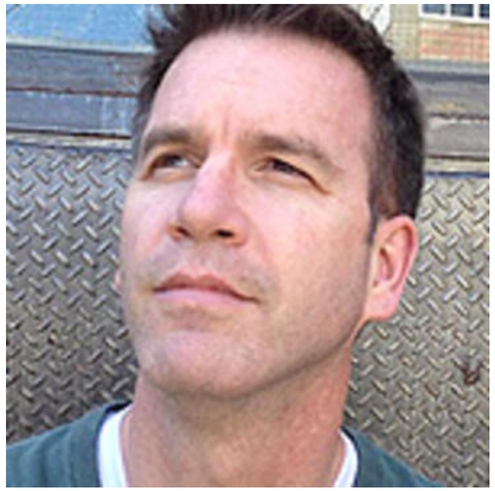
Towards a Universal Code Formatter through Machine Learning

Terence Parr
Google

Jurgen Vinju
Centrum Wiskunde & Informatica
Jurgen.Vinju@cwi.nl
Swat.engineering BV



SLE 2026

July 28th 2026

Rennes, France & Zoom, Internet

TOWARDS A UNIVERSAL CODE FORMATTER …

Amsterdam

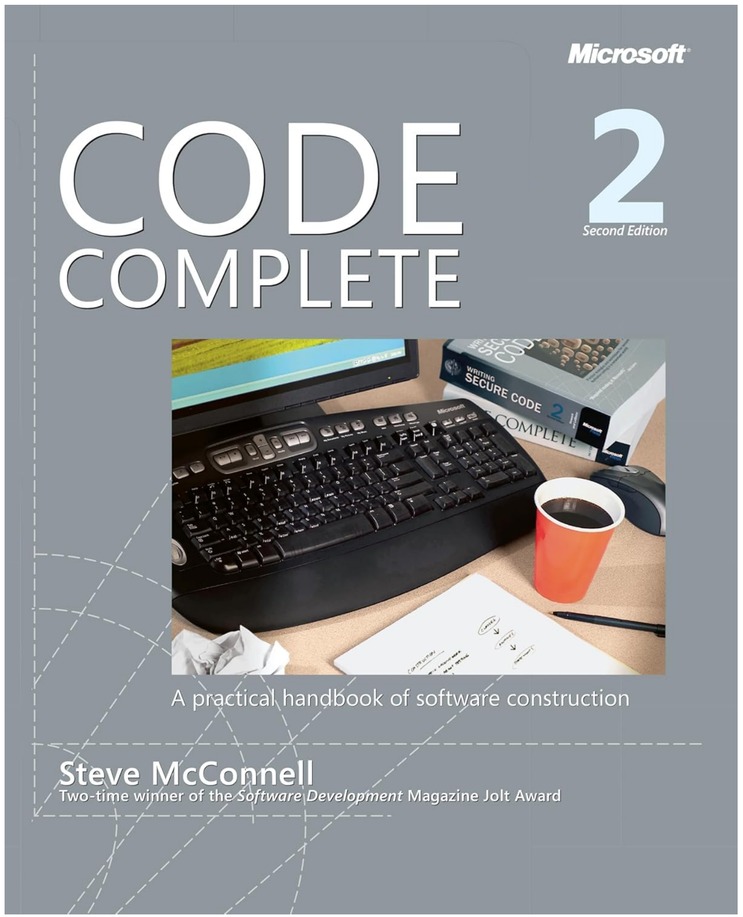
Terence Parr, Google

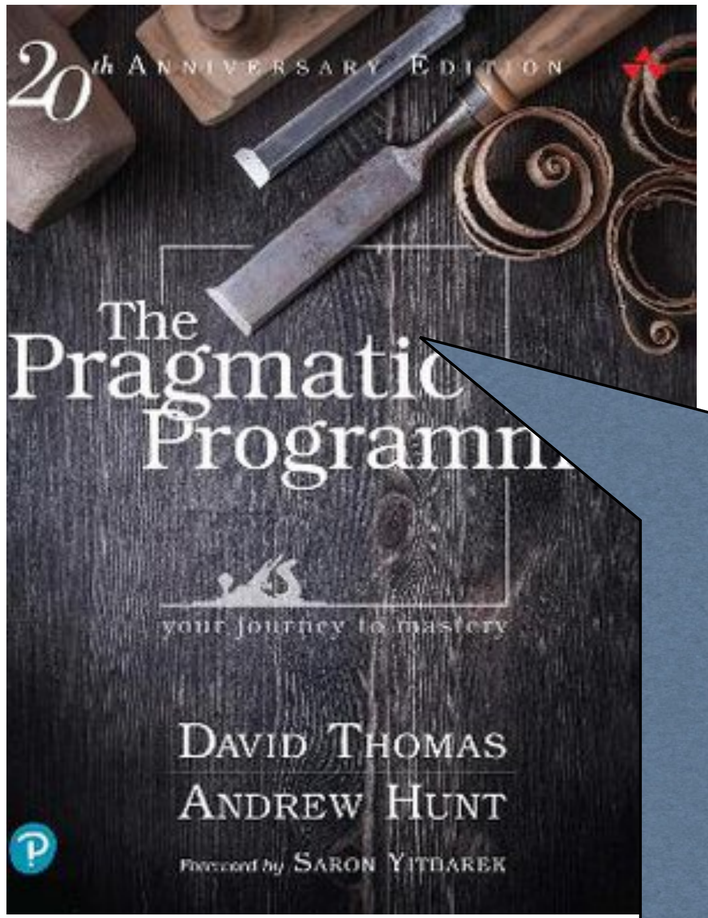
Jurgen Vinju, *Centrum Wiskunde & Informatica*

Swat.engineering BV

November 1, 2016

Why code formatting?






```
{
    OSStatus        err;
    SSLBuffer       hashOut, hashCtx, clientRandom, serverRandom;
    uint8_t         hashes[SSL_SHA1_DIGEST_LEN + SSL_MD5_DIGEST_LEN];
    SSLBuffer       signedHashes;
    uint8_t         *dataToSign;
    size_t          dataToSignLen;

    ...
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
```

consistency:
feeling "at home"
fewer bugs
fewer distractions

```
/** \file
 * Copyright (c) 1999, 2000 Carlo Wood.  All rights reserved. <br>
 * Copyright (c) 1994, 1996, 1997 Joseph Arceneaux.  All rights reserved. <br>
 * Copyright (c) 1992, 2002, 2008 Free Software Foundation, Inc.  All rights reserved.
<br>
 *
 * Copyright (c) 1980 The Regents of the University of California. <br>
 * Copyright (c) 1976 Bo                                          linois. All rights
reserved.
 * Copyright (c) 1985 Su
 *   All rights reserved
 *
 * Redistribution and us                                          ithout
 * modification, are per                                          itions
 * are met:
 * - 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * - 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * - 3. Neither the name of the University nor the names of its contributors
```

Dart people said: "this is the hardest program I've ever written.

C indent has hundreds of options. K&R style:

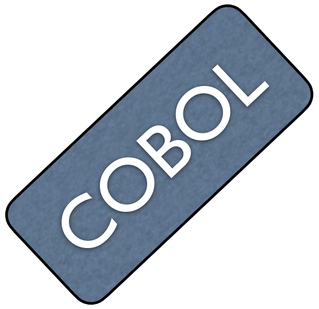
```
$ indent -nbad -bap -bbo -nbc -br -brs -c33 -cd33 -ncdb -ce -ci4
-cli0 -cp33 -cs -d0 -di1 -nfc1 -nfca -hnl -i4 -ip0 -l75 -lp -npcs -nprs
-npsl -saf -sai -saw -nsc -nsob -nss
```

Formatting is at least as complex as a language

```
SELECT DISTINCT
    t.server_name
        , t.server_id
        , 'Message Queuing Service' AS missingmonitors
FROM t_server t INNER JOIN t_server_type_assoc tsta ON t.server_id = tsta.server_id
WHERE t.active = 1 AND tsta.type_id IN ('8')
      AND t.environment_id = 0
      AND t.server_name NOT IN
    (
          SELECT DISTINCT l.address
          FROM ipmongroups g INNER JOIN ipmongroupmembers m ON g.groupid = m.groupid
               INNER JOIN ipmonmonitors l ON m.monitorid = l.monitorid
               INNER JOIN t_server t ON l.address = t.server_name
               INNER JOIN t_server_type_assoc tsta ON t.server_id = tsta.server_id
          WHERE l.name LIKE '%Message Queuing Service%'
                AND t.environment_id = 0
                AND tsta.type_id IN ('8')
                AND g.groupname IN ('Prod O/S Services')
                AND t.active = 1
      )
UNION
ALL
```

Formatting is at least as complex as a language

```
switch ( c ) {
    ...
    default:
        if ( c==delimiterStopChar ) {
            consume();
            scanningInsideExpr = false;
            return newToken(RDELIM);
        }
        if ( isIDStartLetter(c) ) {
            ...
            if ( name.equals("if") ) return newToken(IF);
            else if ( name.equals("endif") ) return newToken(ENDIF);
            ...
            return id;
        }
        RecognitionException re = new NoViableAltException("", 0, 0, input);
        ...
        errMgr.lexerError(input.getSourceName(),
                          "invalid character '"+str(c)+"'",
                          templateToken,
                          re);
```



Formatting is at least as complex as a language

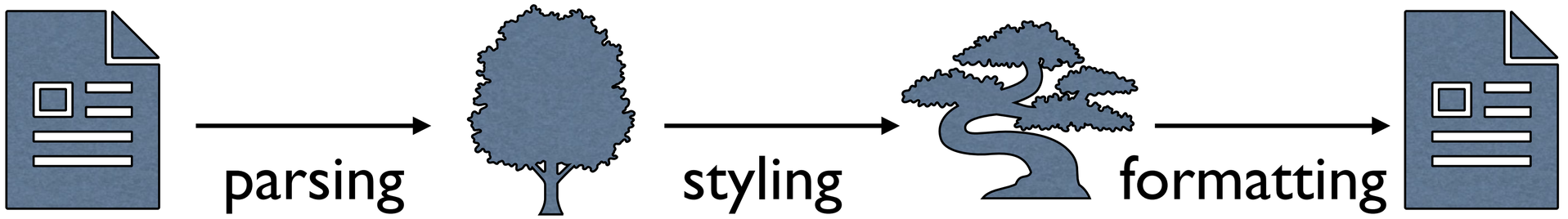
```
DATA DIVISION.
WORKING-STORAGE SECTION.

01  WD_DATE                                          VALUE ZER(
    88  WD_NO_DATE                                   VALUE ZER(
    03  WD_DD                                 PIC 9(02).
    03  WD_MM                                 PIC 9(02).
    03  WD_JJ                                 PIC 9(02).

PROCEDURE DIVISION.
INITIALIZE_DATE SECTION.
INIT_00.
        IF              WD_NO_DATE
            ACCEPT      WD_DATE                 FROM DATE
        END-IF
        MOVE            WD_DD                   TO  WR_DD
        MOVE            WD_MM                   TO  WR_MM
        MOVE            WD_JJ                   TO  WR_JJ.
```

The plateau: declarative formatting

staging allows for separation of concerns



programmatic *automatic*

"Styling" : linear in the amount of grammar rules (e.g. twice as many)

[3] J. Coutaz. The Box, a layout abstraction for user interface toolkits. CMU-CS-84-167, Carnegie Mellon University, 1984.

[10] D. C. Oppen. Prettyprinting. *ACM TOPLAS*, 2(4):465–483, 1980. ISSN 0164-0925.

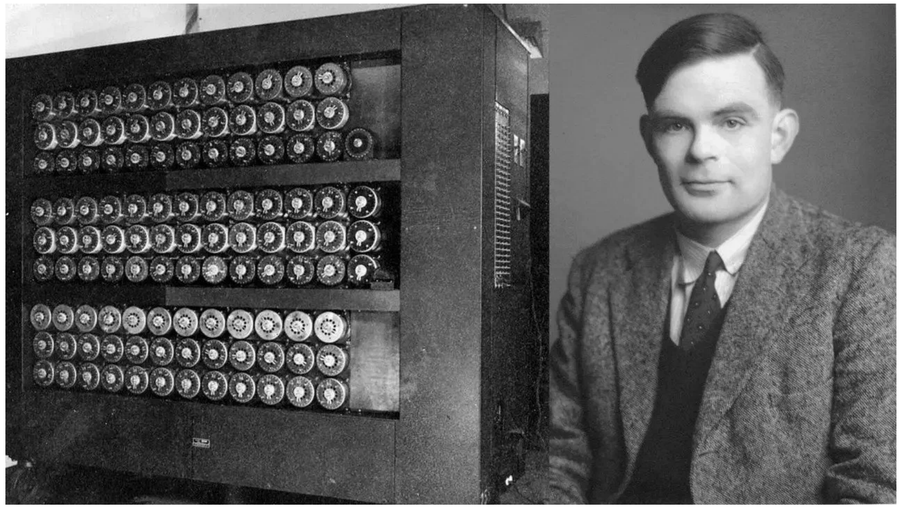
[15] M. G. J. van den Brand and E. Visser. Generation of formatters for context-free languages. *ACM Trans. Softw. Eng. Methodol.*, 5(1):1–41, 1996. ISSN 1049-331X. .

[1] A. H. Bagge and T. Hasu. A pretty good formatting pipeline. In *International Conferance on Software Language Engineering (SLE'13)*, volume 8225 of *LNCS*. Springer, Oct. 2013.

[4] M. de Jonge. Pretty-printing for software reengineering. In *Proceedings of ICSM 2002*, pages 550–559. IEEE Computer Society Press, Oct. 2002.

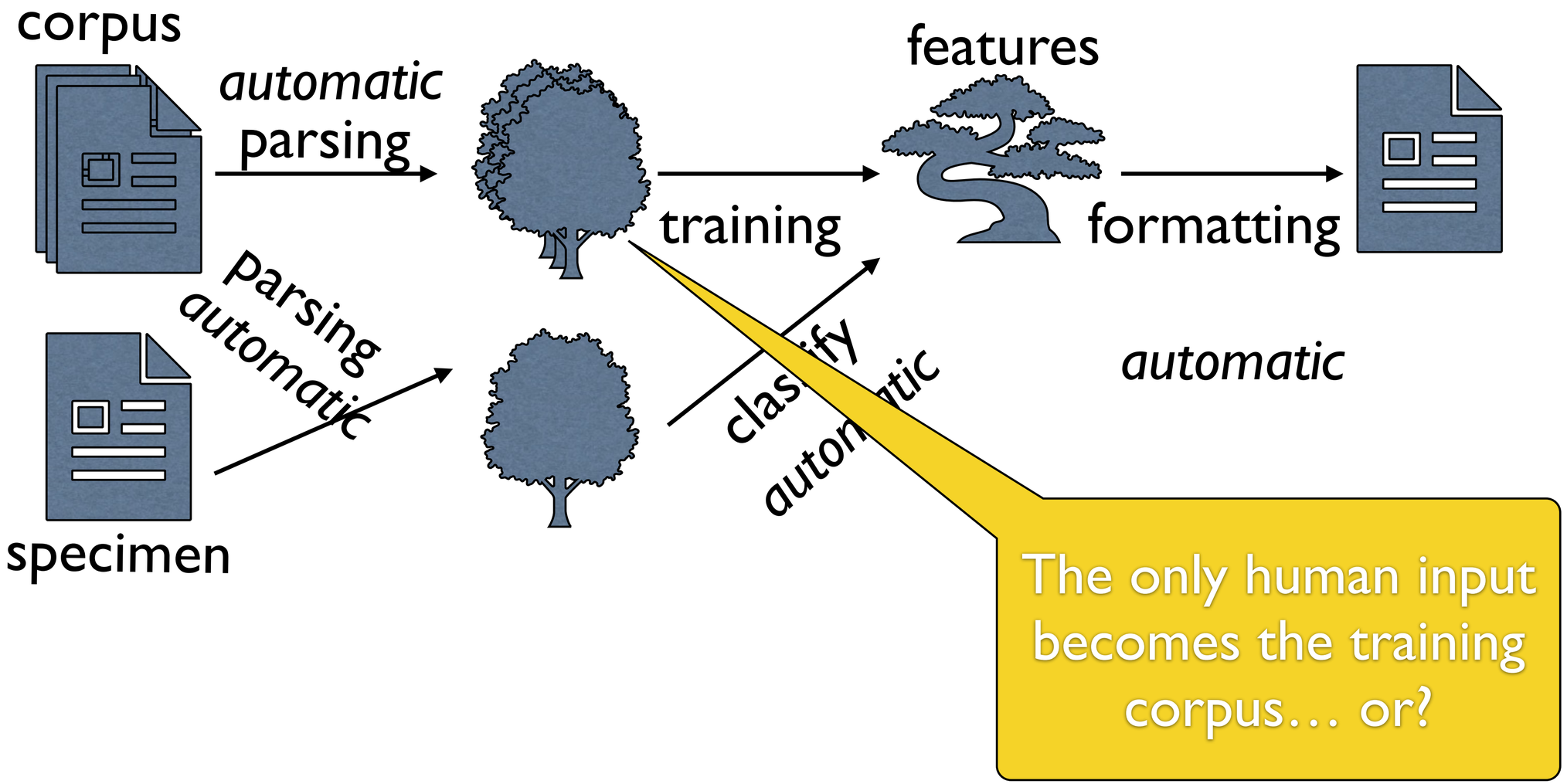
[14] M. G. van den Brand, A. T. Kooiker, J. J. Vinju, and N. P. Veerman. A language independent framework for context-sensitive formatting. In *CSMR 2006*, pages 10–pp. IEEE, 2006.

What if... there were a formatting AI?



- Declarative formatters are still complex
- What if we could configure them “by example”?
- What is the least and simplest AI we can use?
- Model the human (parallel with Turing Machine)
- What do *programmers* do when typing and indenting?
- I.o.w. What are the *mechanics* of formatting?

AI-based formatting



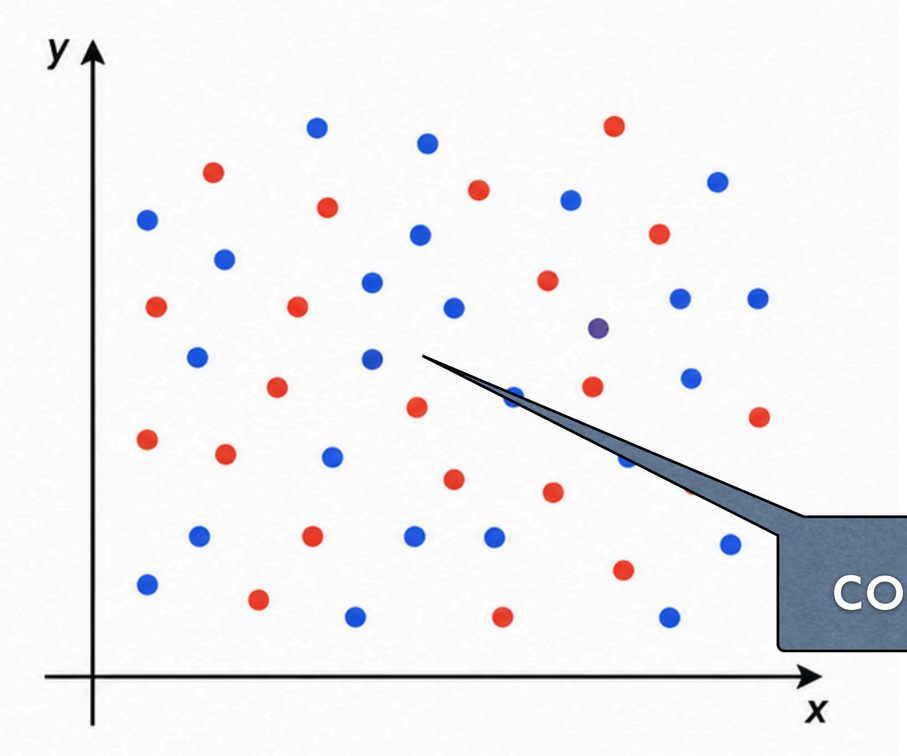
An AI Model for formatting

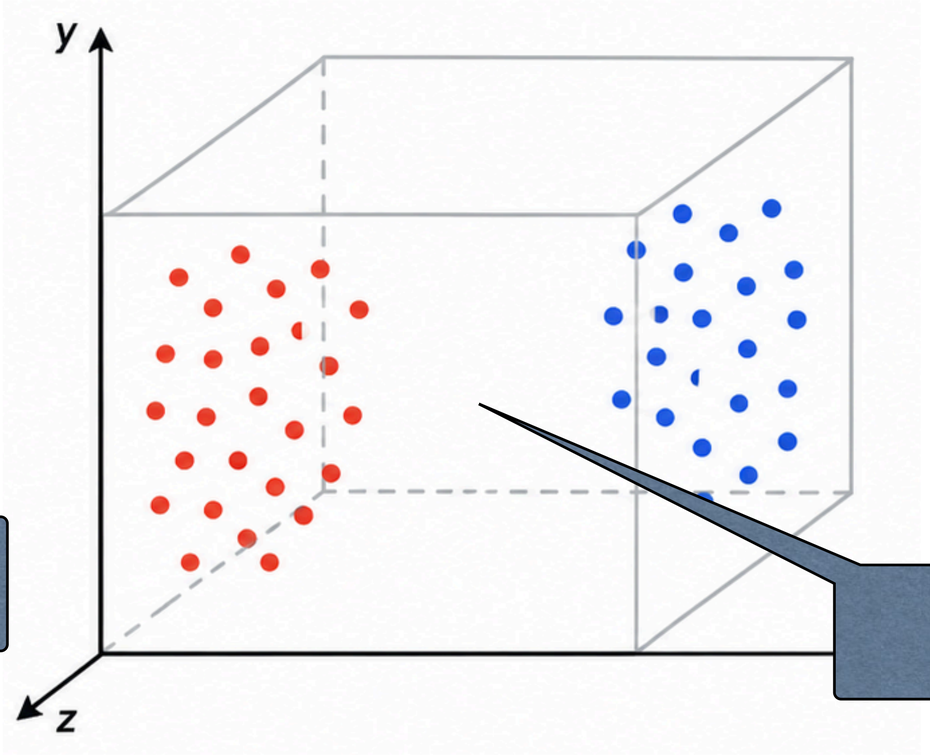
- User is typing...
- After each token
  - newline or not?
  - how many spaces?
- Repeat

“classes” = formatting directives

1. nl: Inject newline
2. sp: Inject space character
3. $(align, t)$: Left align current token with previous token t
4. $(indent, t)$: Indent current token from previous token t
5. $none$: Inject nothing, no indentation, no alignment

A missing feature






```
if (debug)↵
␣␣␣␣println("x");
```

bad layout

```
if (debug)↵
␣␣␣␣{↵
␣␣␣␣␣␣␣␣println("x");↵
␣␣␣␣}↵
```

```
if (debug) {↵
␣␣␣␣println("x")↵
}↵
```

```
if (debug)↵
␣␣␣␣println("x")↵
```

good layout

Features of parse trees

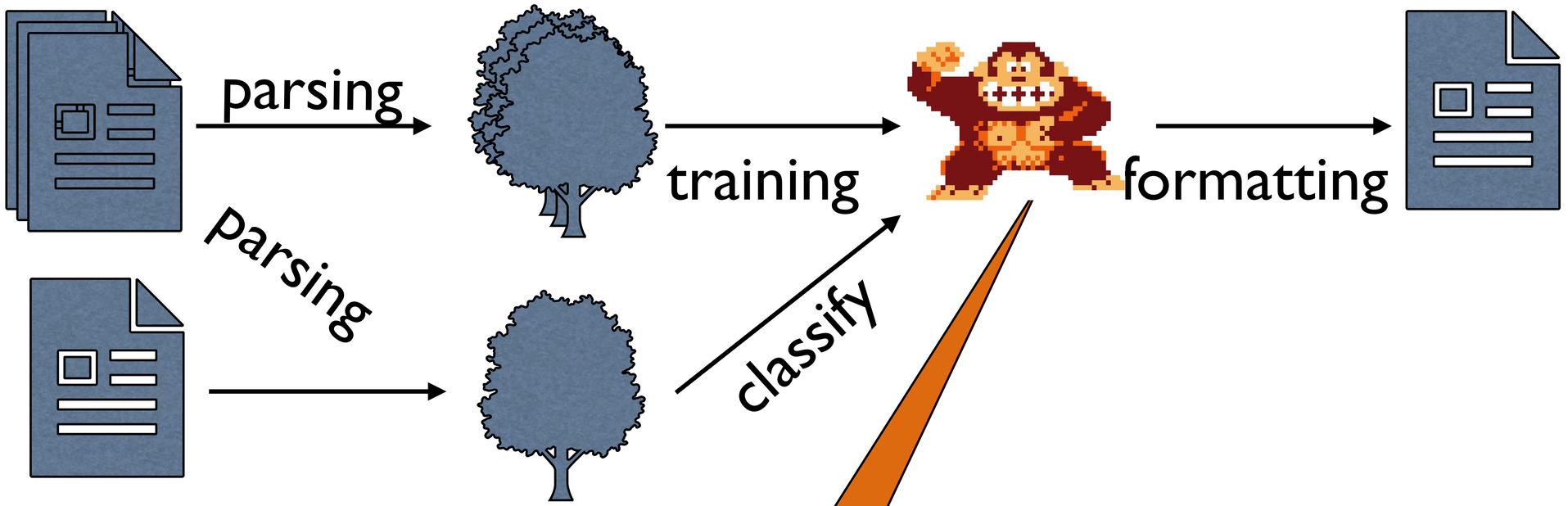
1. t_{i-1}, token type of previous token
2. t_i, token type of current token
3. Is t_{i-1} the first token on a line?
4. Is paired token for t_i the *first* on a line?
5. Is paired token for t_i the *last* on a line?
6. Is $leftancestor(t_i)$ a component of an oversize list?
7. $leftancestor(t_i)$ component type within list from {*prefix token, first member, first separator, member, separator, suffix token*}
8. $childindex(t_i)$
9. $rightancestor(t_{i-1})$
10. $leftancestor(t_i)$
11. $childindex(leftancestor(t_i))$
12. $parent_1(leftancestor(t_i))$
13. $childindex(parent_1(leftancestor(t_i)))$
14. $parent_2(leftancestor(t_i))$
15. $childindex(parent_2(leftancestor(t_i)))$

    ...
20. $parent_5(leftancestor(t_i))$
21. $childindex(parent_5(leftancestor(t_i)))$

Not too specific,
but not too general…
trial and error, and
spreadsheet staring:
i.o.w. expertise on trees

Better add a new “feature”
that distinguises between two classes
of formatted code, than to add an
overly specific constraint.

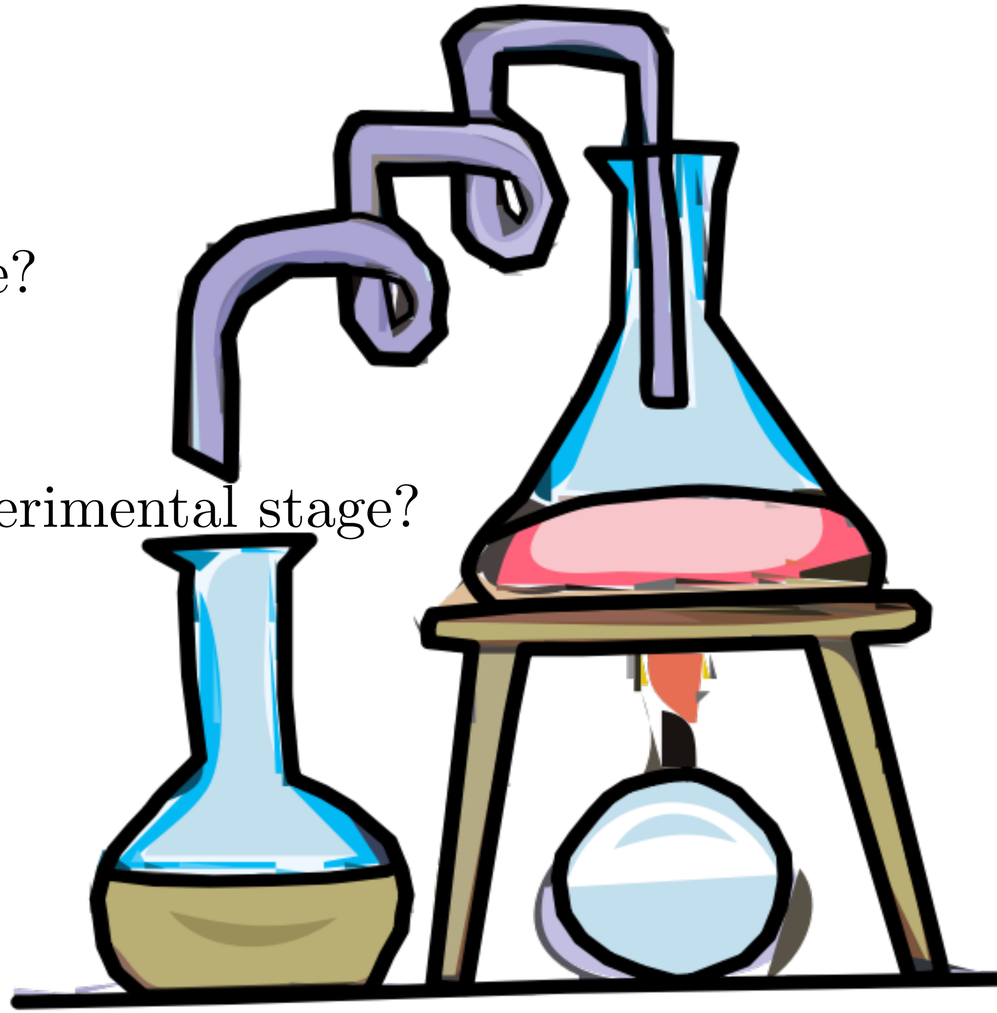
AI-based formatting



The feature set is where the heavy lifting is

Research questions

1. Does the AI format *accurately*?
   A. Is the AI training dependent on grammar shape?
   B. How much training does it need? (2A)
   C. Does it generalize to grammars beyond the experimental stage?
2. Is kNN a viable option, even if simplistic?
   A. How big do the models become? (1B)
   B. What is the right `k`?



Measuring accuracy

- Does the formatted code look like a golden standard?
  - Levenstein distance? Hamming distance?
  - No! One indentation error leads to many *many* edit steps.
- "Misclassification rate" is much better

$$error = \frac{n\_ws\_errors + n\_hpos\_errors}{n\_ws\_decisions + n\_hpos\_decisions}$$

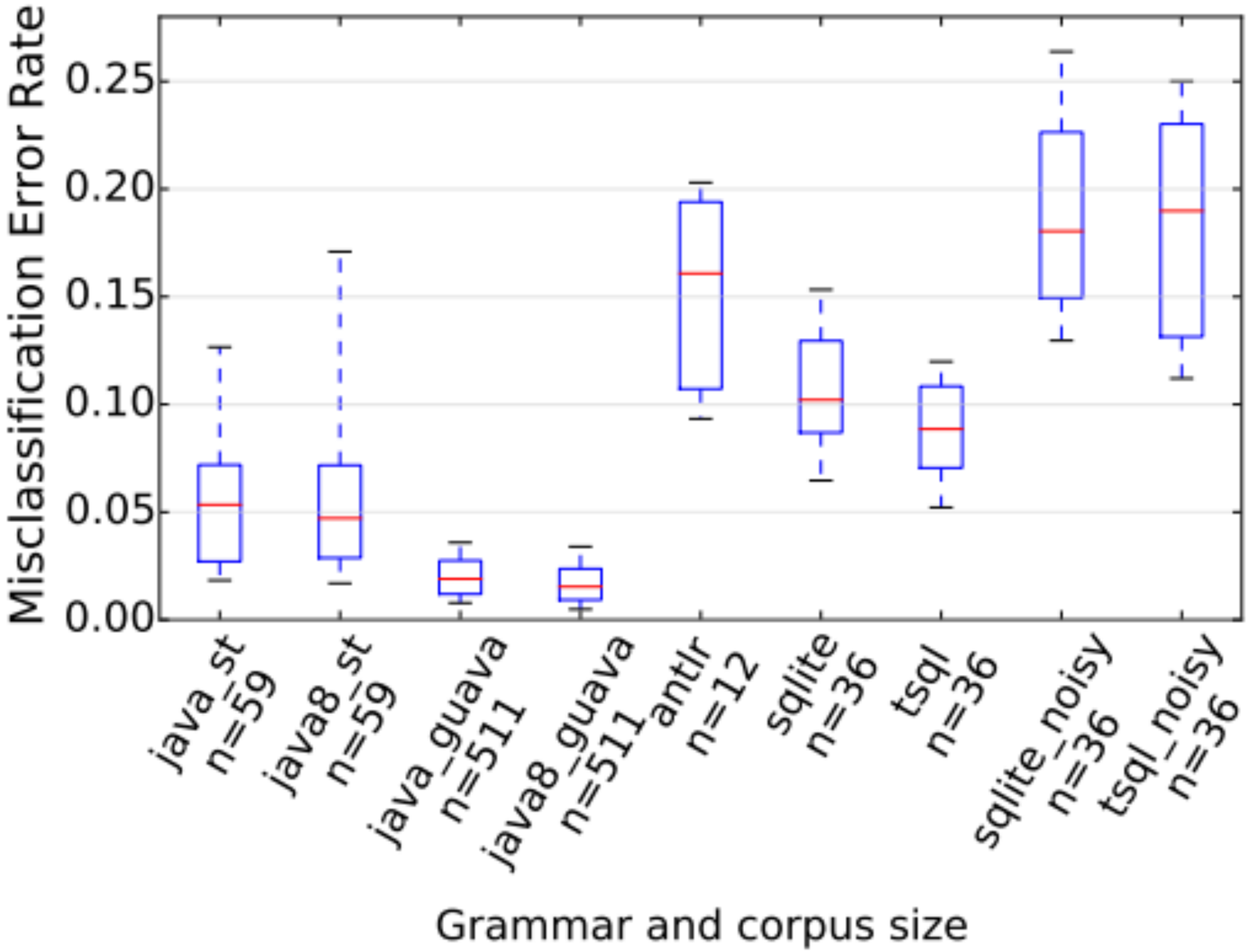


Figure 5. Standard box-plot of leave-one-out validation error rate between formatted document d' and original d.

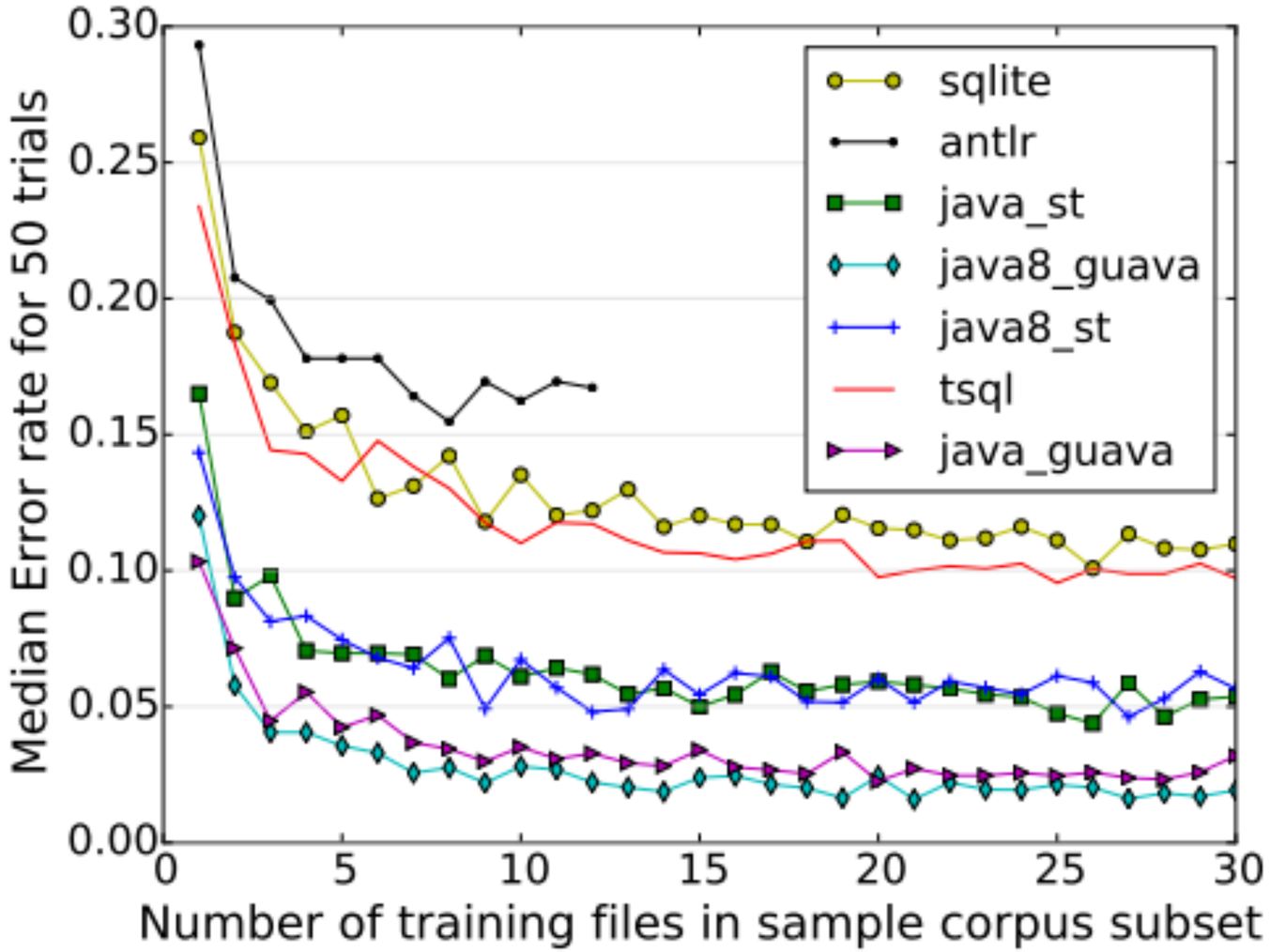


Figure 6. Effect of corpus size on median leave-one-out validation error rate using randomly-selected corpus subsets.

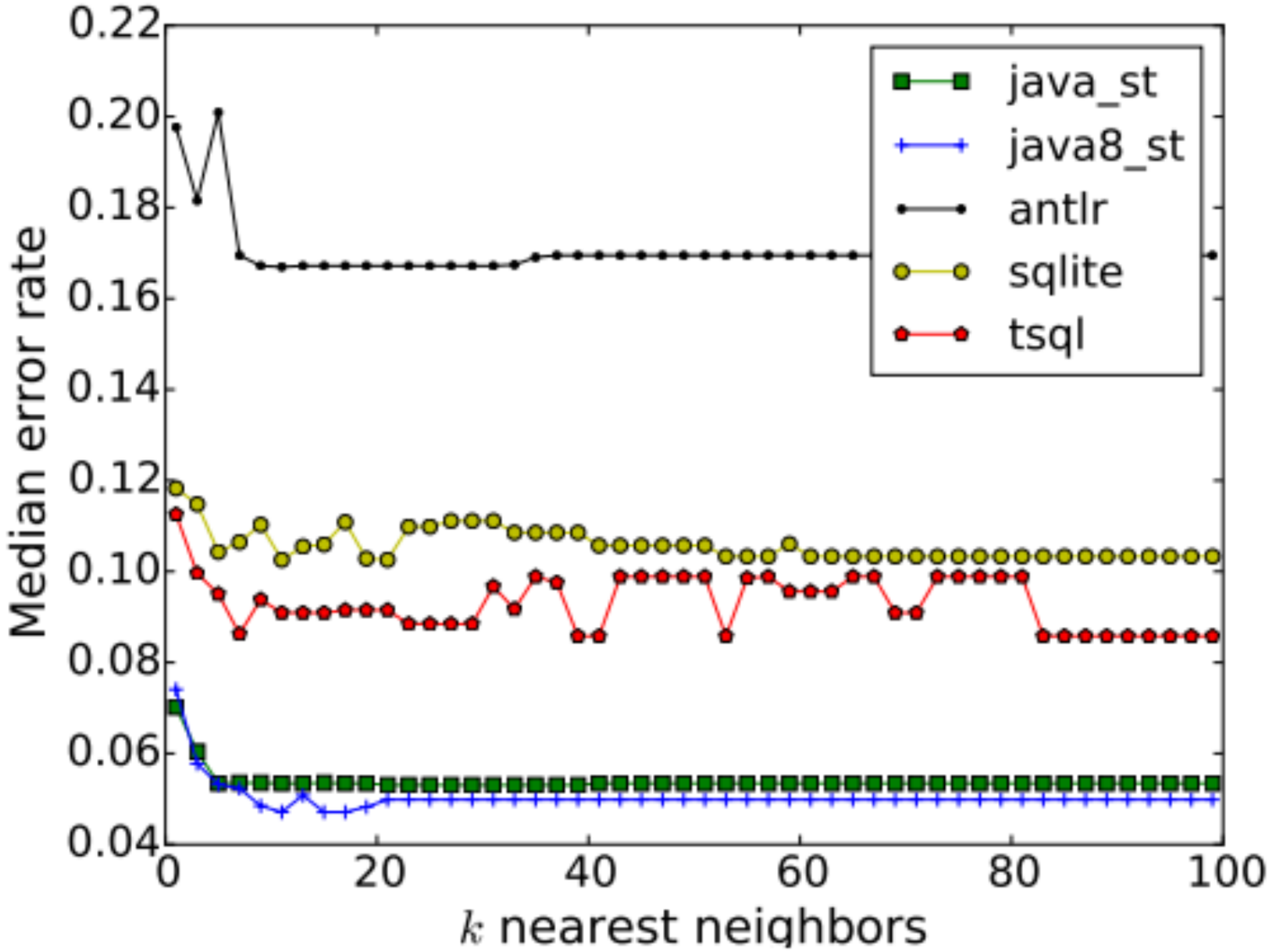


Figure 7. Effect of k on median leave-one-out error rate.

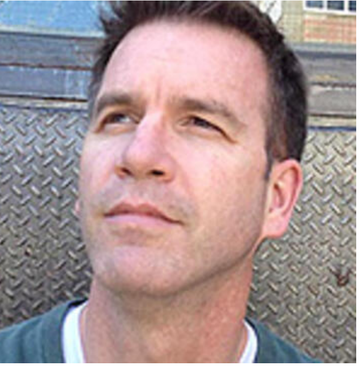
Efficiency

- CodeBuff operates at 2k-5k lines per second
- Worst case formatter is in $O(N^2.\log(N))$
- CodeBuff keeps all the training data in memory (text + trees + classification)
- L1/L2 cache coherence is relatively low
- kNN replaced by “Random Tree/Forest” helps

Reflection

[Escher]

- CodeBuff: very useful and fun tool.
- kNN was *sufficient* and *necessary*
- Finding the right features is a diffucult *"art"*
- Building the right **debugging** tools is *key factor*
- Is 100% accuracy reachable with this approach? unknown...
- Can stream-based AI be ported to box-based AI formatter?



Terence

Thank you!

Jurgen

- Code formatting is important (or `goto fail;`)
- CodeBuff: grammars/parsing meets machine learning
- kNN rocks
  - misclassification rate rocks
  - memory requirements, should be improved.
- Heavy-lifting 21 *discovered* formatting features