

ML09: AI assisted code reviews

Igor Khorlo
PhUSE EU Connect 2019
Amsterdam, The Netherlands

10th–13th November 2019

Agenda

- Motivation, why code reviews are needed?
- Gentle intro into neural networks, machine learning on source code
- Toy model
- Entropy
- Next steps / wrap-up

What is a code review?

- Code review is a software quality assurance (QA) activity in which one or several persons check a program mainly by viewing and reading parts of its source code.
- Review is NOT made by the program developer.
- Google published a document with recommendations for How to do a code review. **PhUSE has a wiki section dedicated to Code Review.**
- Some companies even implemented code reviews as a standard validation procedure for simple SAS programs instead of double programming.

Question to audience

Do you use **code reviews** at work?

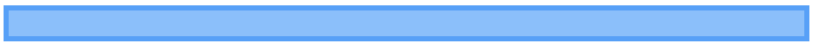
Importance of code reviews

Code Review

Do You Review Code as Part of Your Work?

Hours Per Week Reviewing Code

Yes, because I was told to do so 7.6% 

Yes, because I see value in code review 68.8% 

No 23.6% 

70,390 responses

Most respondents say they review code as part of their work, because it is an important part of the software engineering process. It's not a dominant part of most developers' work week, though. About three-fourths of developers say they spend five hours or less each week reviewing code.

<https://insights.stackoverflow.com/survey/2019#work--code-review>

Importance of code reviews

fact #4

For the 4th straight year, respondents identified **code review** as the **#1 way to improve code quality.**

- State of Code Review 2019



<https://smartbear.com/resources/ebooks/the-state-of-code-review-2019/>

GitHub pull request review

21 tensorflow/python/autograph/operators/py_builtins.py

@@ -384,8 +384,26 @@ def _py_filter(function, iterable):

384 return filter(function, iterable)

385

386

384 return filter(function, iterable)

385

386

387 + def any_(iterable):

388 + if isinstance(iterable, dataset_ops.DatasetV2):

389 + return _tf_dataset_any(iterable)

390 + return _py_any(iterable)

391 +

392 +

393 + # any() operation is essentially a "if first True element exist".

394 + # For that it could be translated to `filter(True)` to filter out

395 + # only `True` element, and then `take(1)`. This works in tf.data

396 + # as tf.data's filter+take is done in pipeline so it will stop

397 + # as soon as `take(1)` returns.

398 + def _tf_dataset_any(iterable):

 **mdanatg** 13 hours ago Contributor

A great addition would be a verification that looks at `iterable.element_spec` and raises an error if the dataset contains anything other than a single `bool` scalar tensor. The error message can be something like this: `in graph mode, the "any" builtin only supports datasets that return bool scalars; got: {} .format(iterable.element_spec)`

 **yongtang** 13 hours ago Author Member

Thanks. The PR has been updated to check and makes sure `element_spec` only consists of `tf.bool`.

Code reviews – back side...

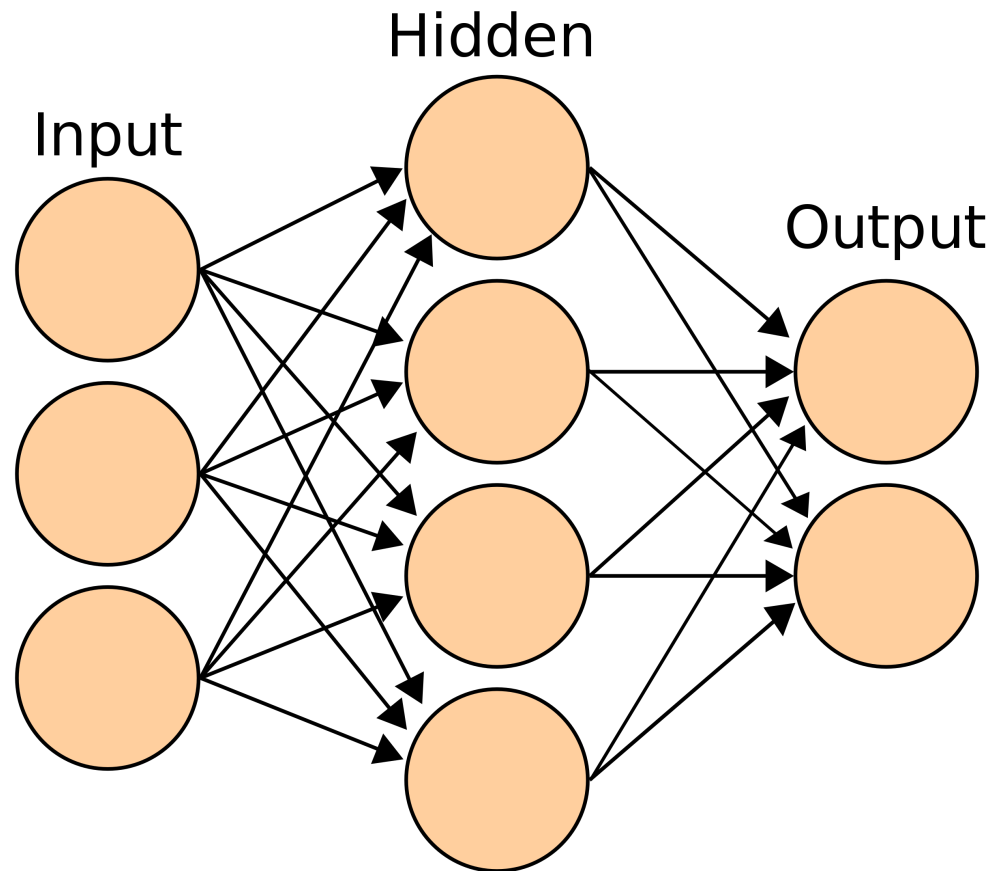


<https://twitter.com/iamdeveloper/status/397664295875805184?lang=en>

Neural networks

- How much data do we have?
95242 LoC, phase 2b study.

```
find . -iname "*.sas" -type f \
| xargs cat \
| sed '/^\s*$/d' \
| wc -l
```
- What is source code?
- How to represent source code as an input?
- What kind of neural network to use?
- How this would help us with code reviews?



By en:User:Cburnett - Own workThis W3C-unspecified vector image was created with Inkscape., CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=1496812>

Representing SAS source code

```
proc means data=ads.ads1;
```

```
  class trt01pn;
```

```
  var age;
```

```
  output out=age_desc
```

```
    n=
```

```
    mean=
```

```
    min=
```

```
    max= / autoname;
```

```
run;
```

```
112 114 111 99 32 109 101 97 110 115 32 100 97 116 97 61
97 100 115 46 97 100 115 108 59 13 10 32 32 32 32 99
108 97 115 115 32 116 114 116 48 49 112 110 59 13 10 32
32 32 32 118 97 114 32 97 103 101 59 13 10 32 32 32
32 111 117 116 112 117 116 32 111 117 116 61 97 103 101 95
100 101 115 99 13 10 32 32 32 32 32 32 32 32 110 61
13 10 32 32 32 32 32 32 32 32 109 101 97 110 61 13
10 32 32 32 32 32 32 32 32 109 105 110 61 13 10 32
32 32 32 32 32 32 32 109 97 120 61 32 47 32 97 117
116 111 110 97 109 101 59 13 10 114 117 110 59 13 10
```

Representing SAS source code on a char level

$$"a" = (1, 0, \dots, 0) \in \mathbb{R}^{26}$$

$$"b" = (0, 1, \dots, 0) \in \mathbb{R}^{26}$$

...

$$"z" = (0, 0, \dots, 1) \in \mathbb{R}^{26}$$

Some data preprocessing to consider

- Remove all comments (pl1, star, macro style comments)
- No datalines content (we don't want NLP inside NLP)
- Remove procedures with `submit;/endsubmit;` blocks.
- SAS Macro...

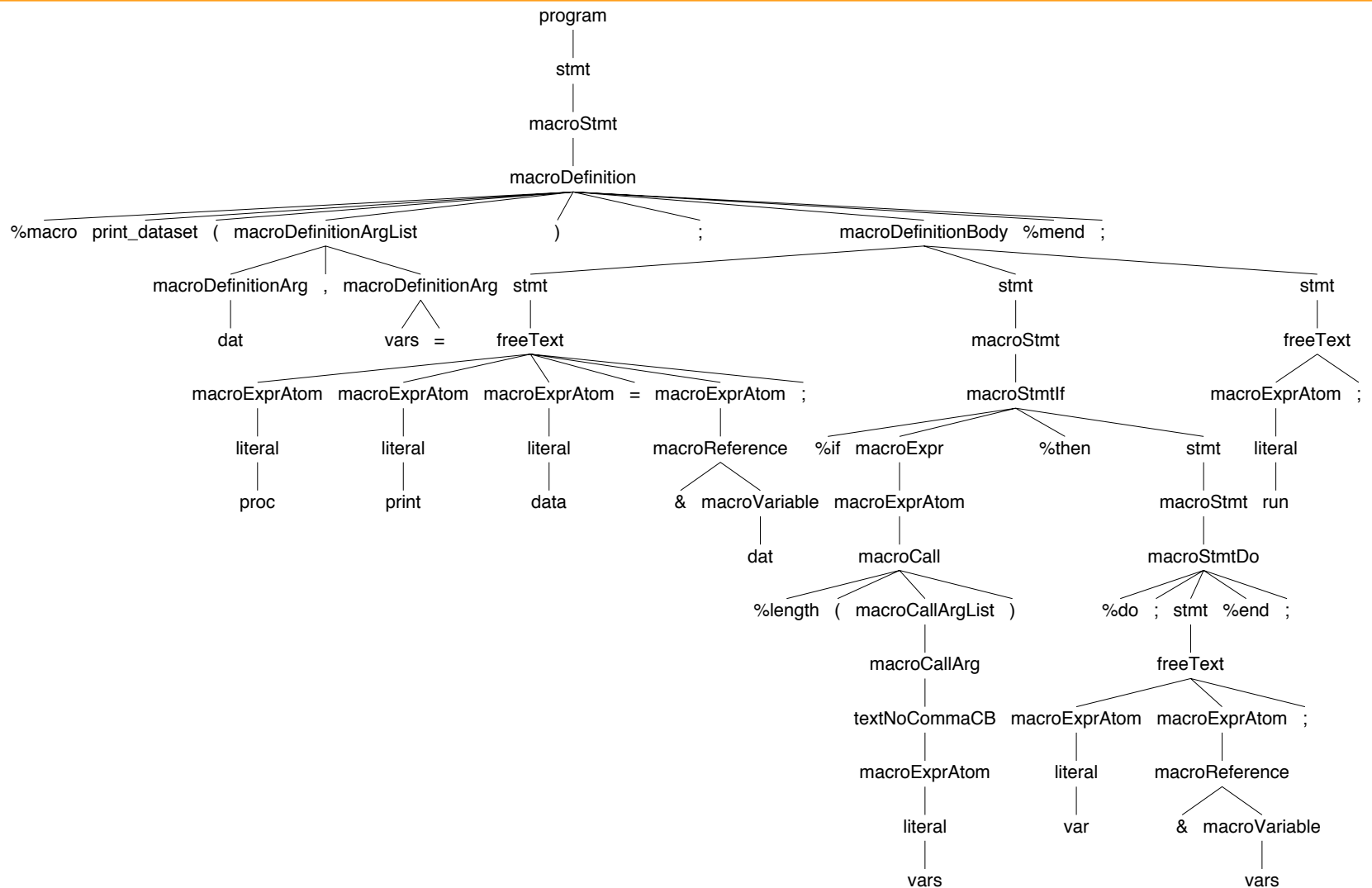
Improving the quality, tokenization of the SAS source code

```
proc means data=ads.ads1;  
  class trt01pn;  
  var age;  
  output out=age_desc  
    n=  
    mean=  
    min=  
    max= / autoname;  
run;
```

Improving the quality, tokenization of the SAS source code

```
proc means data=ads.ads1;  
  class trt01pn;  
  var age;  
  output out=age_desc  
    n=  
    mean=  
    min=  
    max= / autoname;  
run;
```

Parse SAS program and create its parse tree



Tokenizing SAS programs

proc sort data=ae2;	PROC SORT DATA = ID ;
by usubjid aestdtc aeendtc;	BY ID ID ID ;
run;	RUN ;
data mess2;	DATA ID ;
merge mess ae2;	MERGE ID ID ;
by usubjid;	BY ID ;
set ads.adlb(	SET ID . ID (
where=(paramcd='BACT'	WHERE = (ID = <STR>
and visit = 'Screening'))	OP_AND ID = <STR>))
key=keyvar / unique;	KEY = ID / UNIQUE ;
data = 123;	ID = <INT> ;
run;	RUN ;

Vocabulary for tokens

$$\text{DATA} = (1, 0, 0, \dots, 0) \in \mathbb{R}^{57}$$

$$; = (0, 1, 0, \dots, 0) \in \mathbb{R}^{57}$$

$$\text{<ID>} = (0, 0, 1, \dots, 0) \in \mathbb{R}^{57}$$

$$\text{MERGE} = (0, 0, 0, \dots, 0) \in \mathbb{R}^{57}$$

...

$$\text{RUN} = (0, 0, 0, \dots, 1) \in \mathbb{R}^{57}$$

Is the highlighted text correct?

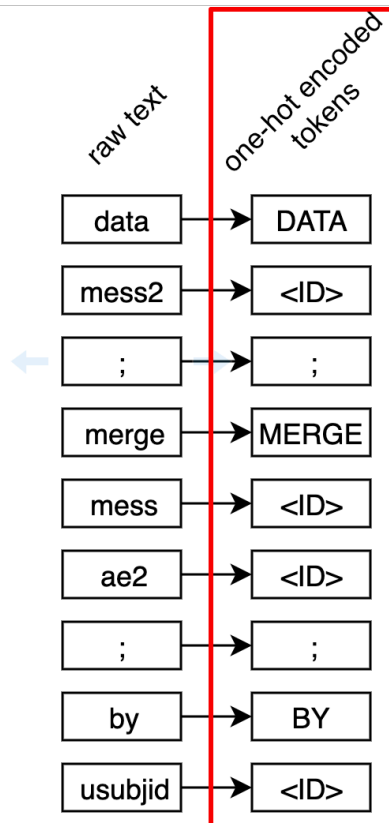
```
data mess2;  
  merge mess ae2;  
  by usubjid ...
```

Starting point is a raw SAS program (textfile)

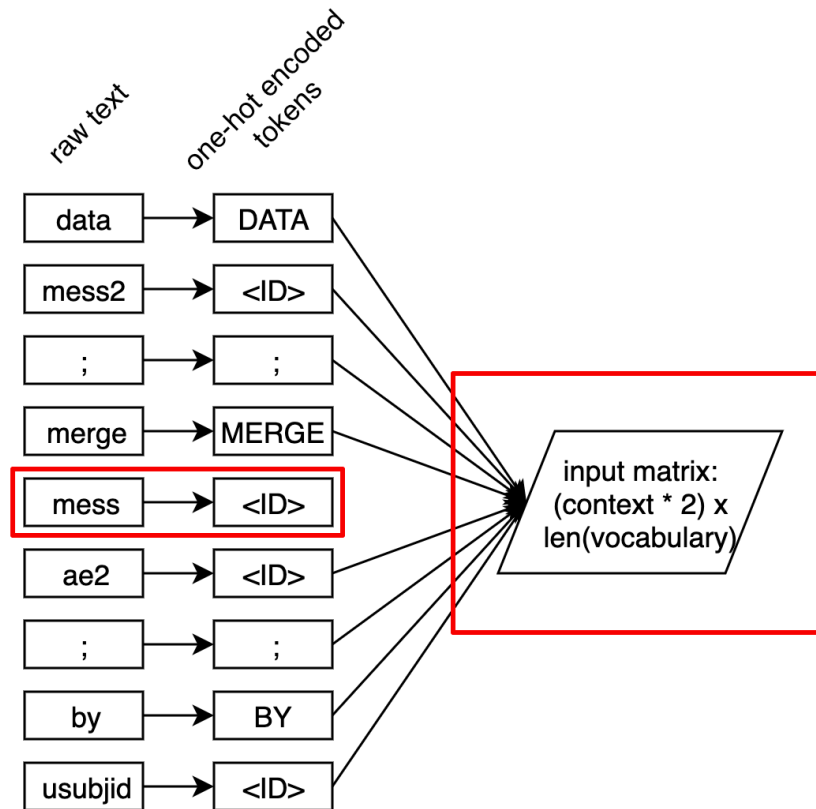
raw text

```
data  
mess2  
;  
merge  
mess  
ae2  
;  
by  
usubjid
```

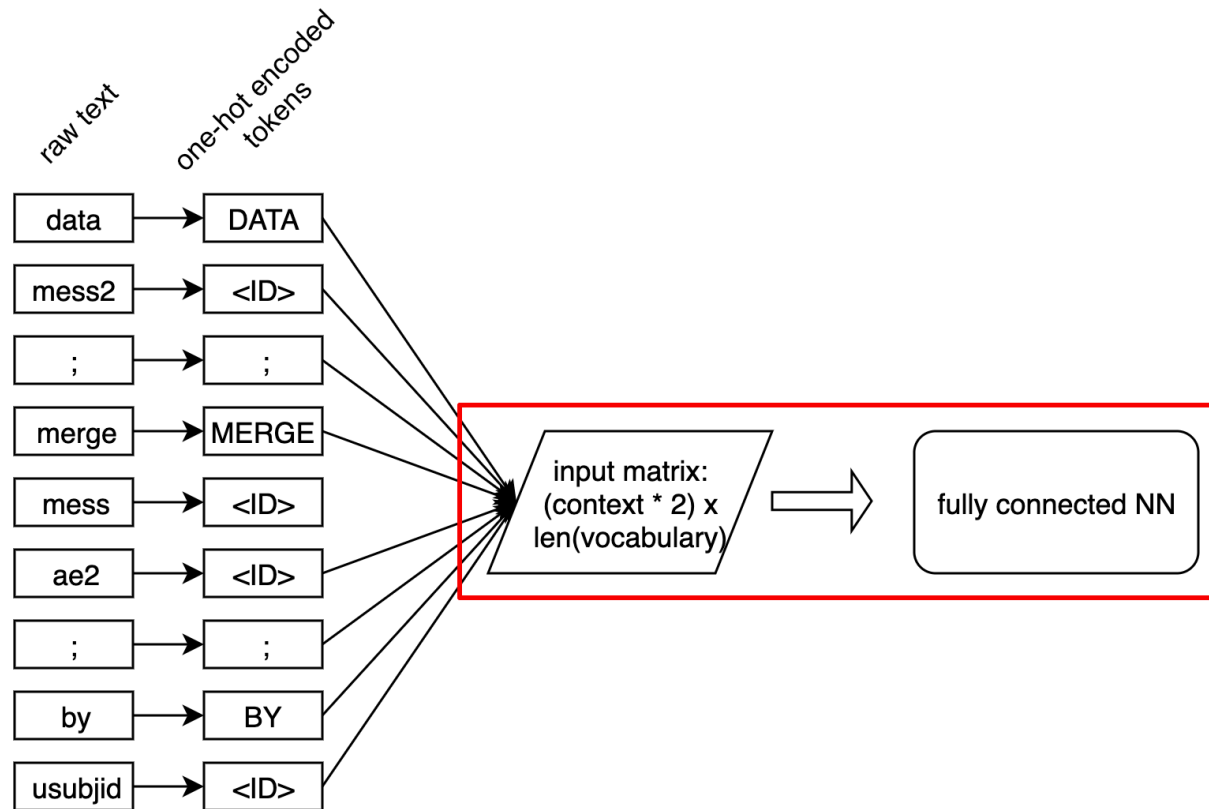
Tokenize and encode as one-hot vectors



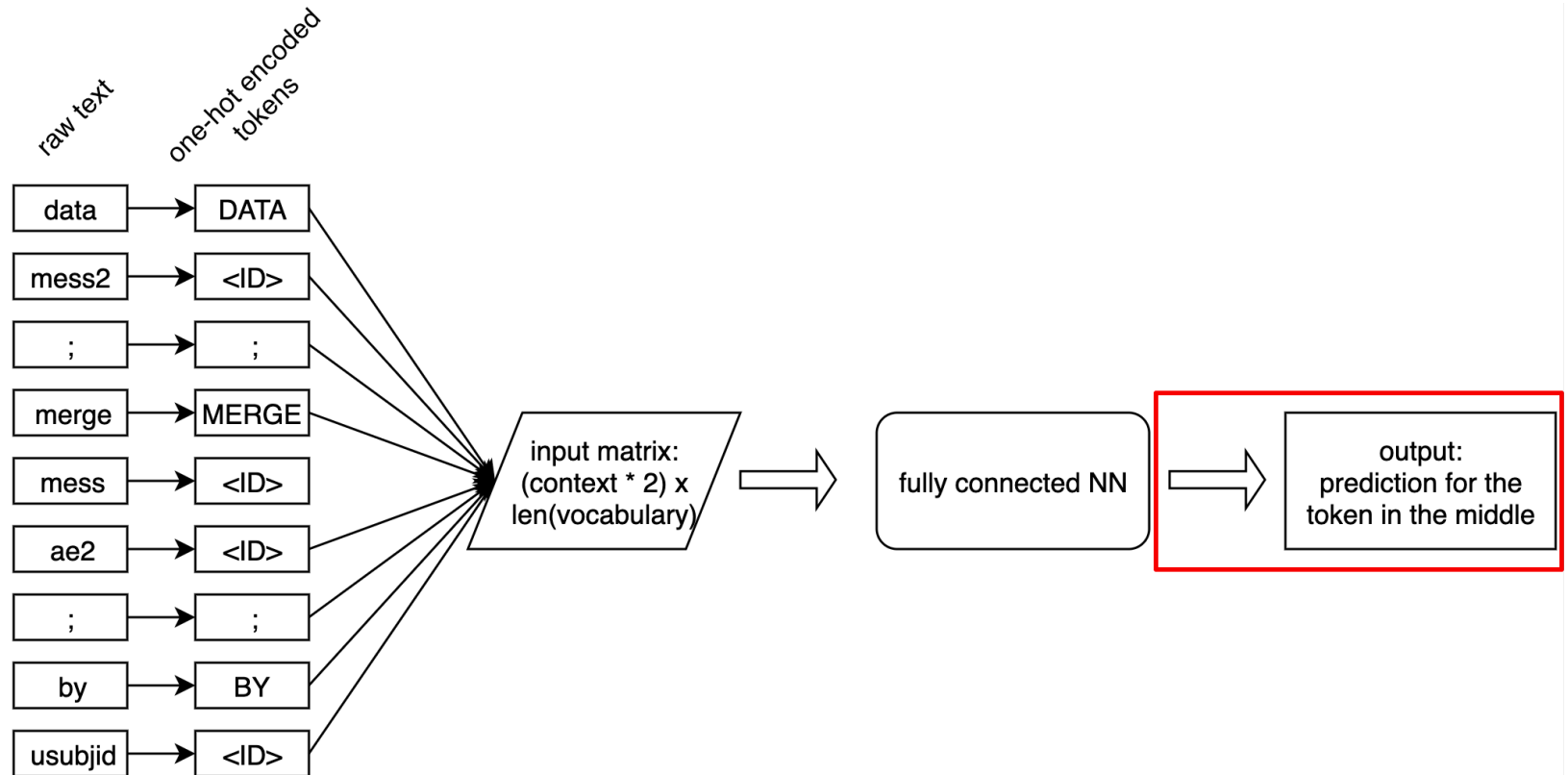
Form input matrix



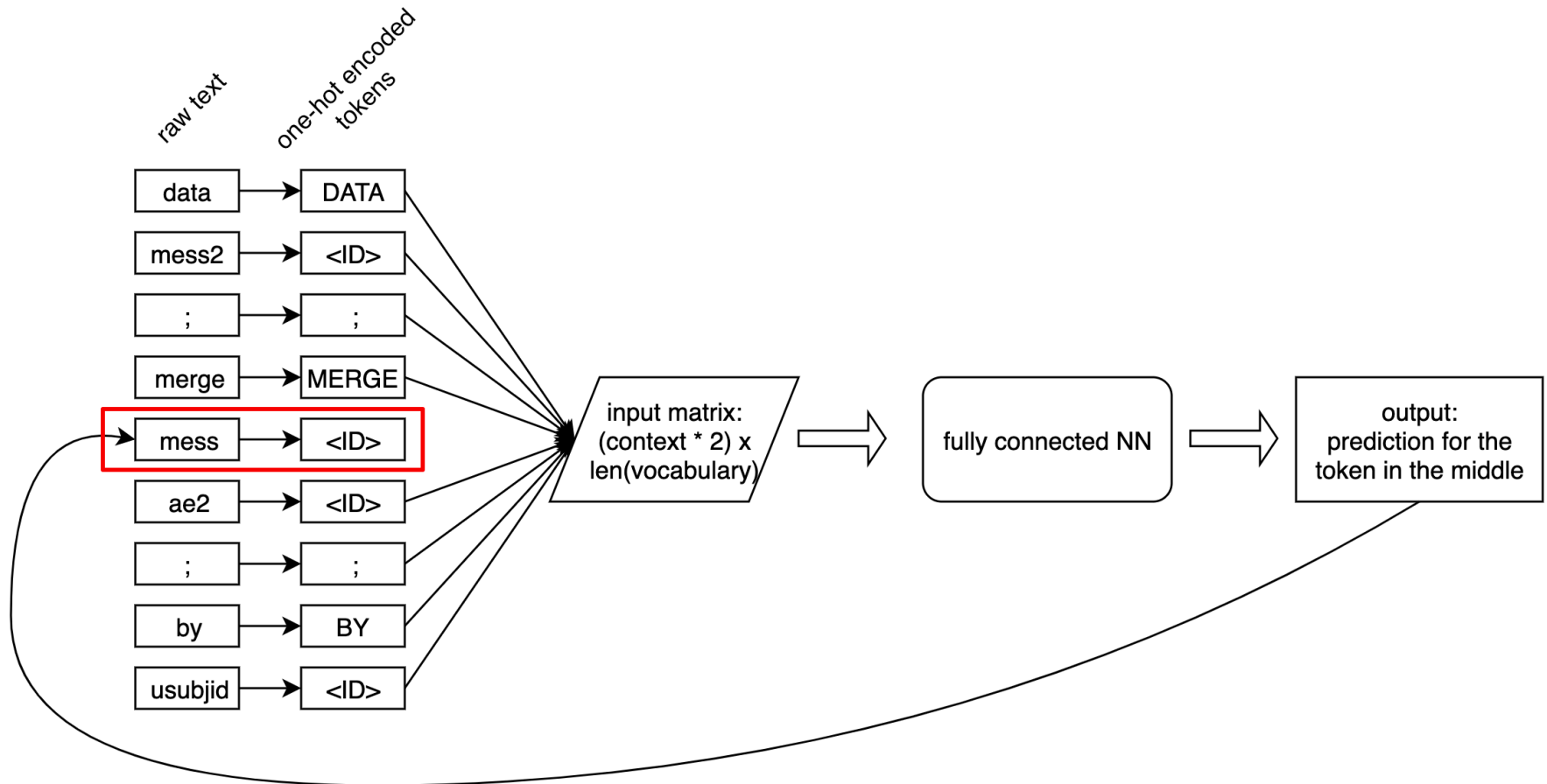
Feed input into FCNN



Get output (vector of vocab size)



Use probability distribution to get a prediction for the middle token



Toy model definition in Keras

```
# define model
model = Sequential([
    Dense(100, activation='relu', input_shape=(context*2, len(vocab))),
    Dense(50, activation='relu'),
    Flatten(),
    Dense(len(vocab), activation='softmax')
])

model.compile(Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

Toy model summary

Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
dense_1 (Dense)	(None, 20, 100)	5800
dense_2 (Dense)	(None, 20, 50)	5050
flatten_1 (Flatten)	(None, 1000)	0
dense_3 (Dense)	(None, 57)	57057
=====		

Total params: 67,907
Trainable params: 67,907
Non-trainable params: 0

Softmax

Output layer is:

```
Dense(len(vocab), activation='softmax')
```

What gives us a vector of `len(vocab)` real numbers from the interval (0, 1). We would treat it as a probability distribution for tokens from the vocabulary:

```
p = model.predict(x) # p.shape = (1, 57)
s = np.sum(p)        # s == 1
print(p)             # [0.0004120028, 0.0011239, ...]
```

Probability distribution for the token in the middle

```
seq = encode('data mess2; merge mess ae2; by usubjid; set ads.adlb(')
x = select_input(seq, 4)
p = model.predict(x) # p.shape = (1, 57)
```

```
# print 5 highest probable tokens
```

```
for times in range(5):
```

```
    i = np.argmax(p) # select highest probability element from p
```

```
    print(code_to_token[], p[0][i])
```

```
    p[0][i] = 0
```

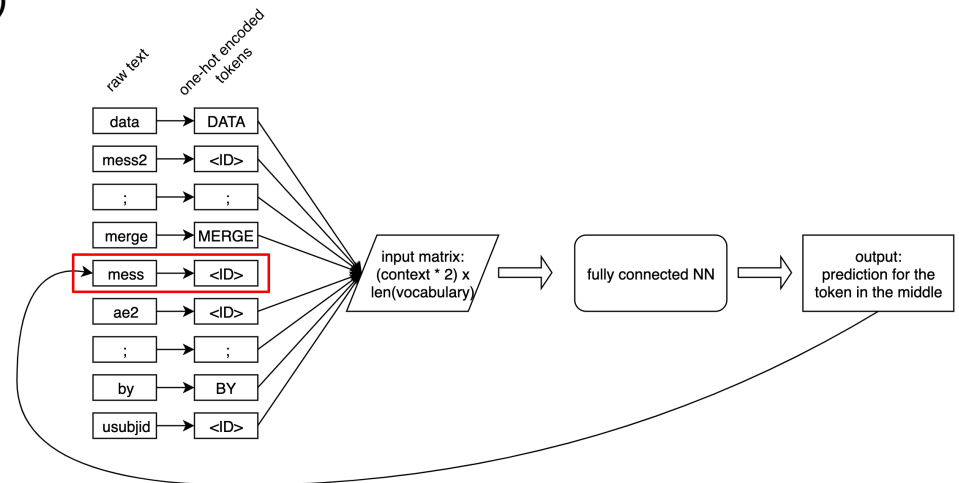
```
# MERGE 0.7913401
```

```
# SET 0.1522897
```

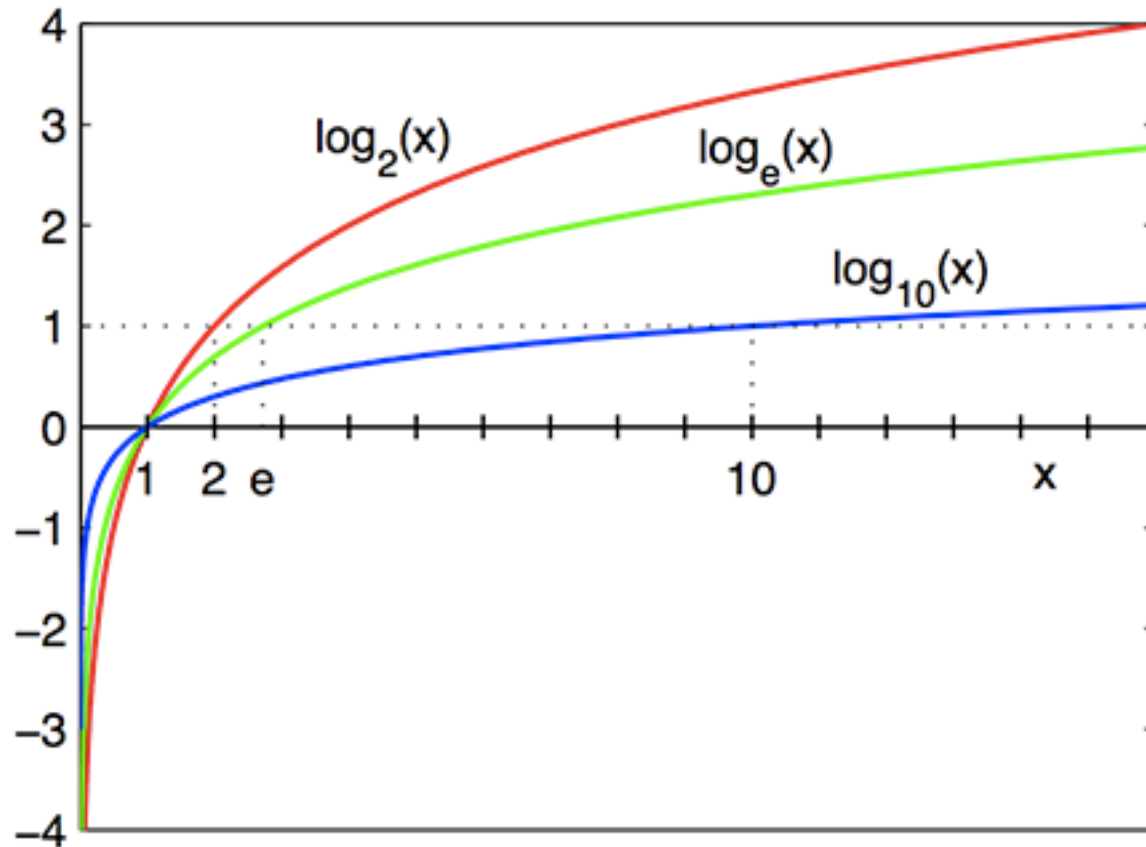
```
# FORMAT 0.013148197
```

```
# INFORMAT 0.00796120195
```

```
# FILE 0.00093120028
```



Logarithm: entropy = $\log(p)$



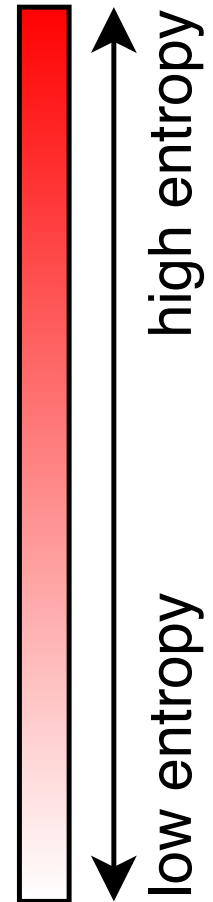
By Richard F. Lyon - made myself, alt version of Logarithm plots.svg with better text, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=13257335>

Entropy heatmap

```
data medi_rdi;  
  length subjid $50 fatestdcd $8 invpeam1 $20;  
  set freqs;  
  by subjid;  
  array cols{*} col1-col8 {1};  
run;
```

```
options compress=binary_2;
```

```
data num;  
  set means_work;  
  blbmin = .;  
run;
```



Next steps

- RNN/LSTM/GRU and their bidirectional analogs
- Transformer architecture – attention models would help to address data/run or do/end blocks.
- Representation for identifiers – AESTDTC is not just an ID.
- Data augmentations – e.g. shuffle options in a procedure.
- Encode variable type information in IDs

Shortening the Distance From Lab to Life™
