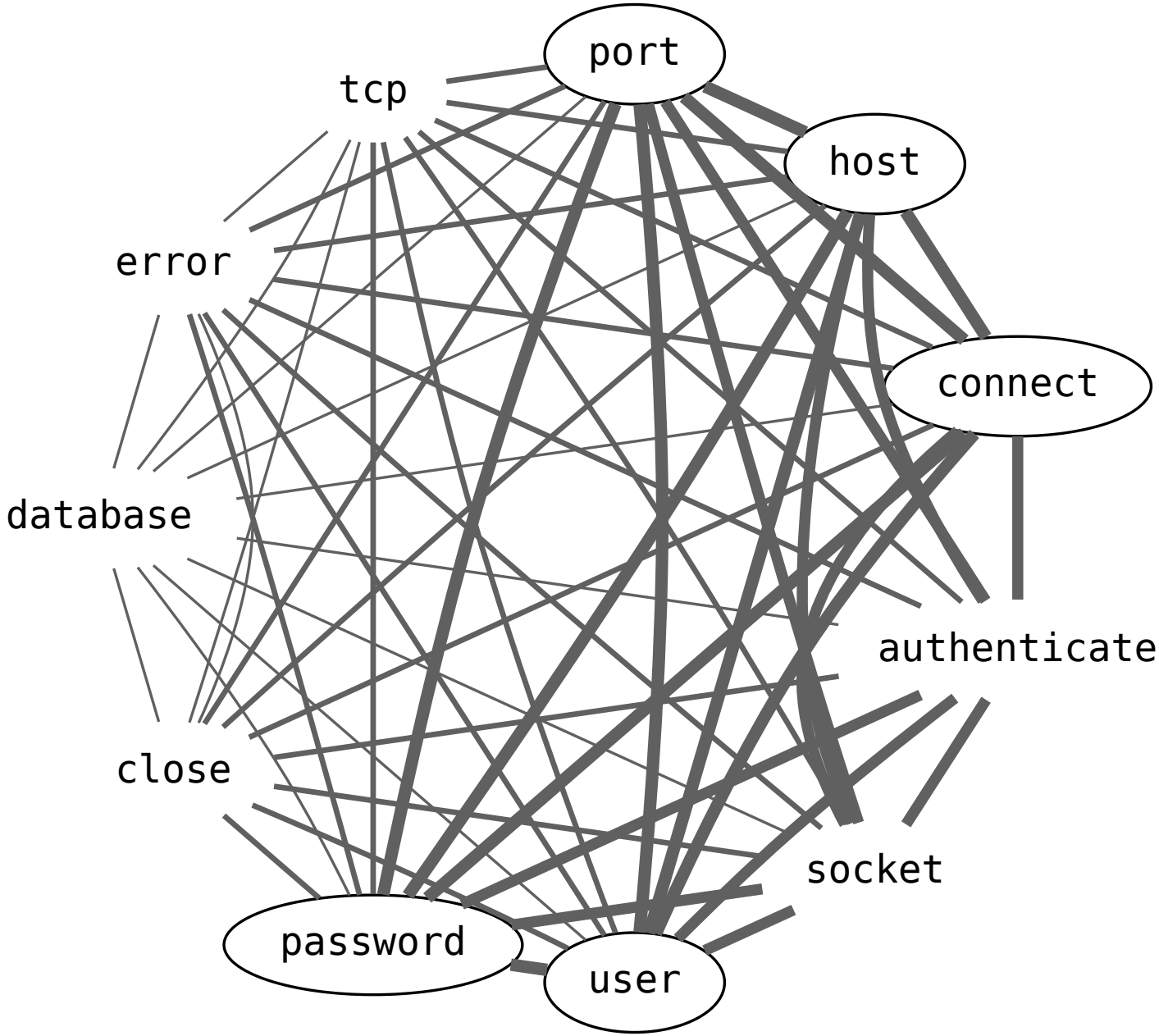

port
tcp
host
error
connect
database
authenticate
close
socket
password
user

```
01. class Database:
02.     def connect(self, user, password, host, port):
03.         self._tcp_socket_connect(host, port)
04.         try:
05.             self._authenticate(user, password)
06.         except AuthenticationError as e:
07.             self.socket.close()
08.             raise e from None
```

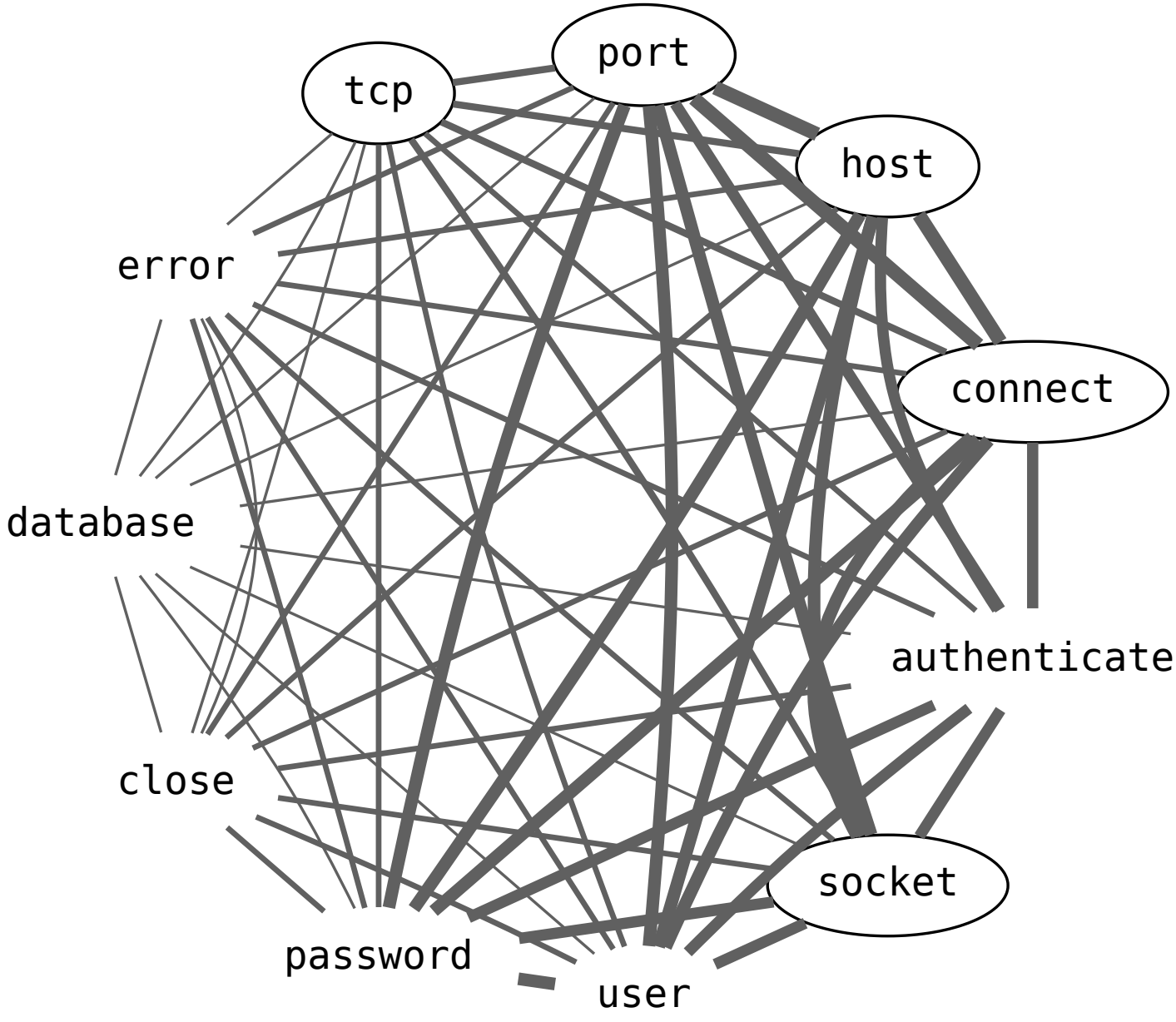
```
>>> connect², user², password², host², port², tcp, socket²,
    authenticate², error, close
```


port
tcp
host
error
connect
database
authenticate
close
socket
password
user

```
01. class Database:
02.     def connect(self, user, password, host, port):
03.         self._tcp_socket_connect(host, port)
04.         try:
05.             self._authenticate(user, password)
06.         except AuthenticationError as e:
07.             self.socket.close()
08.             raise e from None

>>> connect, user, password, host, port
```

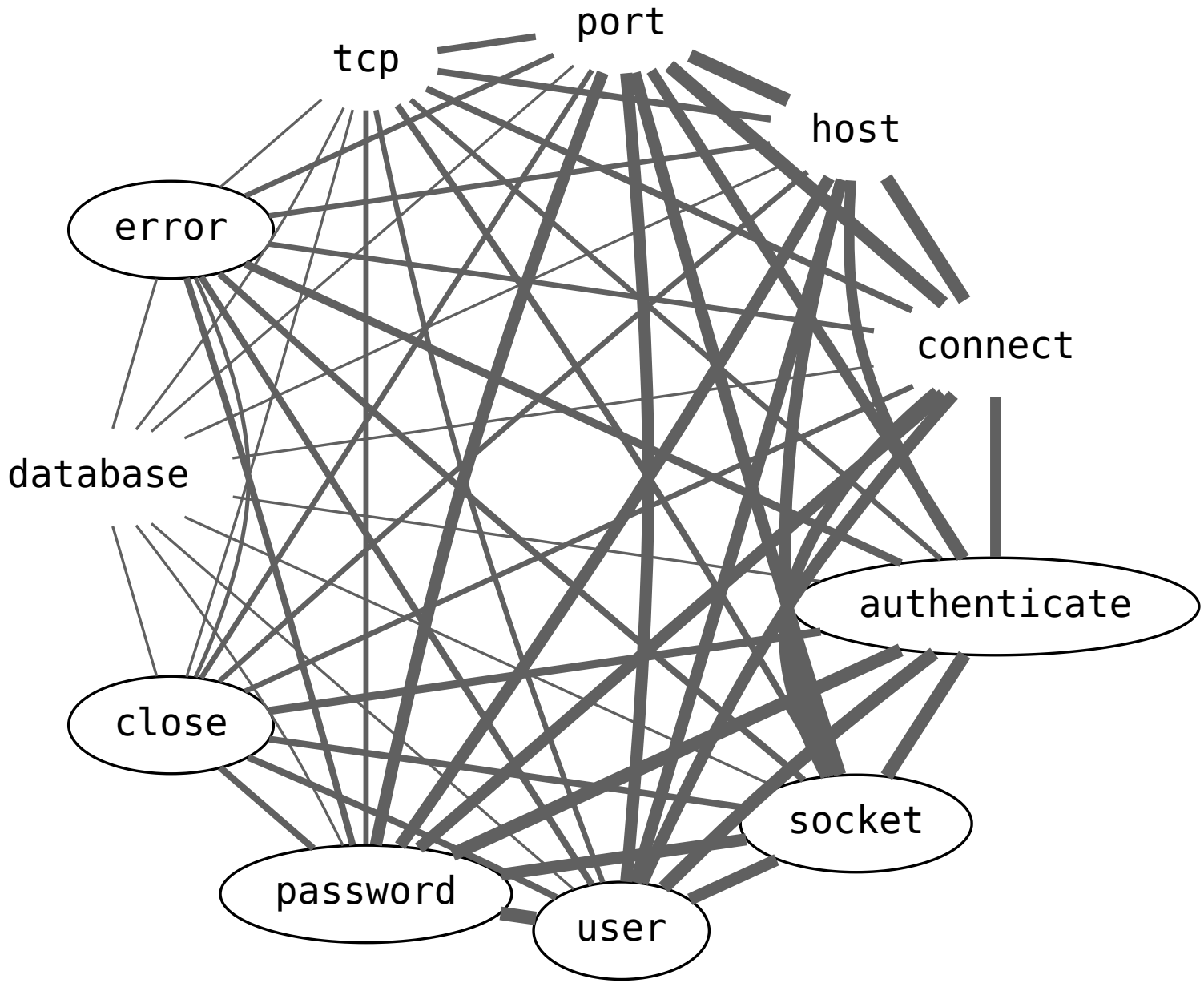

port
tcp
host
error
connect
database
authenticate
close
socket
password
user

```
01. class Database:
02.     def connect(self, user, password, host, port):
03.         self._tcp_socket_connect(host, port)
04.         try:
05.             self._authenticate(user, password)
06.         except AuthenticationError as e:
07.             self.socket.close()
08.             raise e from None

>>> tcp, socket, connect, host, port
```

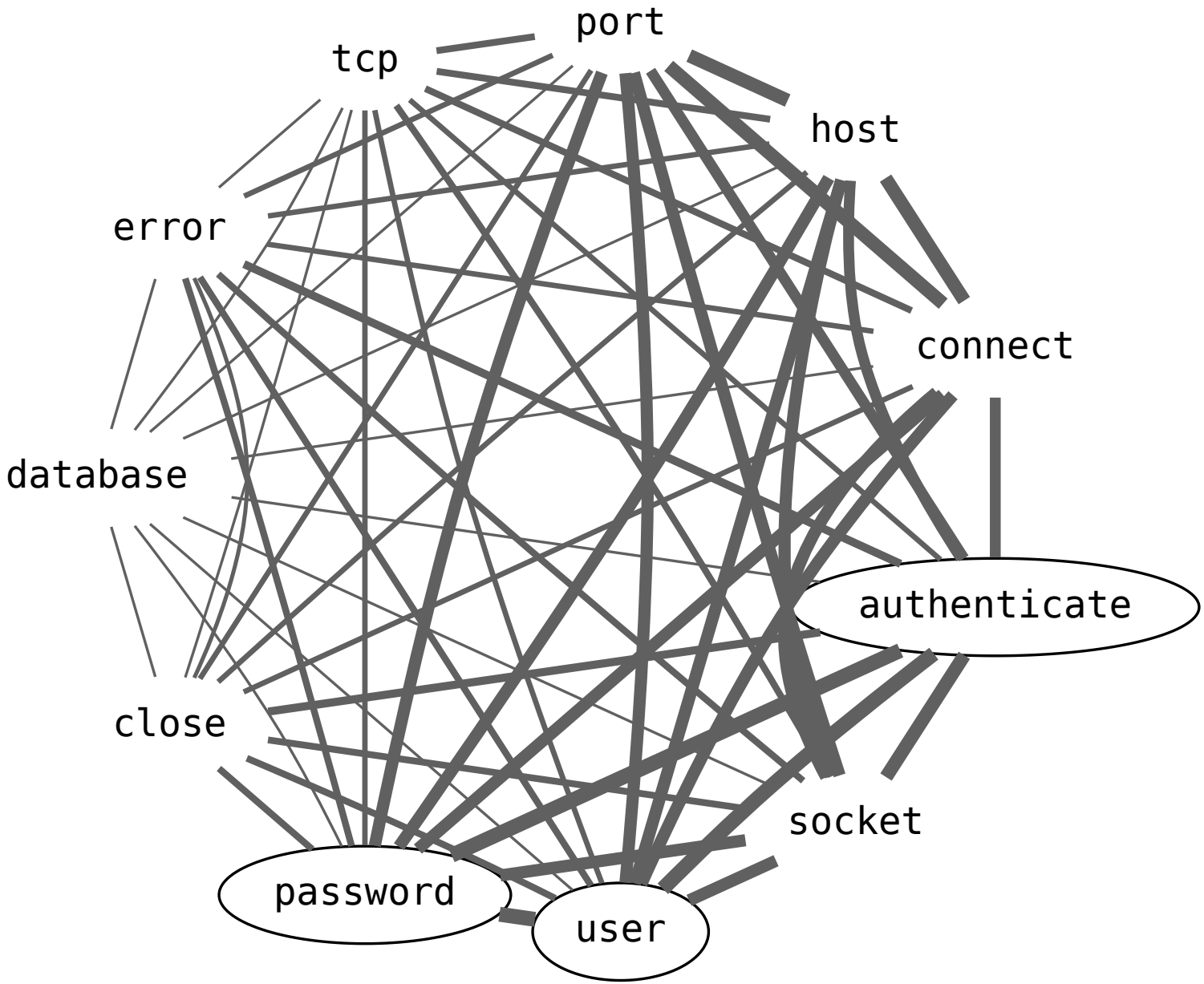

port
tcp
host
error
connect
database
authenticate
close
socket
password
user

```
01. class Database:
02.     def connect(self, user, password, host, port):
03.         self._tcp_socket_connect(host, port)
04.         try:
05.             self._authenticate(user, password)
06.         except AuthenticationError as e:
07.             self.socket.close()
08.             raise e from None
```

```
>>> authenticate², user, password, error, socket, close
```

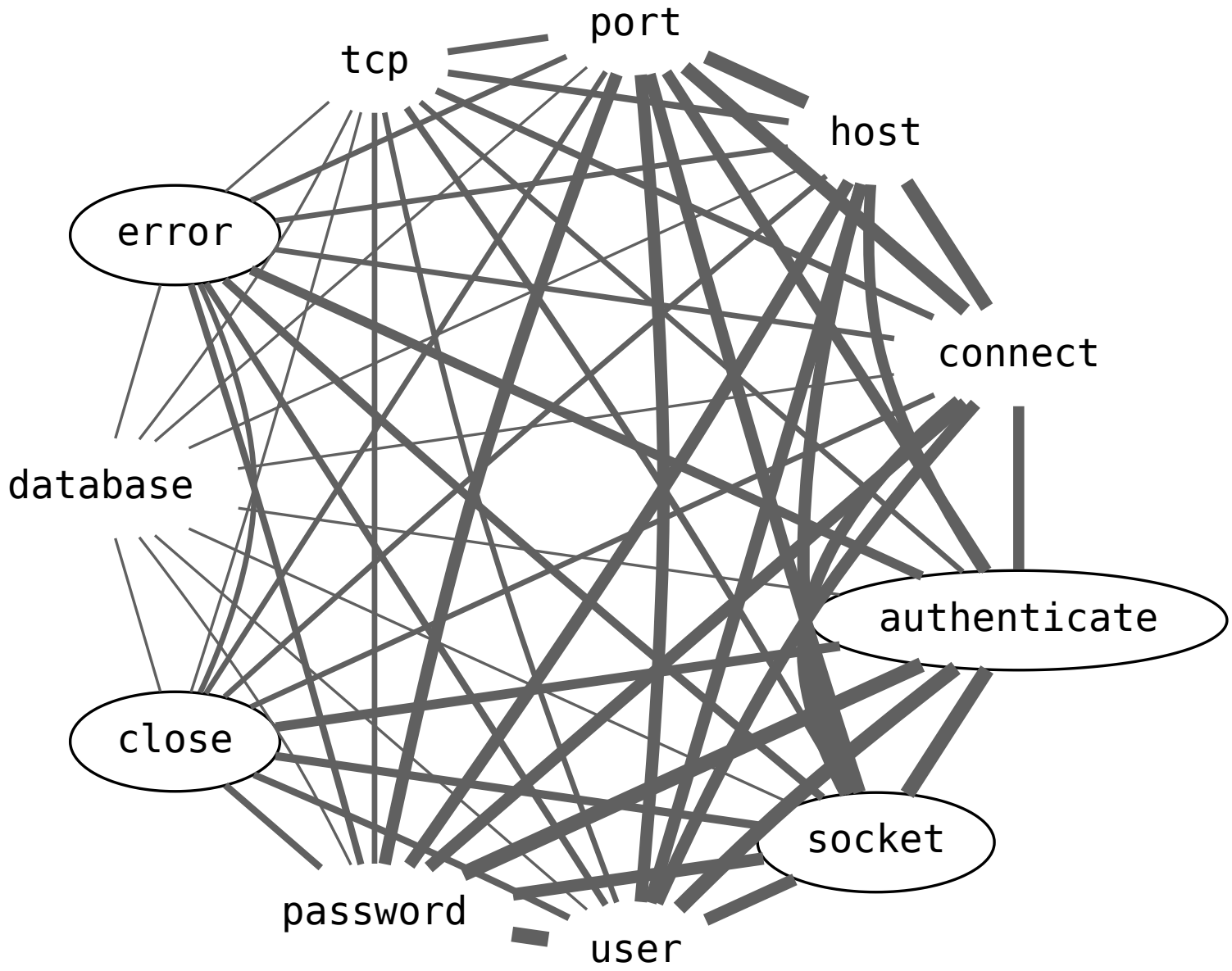

port
tcp
host
error
connect
database
authenticate
close
socket
password
user

```
01. class Database:
02.     def connect(self, user, password, host, port):
03.         self._tcp_socket_connect(host, port)
04.         try:
05.             self._authenticate(user, password)
06.         except AuthenticationError as e:
07.             self.socket.close()
08.             raise e from None

>>> authenticate, user, password
```

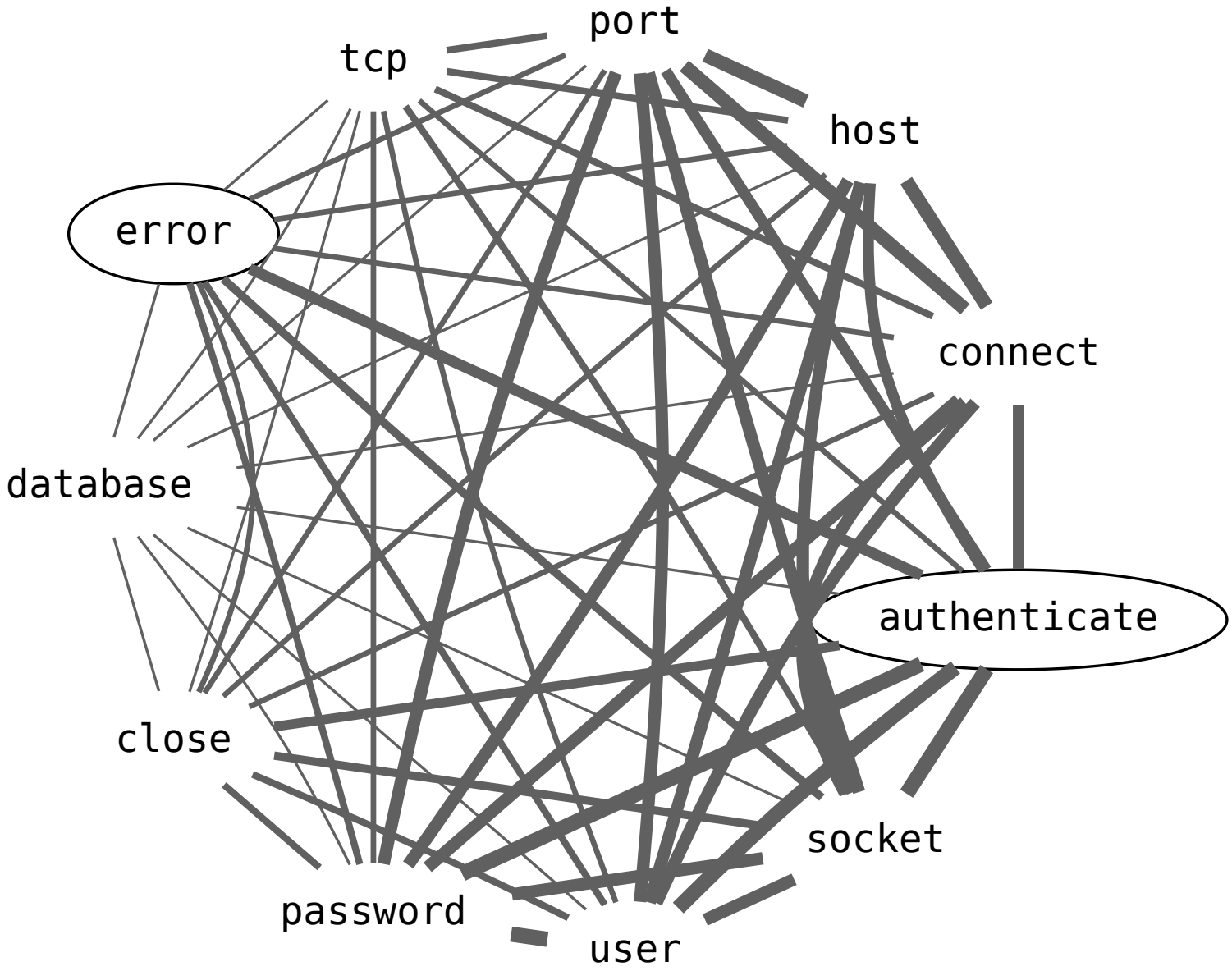

port
tcp
host
error
connect
database
authenticate
close
socket
password
user

```
01. class Database:
02.     def connect(self, user, password, host, port):
03.         self._tcp_socket_connect(host, port)
04.         try:
05.             self._authenticate(user, password)
06.         except AuthenticationError as e:
07.             self.socket.close()
08.             raise e from None
```

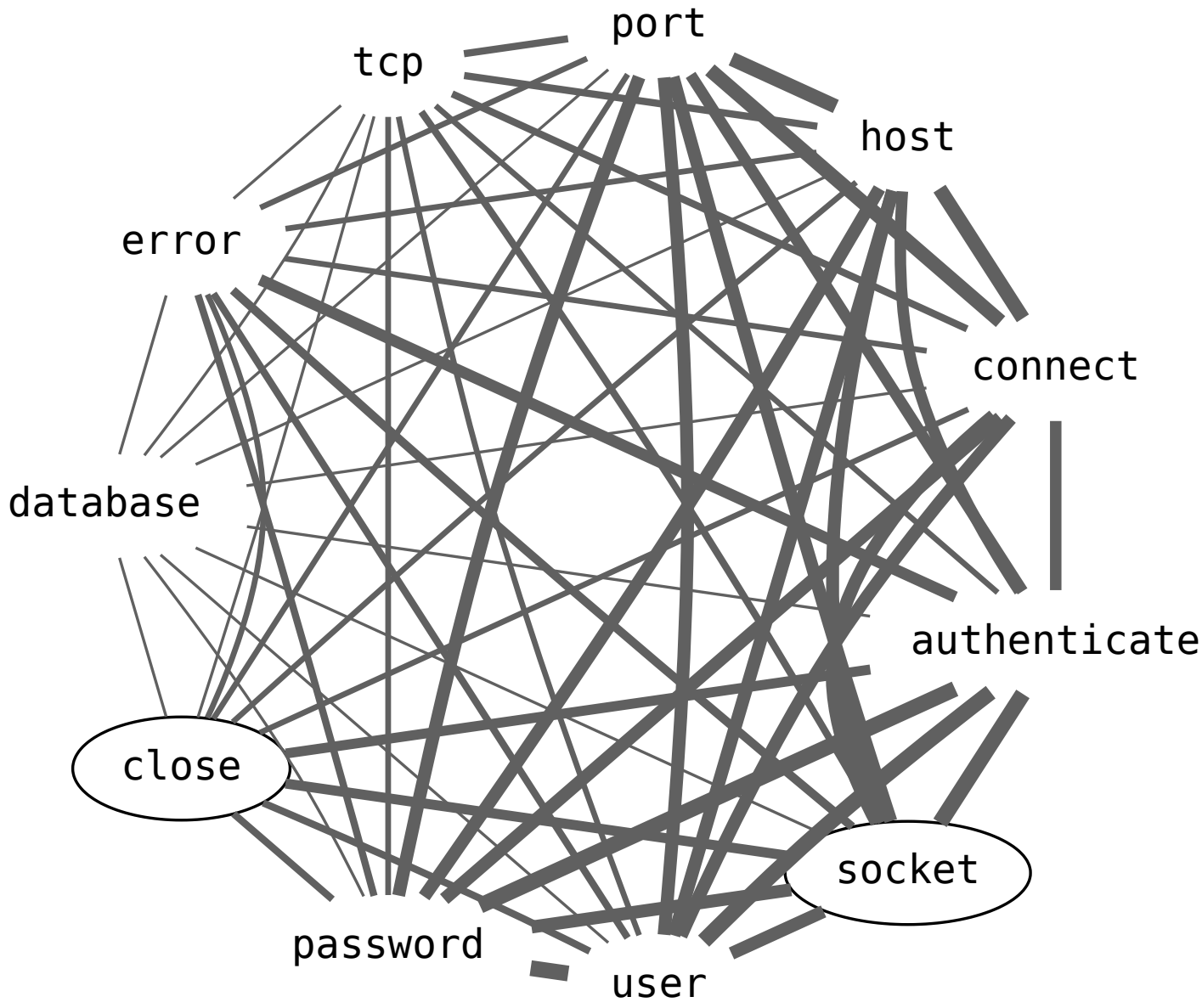
```
>>> authenticate, error, socket, close
```

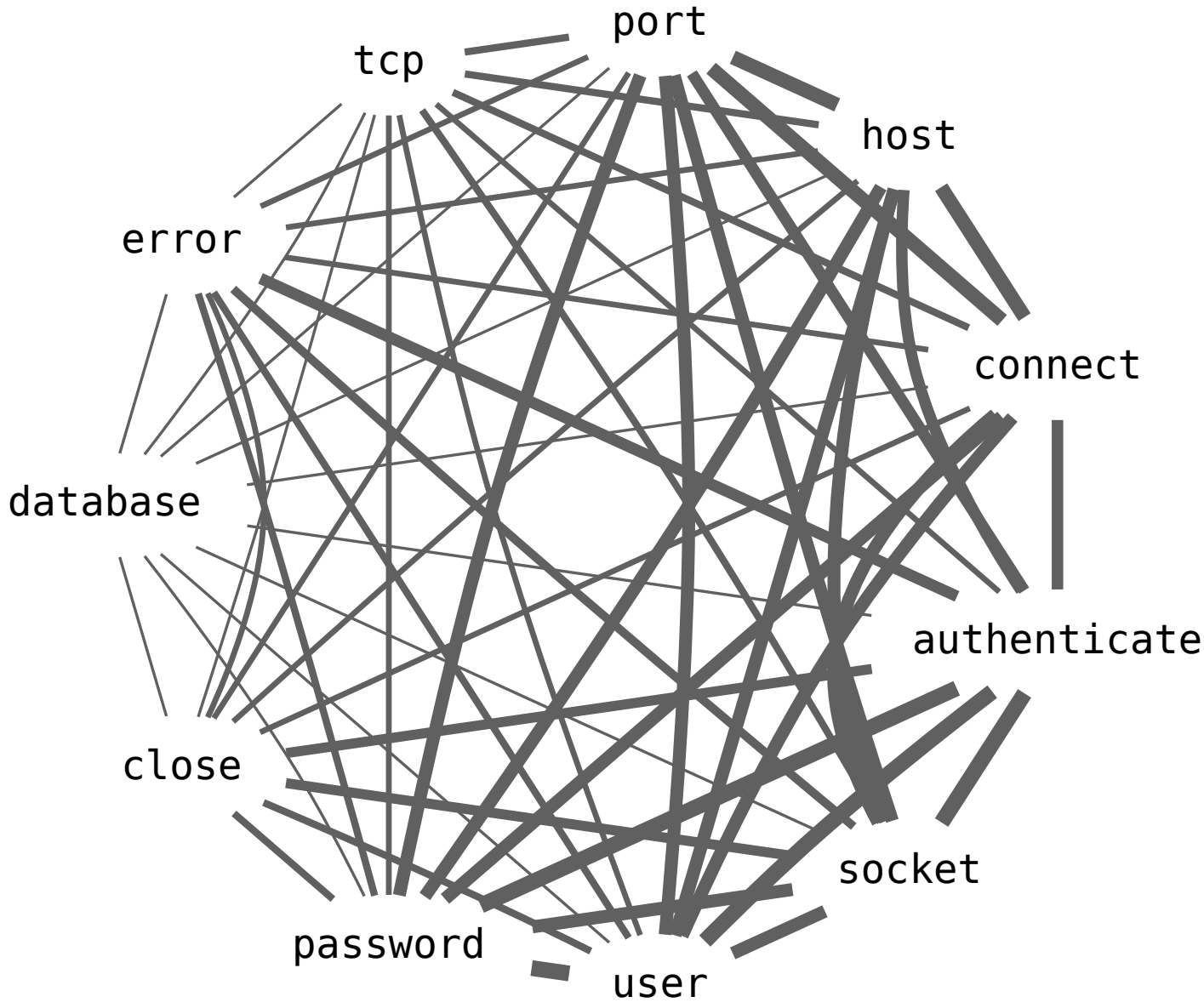

port
tcp
host
error
connect
database
authenticate
close
socket
password
user

```
01. class Database:
02.     def connect(self, user, password, host, port):
03.         self._tcp_socket_connect(host, port)
04.         try:
05.             self._authenticate(user, password)
06.         except AuthenticationError as e:
07.             self.socket.close()
08.             raise e from None

>>> authenticate, error
```


port
tcp
host
error
connect
database
authenticate
close
socket
password
user

```
01. class Database:
02.     def connect(self, user, password, host, port):
03.         self._tcp_socket_connect(host, port)
04.         try:
05.             self._authenticate(user, password)
06.         except AuthenticationError as e:
07.             self.socket.close()
08.             raise e from None

>>> socket, close
```


port
tcp
host
error
connect
database
authenticate
close
socket
password
user


port
tcp
host
error
connect
database
authenticate
close
socket
password
user

Incidence matrix C_{ij}

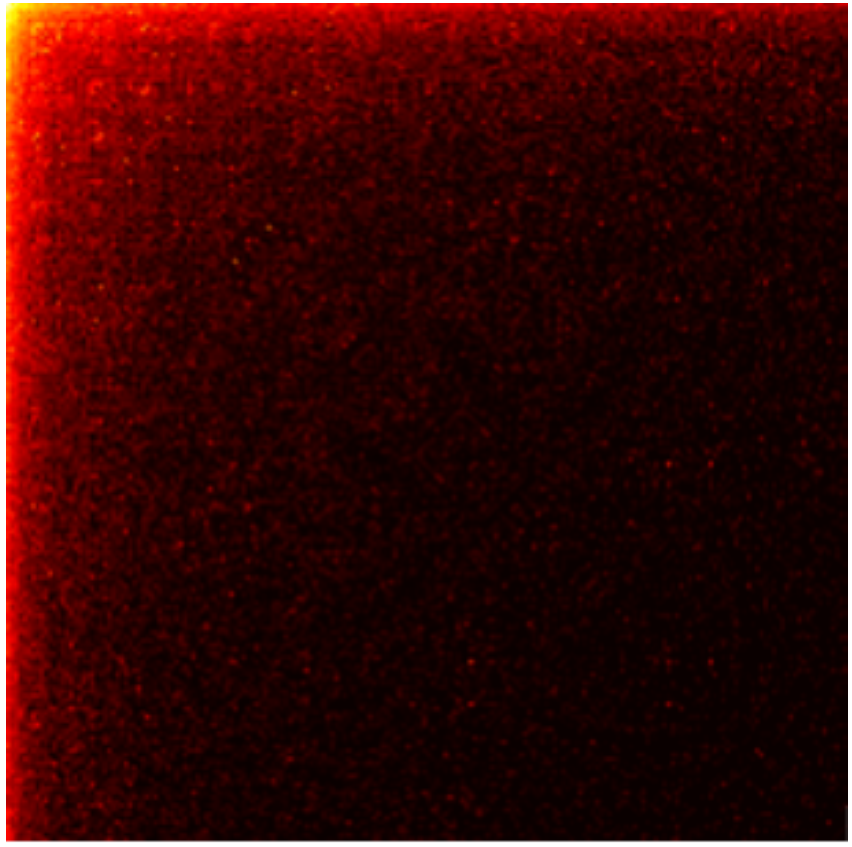
- $C_{ij} =$ number of times i and j were together ❤
- Also known as the **co-occurrence matrix**

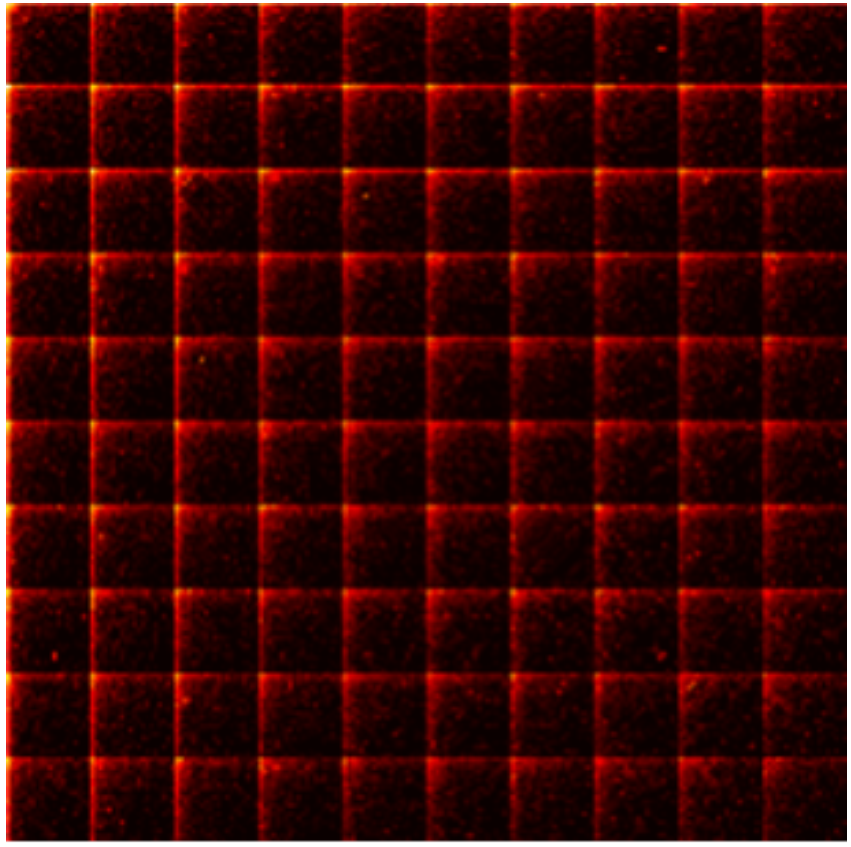
Pointwise Mutual Information (PMI)

$$V_i \cdot V_j = PMI_{ij} = \log \frac{C_{ij} \sum C}{\sum_{k=1}^{N} C_{ik} \sum_{k=1}^{N} C_{jk}}$$

Representation Learning on Explicit Matrix

Stochastic Gradient Descent





Swivel

- ✔ Multi-GPU
- ✔ Multi-node
- ✔ Quality tricks
- ✔
- ✔

Similar repository search

torch/nn

src-d/go-git

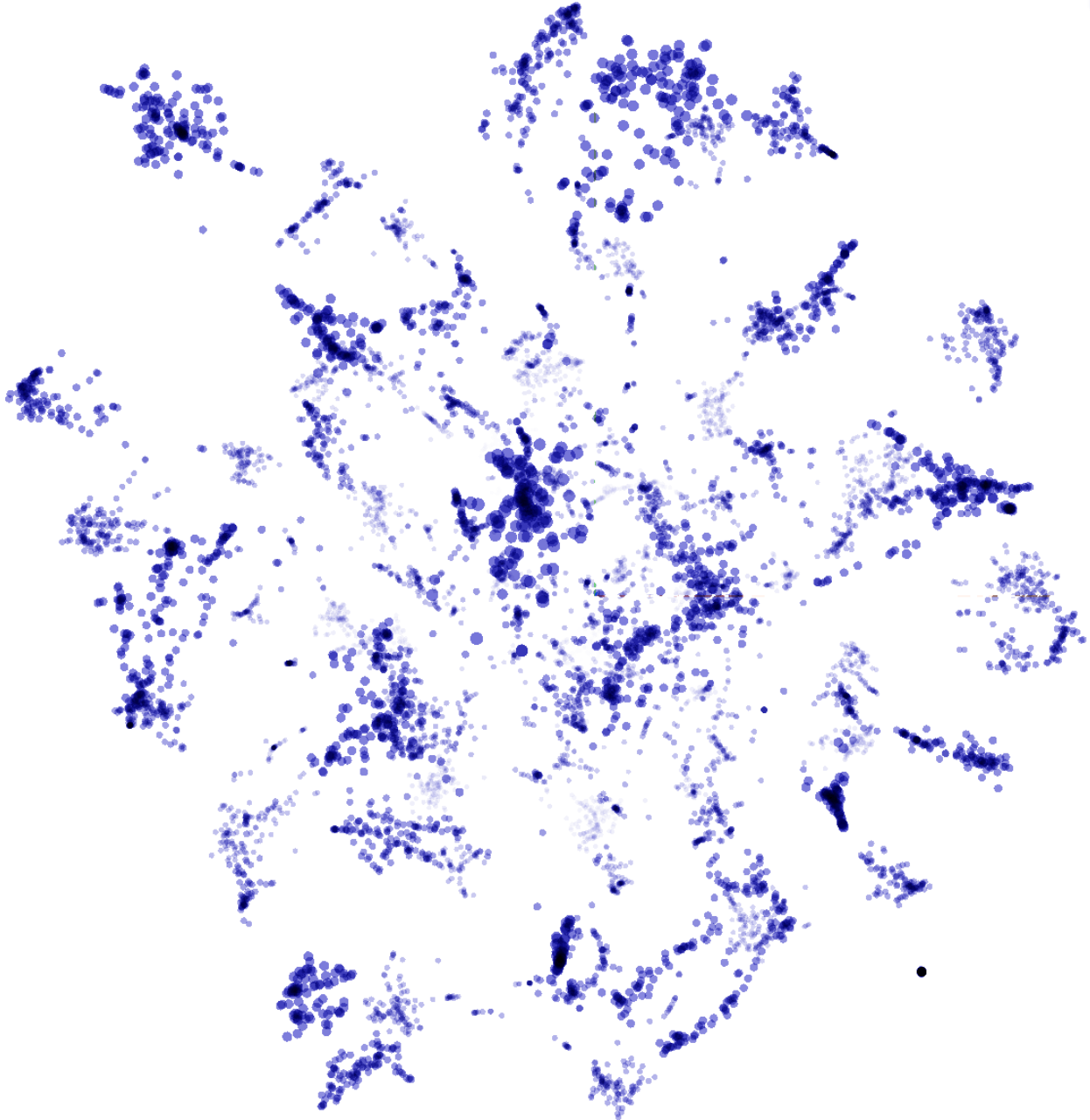
Pipeline

1. Clone the repositories.
2. Classify and parse source code.
3. Preprocess identifiers: split, normalize, filter, stem.
4. Prepare embeddings - cooc matrix & Swivel.
5. Weighted BOW per repository.
6. WMD.

Training identifier embeddings

Stage	Time	Resources	Size on disk
Cloning 143k repos	3 days	20x2 cores, 256 GB RAM	2.6 TB
Dataset	4 days	20x2 cores, 256 GB RAM	2TB (31GB in xz)
fastprep	2 days	16x2 cores, 256 GB RAM	20 GB
Swivel	14 hours	2 Titan X'2016 + 2 1080Ti	5.6 GB

Embeddings



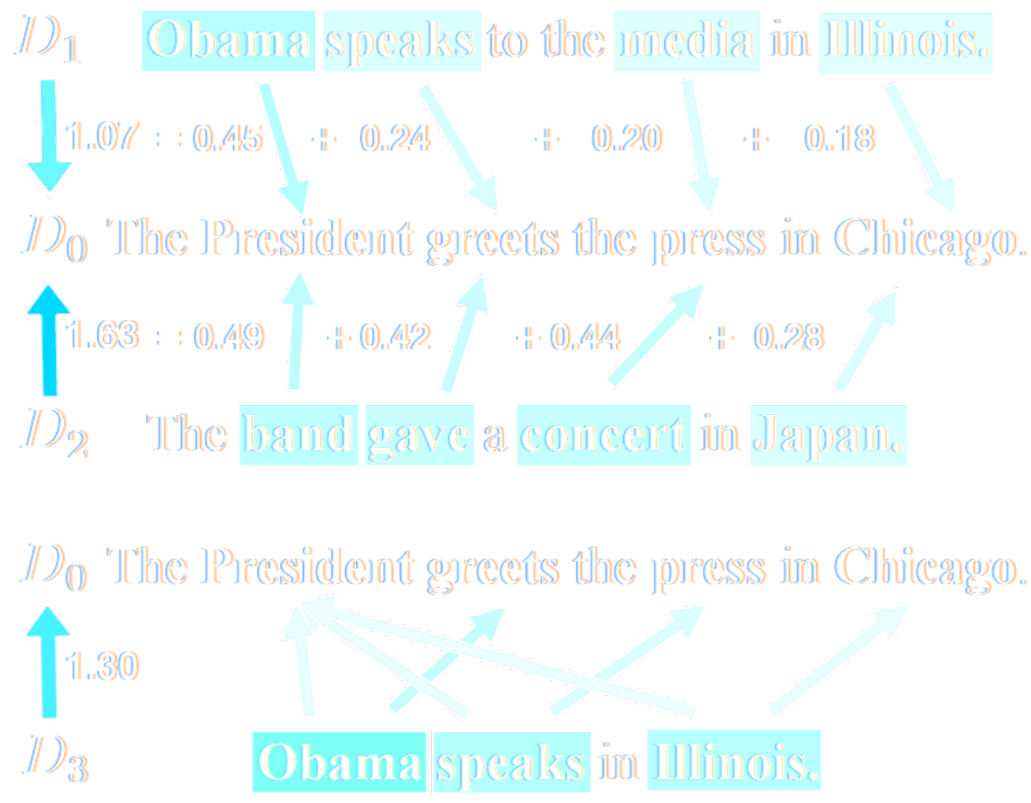
Weighted nBOW

Let's interpret every repository as the weighted bag-of-words.

We calculate TF-IDF to weight the occurring identifiers.

`tensorflow/tensorflow:`

tfreturn	67.780249
oprequires	63.968142
doblas	63.714424
gputools	62.337396
tfassign	61.545926
opkernel	60.721556
sycl	57.064558
hlo	55.723587
libxsmm	54.820668
tfdisallow	53.666890


D_1 Obama speaks to the media in Illinois.
1.07
0.45
0.24
0.20
0.18
D_0 The President greets the press in Chicago.
1.63
0.49
0.42
0.44
0.28
D_2 The band gave a concert in Japan.
D_0 The President greets the press in Chicago.
1.30
D_3 Obama speaks in Illinois.

WMD calculation in a nutshell

WMD evaluation is $O(N^3)$, becomes slow on $N \approx 100$.

1. Sort samples by centroid distance.
2. Evaluate first k WMDs.
3. For every subsequent sample, solve the relaxed LP which gives an upper estimation.
4. If it is greater than the farthest among the k NN, go next.
5. Otherwise, evaluate the WMD.

This allows to avoid 95% WMD evaluations on average.

Code review

5K

7M+

25M+

1M

0

109

14K

495

1,054,066

121,280

1,175,346 issues

So

Typo

"Use the get keyword to make a getter method isValid that checks if the the given 4 side
is valid. A square is valid when the lengths of all sides are equal." Bolded section should

learn-co-curriculum/fewpjs-class-extensions-extends-lab Opened by RylanBauermeister 2 days ag

typo

ThinkR-open/building-shiny-apps-workflow Opened by tellyshia a day ago

Typo

Label above JSON example for filtering by Subject reads "The JSON syntax for filtering b
is:". It should read "The JSON syntax for filtering by subject is:" Document Details ...

MicrosoftDocs/azure-docs Opened by rajues 3 days ago • 2 comments

Typos correction inside code identifiers

```
01. class ClasName:
02.     @classmethod
03.     def function_name(cls) -> str:
04.         varyable_name = "Hello, I'm ClassName object!"
05.         return varyable_name
```

Typos correction

funktion → function

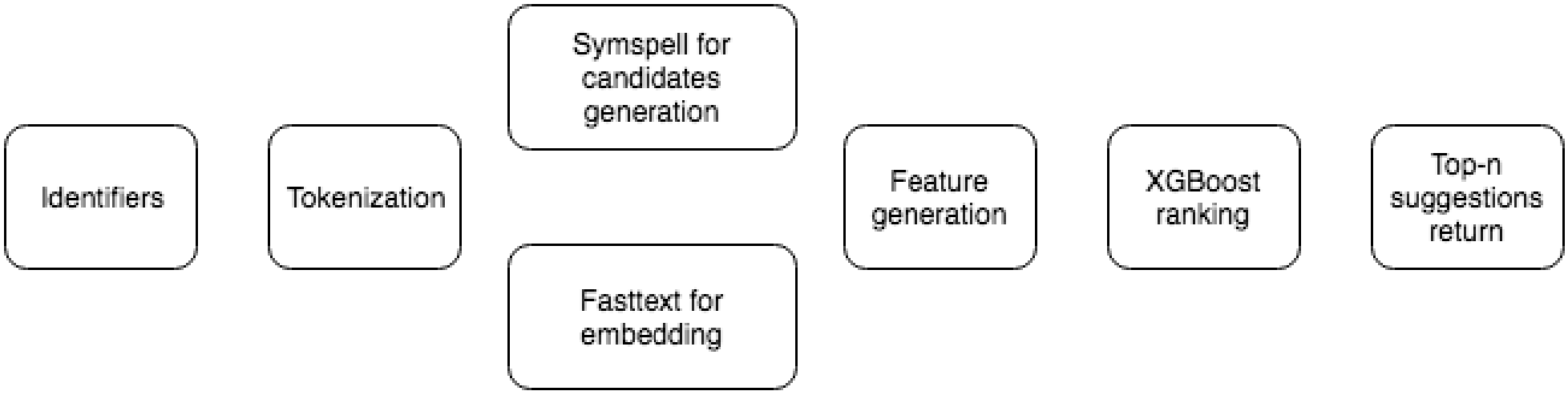
GetValu → GetValue

str_lenght → str_length

Typos correction inside code identifiers

```
01. class ClassName:
02.     @classmethod
03.     def function_name(cls) -> str:
04.         variable_name = "Hello, I'm ClassName object!"
05.         return variable_name
```

The correction pipeline



Candidates generation

1. Find candidates with the
2. Identifier to which the token belongs - a context
3. Features: based on the frequencies, embeddings, similarity and edit distance between the elements

Candidates ranking

- Binary classification model on tuples `(identifier, token, candidate)`:
  - `label=1` if the candidate is the correct suggestion
  - `label=0` otherwise
- Group by the token and sort
- is used for binary classification

Suggestions

- Suggested token itself - not correcting
- Otherwise return corrections

Training and testing

- Train and test datasets are sampled from the dataset of identifiers
- Random artificial typos
- **Dataset itself contains typos**

Results on a token level

- Detection precision - 98%
- Detection recall - 82%
- Top3 correction accuracy - 95%

Note: the test dataset is not a ground truth.

Developer similarity

Pipeline

1. Clone the organization.
2. Classify and parse source code.
3. Extract identifiers per file.
4. Extract experience per developer - languages, files, etc.
5. Topic modeling.
6. Usage: KD-tree, clustering and so on.

From git history

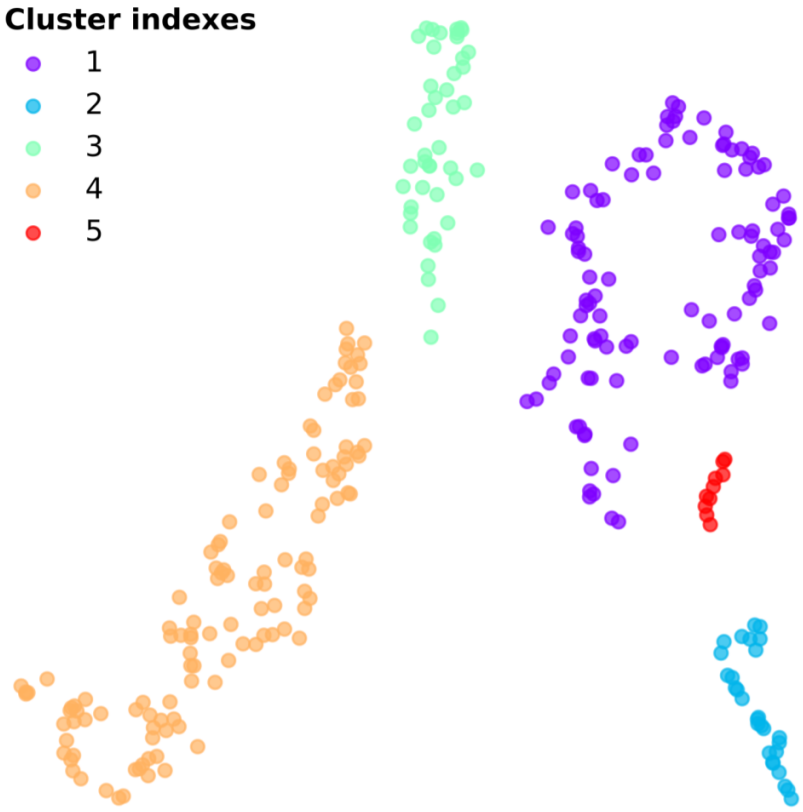
Dev\Files	File 1	File 2	File 3
Dev 1	0	0.1	0.8
Dev 2	0.4	0.1	0.5

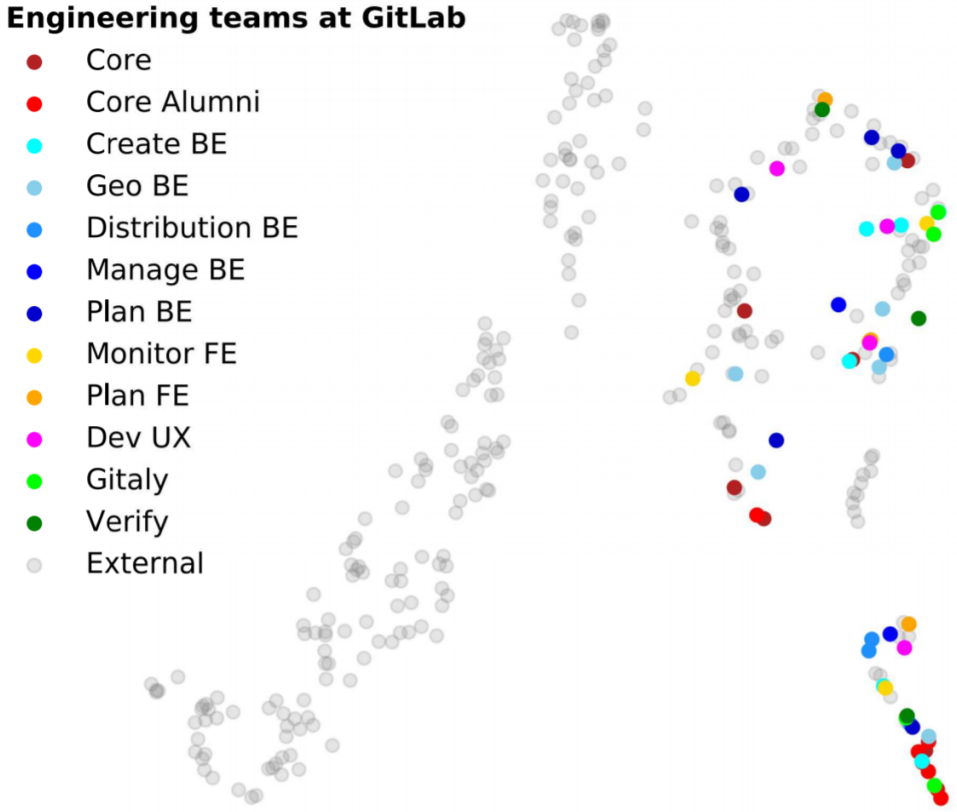
From head revision

Files\Tokens	Token 1	Token 2	Token 3
File 1	0	0.1	0.8
File 2	0.4	0.1	0.56
File 3	0.43	0.18	0.9

Result after multiplication

Dev\Token	Token 1	Token 2	Token 3
Dev 1	0	0.1	0.8
Dev 2	0.4	0.1	0.5


Cluster indexes
1
2
3
4
5


Engineering teams at GitLab
Core
Core Alumni
Create BE
Geo BE
Distribution BE
Manage BE
Plan BE
Monitor FE
Plan FE
Dev UX
Gitaly
Verify
External

Summary

- MLonCode is fun
- There is data
- There are tools
- Community is forming

Idea? Doubt? Write us!

Thank you