BigDat 2020
Ancona, Italy

BIG CODE

(Part 3/3)

Dr. Jayanti Prasad

6th Winter school on Big Data

(13-18 January, 2020)

Ancona, ITALY

PLAN

- Machine Learning ecosystem
- Github data crawling
- Data Preparation
- Model building
- Seq2Seq model
- TreeLSTM
- Discussion and conclusions

MACHINE LEARNING ECOSYSTEM

Training

- Data identification
- Data acquisitions
- Data cleaning
- Feature Engineering
- Resource identification
- Frameworks
- Models
- Training
- Validation & Testing

Compute/data intensive

Inference

- Identifying use/business case
- Deployment

Light Weight

DATA CRAWLING

GITHUB

- Github is one of the biggest open source software hosting platforms.
- Data from Github can be crawled using the REST API provided by Github.
- We will be using 'PyGithub==1.45' which uses Github API to get the list of repositories for a given language with some attributes (minimum number of commits, stars etc.).
- We will be mostly interested in (open source) repos which are not forked and have source code provided.
- See the program github_data_crawler.py for implementation.

GITHUB (JAVA) REPOS

s_no	project_name	num_commits	project_url
0	platform_frameworks_base	377995	https://github.com/aosp-mirror/platform_frameworks_base
1	liferay-portal	290299	https://github.com/liferay/liferay-portal
2	intellij-community	235299	https://github.com/JetBrains/intellij-community
3	android_frameworks_base	104576	https://github.com/dreamcwli/android_frameworks_base
4	consulo	104096	https://github.com/consulo/consulo
5	MPS	81999	https://github.com/JetBrains/MPS
6	zm-mailbox	76307	https://github.com/Zimbra/zm-mailbox
7	frameworks_base	64154	https://github.com/GenetICS/frameworks_base
8	neo4j	59921	https://github.com/neo4j/neo4j
9	idea-community	59329	https://github.com/joewalnes/idea-community
10	ballerina-lang	55934	https://github.com/ballerina-platform/ballerina-lang
11	platform_frameworks_support	52864	https://github.com/aosp-mirror/platform_frameworks_support
12	Osmand	52583	https://github.com/osmandapp/Osmand
13	openmicroscopy	47209	https://github.com/openmicroscopy/openmicroscopy
14	packages_apps_Settings	44854	https://github.com/AOKP/packages_apps_Settings
15	idea2	43586	https://github.com/jexp/idea2
16	elasticsearch	42894	https://github.com/elastic/elasticsearch
17	opennms	42685	https://github.com/OpenNMS/opennms
18	android_packages_inputmethods_LatinIME	39120	https://github.com/CyanogenMod/android_packages_inputmethods_LatinIME
19	packages_inputmethods	39102	https://github.com/SlimRoms/packages_inputmethods_LatinIME
20	fenixedu-academic	37809	https://github.com/FenixEdu/fenixedu-academic

Some of the Big Github Java repos

GIT VERSION CONTROL SYSTEM

- Git is a distributed version control system (vcs) which was developed by Linus Torvalds for linux kernel development.
- We ‘clone’ remote git repos on a local system and modify those and ‘push’ those to the remote system.
- All the code changes on git repo are in the form of ‘commits’.
- A single git commit may have multiple file changes (add, delete, modified).
- A single file may have multiple code ‘hunks’ changed.
- There is associated information (‘commit message’, ‘author’, ‘date’..) with every commit.

GIT DIFF

Multiple parent problem resolved

master

Browse files

jayanti-prasad committed on 8 Nov 2019 — 1 parent f811c22 — commit 733b2479b38401345883ed46c9a6380e3f6d53a2

Showing 2 changed files with **9 additions** and **6 deletions**.

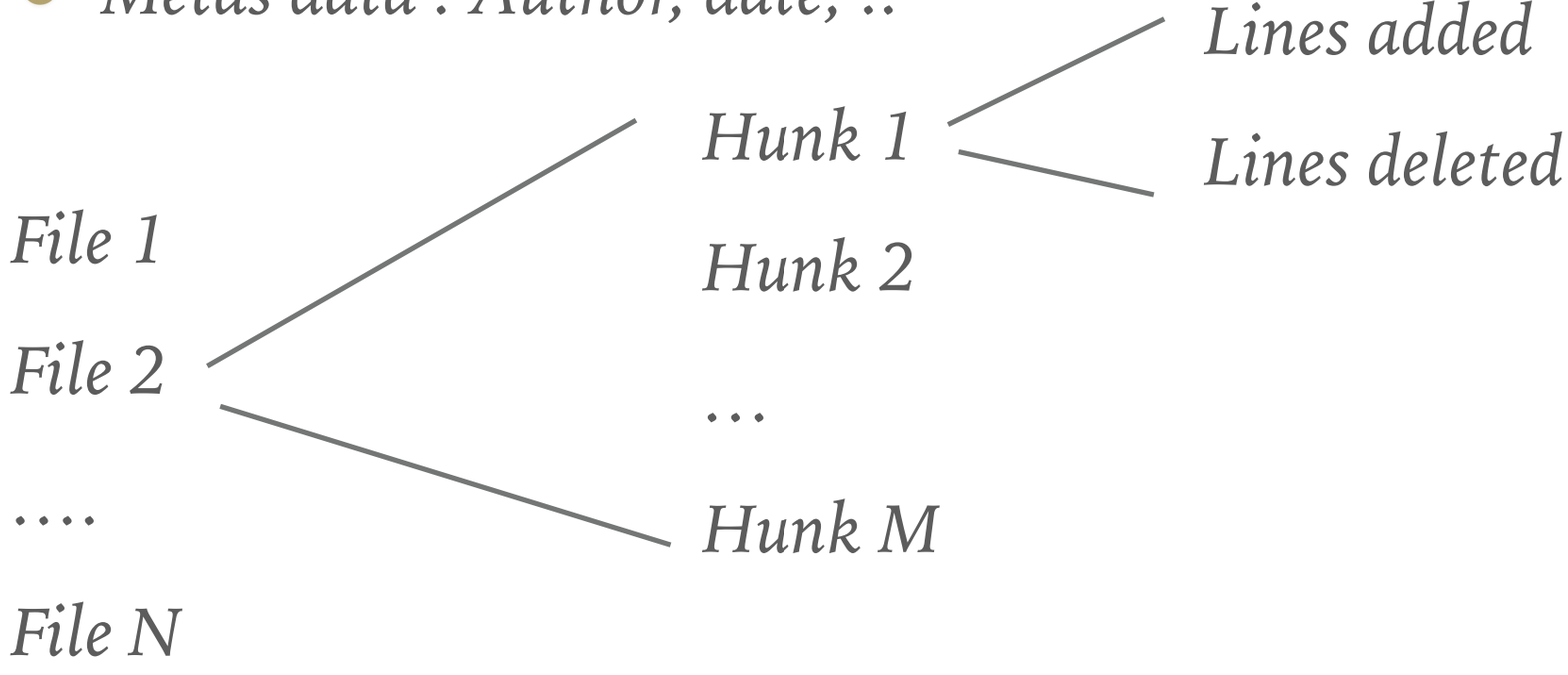
Unified | Split

15 big_code_ast_model.py

```
@@ -42,6 +42,7 @@ def __init__(self, source_file):
42 42            self.nodes = []
43 43            self.nmap = {}
44 44            self.anytree = None
   45 +          self.visited = []
45 46            self.__process__(tree, tree)
46 47            self.__node_mapping__()
47 48            self.anytree = self.__get_any_tree__()
@@ -53,12 +54,14 @@ def __process__(self, parent, tree):
53 54            node_properties['hash'] = get_hash(node_properties)
54 55            node_properties['parent'] = get_hash(get_node_properties(parent))
55 56            node_properties['id'] =  self.id
56    -          self.nodes.append(node_properties)
57    -          self.id = self.id + 1
58    -          num_children = len(tree.children)
59    -          for i in range(0, num_children):
60    -               if '@type' in tree.children[i].get_dict():
61    -                    self.__process__(tree, tree.children[i])
   57 +          if node_properties['hash'] not in self.visited :
```

ANATOMY OF A GIT COMMIT

- Commit id (hash)
- *Commit message*
- *Metas data : Author, date, ..*



- *Parents*
- *Children*

GIT COMMIT PROCESSING

Program : git_tester.py

```
 1 import os
 2 import re
 3 import sys
 4 import git
 5 from unidiff import PatchSet
 6
 7 if __name__ == "__main__":
 8
 9    repo = git.Repo(sys.argv[1])
10    commits = list(repo.iter_commits())
11
12    for i in range(len(commits)):
13         diff = repo.git.diff(commits[i].hexsha, commits[i].hexsha+'^')
14         patch_set = PatchSet(diff)
15         for p in patch_set:
16              if p.is_modified_file:
17                   try:
18                        if os.path.basename(p.path).split('.')[1] == 'java':
19                             source_file = re.sub('^a\/', '', p.source_file)
20                             target_file = re.sub('^b\/', '', p.target_file)
21                             curr_code = repo.git.show('{}:{}'.format(commits[i].hexsha, source_file))
22                             prev_code = repo.git.show('{}:{}'.format(commits[i].hexsha+'^', target_file))
23                             for h in p:
24                                l1, d1 =  h.target_start,  h.target_length
25                                l2, d2 =  h.source_start, h.source_length
26                                print(commits[i].hexsha, commits[i].summary, source_file, l1, d1, l2, d2)
27                   except:
28                        pass
29
```

GIT DIFF DATA

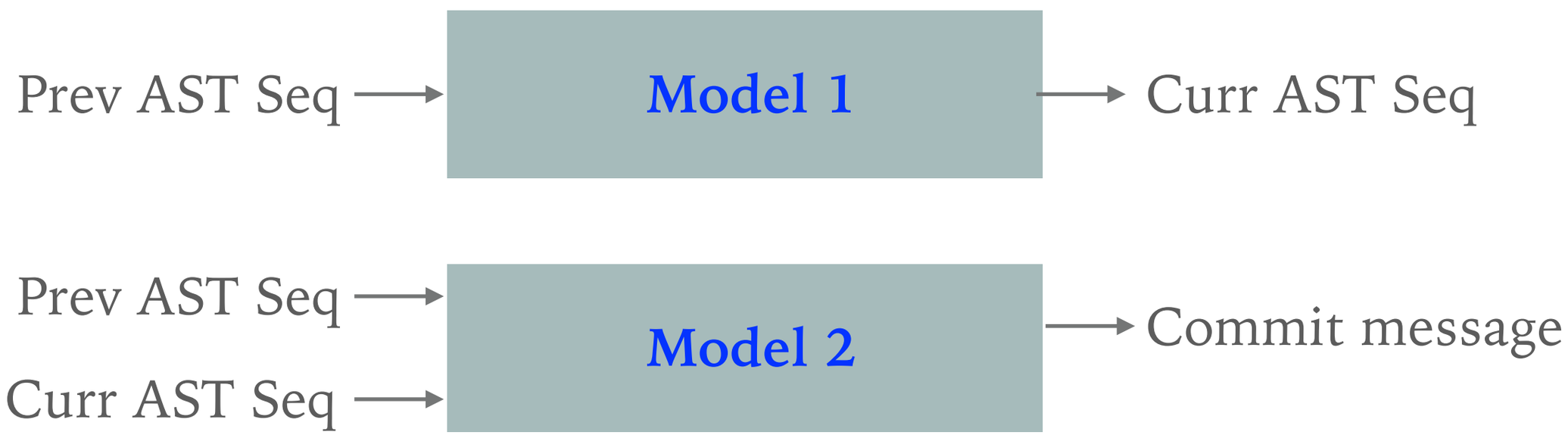
We can process a git hub repo & can create data in a csv form with the following columns :

```
>>> df=pd.read_csv("apache_kafka_data.csv")
>>> df.shape
(368, 13)
>>> d=df.loc[0]
>>> d
Unnamed: 0                                                      0
project_name                                                kafka
commit_id                5d0c2f3b2ad3cb12c8727b4fbf3a64c25ece6209
commit_msg      MINOR: Add validation in MockAdminClient for r...
file_name       clients/src/test/java/org/apache/kafka/clients...
prev_raw        \n175:                  continue;\n176:         ...
curr_raw        \n175:                  continue;\n176:         ...
prev_ast        \n175:java:ContinueStatement\n177:java:Variabl...
curr_ast        \n175:java:ContinueStatement\n177:java:Variabl...
prev_start                                                    175
prev_length                                                     6
curr_start                                                    175
curr_length                                                    11
Name: 0, dtype: object
>>>
```

A typical data unit (row)

SEQ2SEQ MODEL

- For supervised learning with need input data (source) & output data (target).
- Neural network allow multiple inputs & outputs so we can select any number of columns as inputs and any number of columns as output.



[Sutskever et. al. (2014)]

SEQSEQ MODEL-1

Driver Program

```
56 if __name__ == "__main__":
57     parser = argparse.ArgumentParser(description='cmod')
58     parser.add_argument('-itn', '--num_input_tokens',type=int,
59         help='Number of input tokens', required=True)
60     parser.add_argument('-otn', '--num_output_tokens',type=int,
61         help='Number of output tokens', required=True)
62     parser.add_argument('-isl', '--len_input_seq',type=int,
63         help='Length of input sequence', required=True)
64     parser.add_argument('-osl', '--len_output_seq',type=int,
65         help='Length of input sequence', required=True)
66     parser.add_argument('-ldm', '--latent_dim',type=int,
67         help='Latent dimension', required=True)
68     parser.add_argument('-n', '--epoch', type=int,
69         help='Epochs', required=True)
70
71     cfg = parser.parse_args()
72
73     encoder_model, encoder_inputs, encoder_outputs  = get_encoder_model (cfg)
74
75     print(encoder_model.summary())
76     utils.plot_model(encoder_model, to_file = "encoder.png")
77
78     model = get_model (cfg, encoder_inputs, encoder_outputs)
79     print(model.summary())
80     utils.plot_model(model, to_file = "model.png")
81
82
```

SEQ2SEQ MODEL

Encoder Model

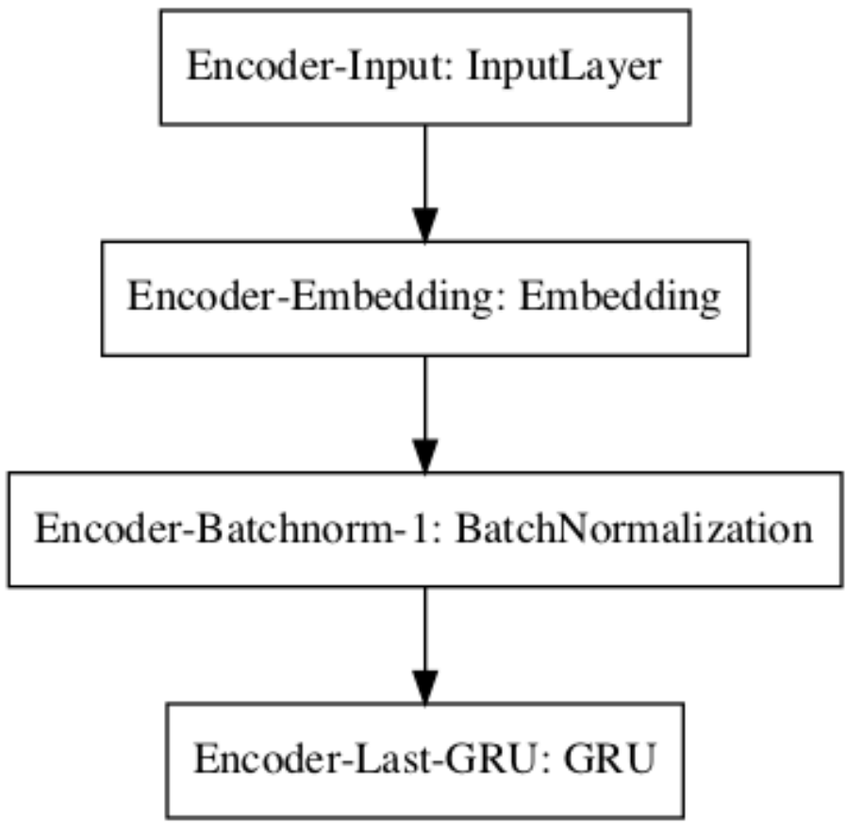
```
 1 import sys
 2 import argparse
 3 import tensorflow.compat.v1.keras.layers as layers
 4 import tensorflow.compat.v1.keras.models as models
 5 import tensorflow.compat.v1.keras.utils  as utils
 6 import tensorflow.compat.v1.keras.optimizers as optimizers
 7 import tensorflow.compat.v1.keras.callbacks as callbacks
 8
 9 def get_encoder_model (cfg):
10     encoder_inputs  = layers.Input(shape=(cfg.len_input_seq,),
11         name='Encoder-Input')
12
13     x = layers.Embedding(cfg.num_input_tokens, cfg.latent_dim,
14         name='Encoder-Embedding', mask_zero=False) (encoder_inputs)
15
16     x = layers.BatchNormalization(name='Encoder-Batchnorm-1')(x)
17
18     _, state_h = layers.GRU(cfg.latent_dim, return_state=True,\
19         name='Encoder-Last-GRU')(x)
20
21     encoder_model = models.Model(inputs=encoder_inputs,
22         outputs=state_h, name='Encoder-Model')
23
24     encoder_outputs = encoder_model(encoder_inputs)
25
26     return encoder_model, encoder_inputs, encoder_outputs
```

SEQ2SEQ MODEL

Encoder Decoder Model

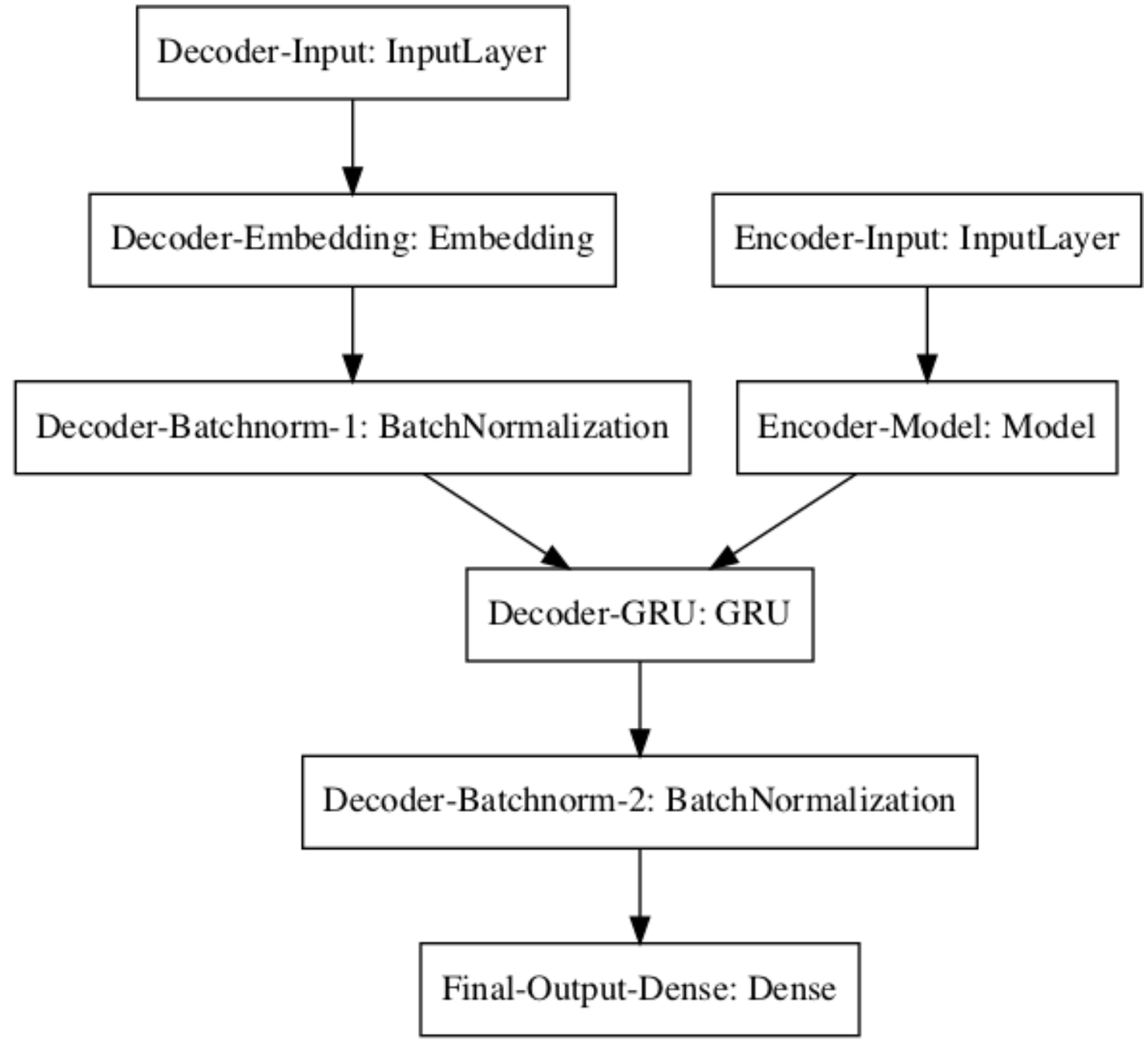
```
29 def get_model (cfg, encoder_inputs, encoder_outputs):
30 
31     decoder_inputs = layers.Input(shape=(None,),
32         name='Decoder-Input')  # for teacher forcing
33 
34     dec_emb = layers.Embedding(cfg.num_input_tokens, cfg.latent_dim,
35         name='Decoder-Embedding', mask_zero=False)(decoder_inputs)
36 
37     dec_bn = layers.BatchNormalization(name='Decoder-Batchnorm-1')(dec_emb)
38 
39     decoder_gru = layers.GRU(cfg.latent_dim, return_state=True,
40         return_sequences=True, name='Decoder-GRU')
41 
42     decoder_gru_output, _ = decoder_gru(dec_bn, initial_state=encoder_outputs)
43 
44     x = layers.BatchNormalization(name='Decoder-Batchnorm-2')(decoder_gru_output)
45     decoder_dense = layers.Dense(cfg.num_output_tokens,
46         activation='softmax', name='Final-Output-Dense')
47 
48     decoder_outputs = decoder_dense(x)
49 
50     model = models.Model([encoder_inputs, decoder_inputs], decoder_outputs)
51 
52     return model
```

MODEL ARCHITECTURE



Encoder Model

Sequence to vector utility



Encoder-Decoder Model

Sequence to sequence utility

FITTING THE MODEL

```
56 def fit_model (cfg, model, X, Y):
57
58     model.compile(optimizer=optimizers.Nadam(lr=0.01),
59           loss='sparse_categorical_crossentropy',metrics=['acc'])
60
61     encoder_input_data = X
62     decoder_input_data = Y[:, :-1]
63     decoder_output_data = Y[:, 1:]
64
65     history =   model.fit([encoder_input_data,
66                 decoder_input_data],  np.expand_dims(decoder_output_data, -1),
67                 batch_size =100,
68                 epochs = cfg.epoch, validation_split = 0.12)
69
70     return history
```

```
100     # create fake data
101     X  =  np.random.randint(cfg.num_input_tokens,
102        size=(1000, cfg.len_input_seq))
103     Y  =  np.random.randint(cfg.num_output_tokens,
104        size=(1000, cfg.len_output_seq))
105
106     print(X.shape)
107     print(Y.shape)
108
109     # fit the model
110     h = fit_model (cfg, model, X, Y)
```

INFERENCE

- Read the input trained model & encoder model.
- Build the decoder model.
- Get the encoder ‘state’ for a given input sequence.
- With decoder output as ‘start’ token and encoder input state predict the 1st token and ‘state’.
- Predict the next ‘token’ using the 1st token & state and update state.
- Keep iterating till ‘stop’ token is predicted or maximum sequence length is reached.

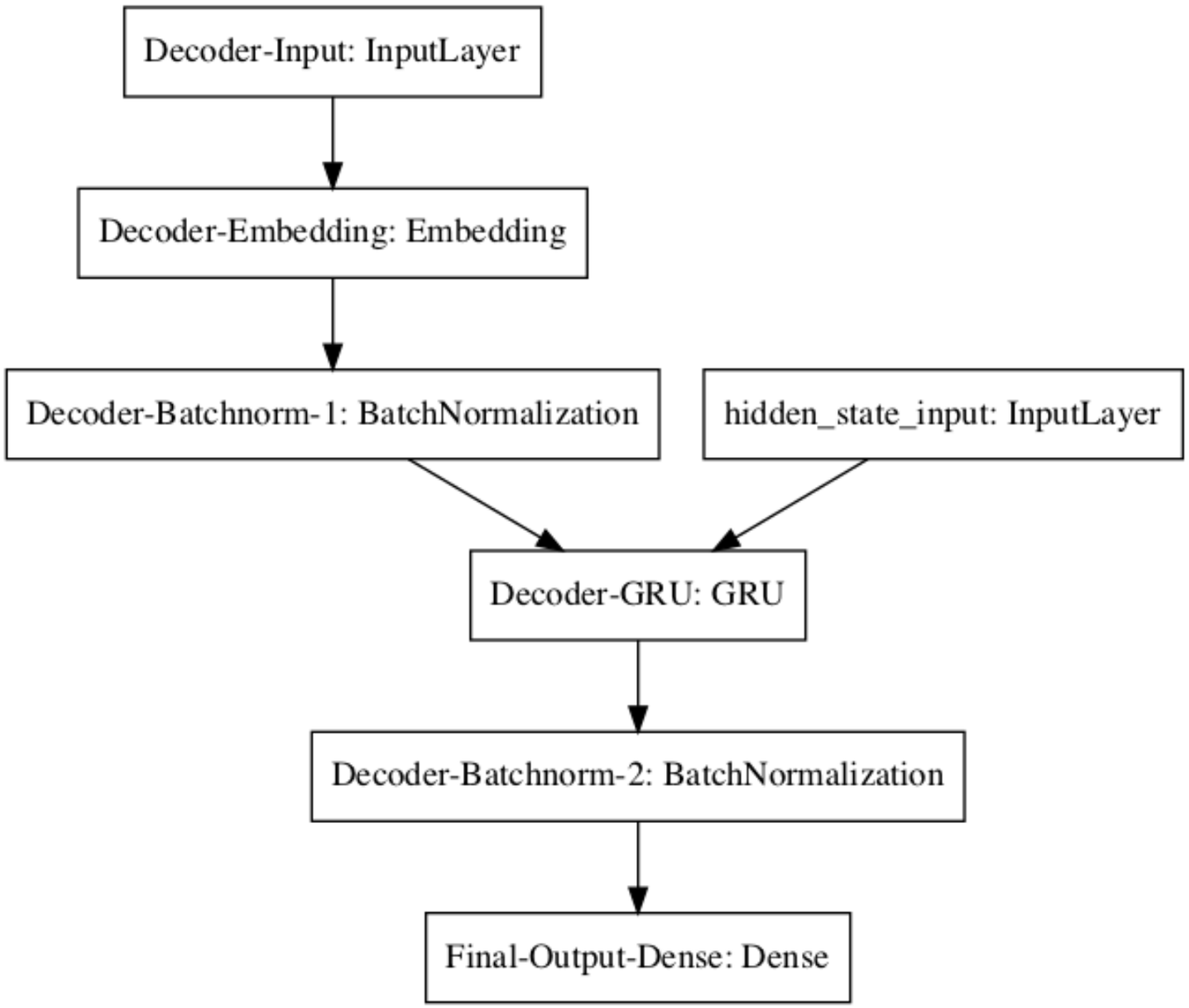
INFERENCE

```
10 def get_decoder_model (model):
11
12     latent_dim = model.get_layer('Decoder-Embedding').output_shape[-1]
13
14     decoder_inputs = model.get_layer('Decoder-Input').input
15     dec_emb = model.get_layer('Decoder-Embedding')(decoder_inputs)
16     dec_bn = model.get_layer('Decoder-Batchnorm-1')(dec_emb)
17
18     gru_inference_state_input = layers.Input(shape=(latent_dim,),
19        name='hidden_state_input')
20
21     gru_out, gru_state_out = model.get_layer('Decoder-GRU')
22        ([dec_bn, gru_inference_state_input])
23
24     dec_bn2 = model.get_layer('Decoder-Batchnorm-2')(gru_out)
25     dense_out = model.get_layer('Final-Output-Dense')(dec_bn2)
26     decoder_model = models.Model([decoder_inputs, gru_inference_state_input],
27                                  [dense_out, gru_state_out])
28     return decoder_model
29
30
31 def load_model (cfg):
32
33     model = models.load_model(cfg.model_file)
34
35     encoder_model = model.get_layer('Encoder-Model')
36
37     decoder_model = get_decoder_model (model)
38
39     return encoder_model, decoder_model, model
40
```

INFERENCE

```
42 def predict_seq (cfg, encoder_model, decoder_mode, X):
43 
44     start_token = 0
45     start_token = 10
46 
47     embd_vec = encoder_model.predict(X)
48 
49     state_value =  start_token
50 
51         decoded_sentence = []
52         stop_condition = False
53 
54         while not stop_condition:
55 
56             preds, st = decoder_model.predict([state_value, embd_vec])
57 
58             # We are going to ignore indices 0 (padding) and indices 1 (unknown)
59             # Argmax will return the integer index corresponding to the
60             #  prediction + 2 b/c we chopped off first two
61 
62             pred_idx = np.argmax(preds[:, :, 2:]) + 2
63 
64             if pred_idx== end_token  or len(decoded_sentence) >=  cfg.max_target_seq:
65                   stop_condition = True
66                   break
67             decoded_sentence.append(pred_idx)
68 
69             # update the decoder for the next word
70             embd_vec = st
71             state_value = np.array(pred_idx).reshape(1, 1)
72 
73         return decoded_sentence
```

DECODER ARCHITECTURE



Decoder Architecture

SUMMARY

- Source code does have some similarities with the text in natural languages so language modelling can be applied on source code repos at massive scale [Allamanis (2013)].
- Github which hosts millions of open source projects (billion of lines of code) can be the ultimate source for data mining & applying machine learning on source code.
- Machine learning on source code can be used to identify useful patterns in source code that can be used in software development in different ways such as identifying risky commits, bug & defect prediction, security vulnerabilities, code summarisation, text generation etc., [Allamanis (2018)].

SUMMARY

- Neural machine translation models based on LSTM have been used on the AST of source code to find useful patterns:

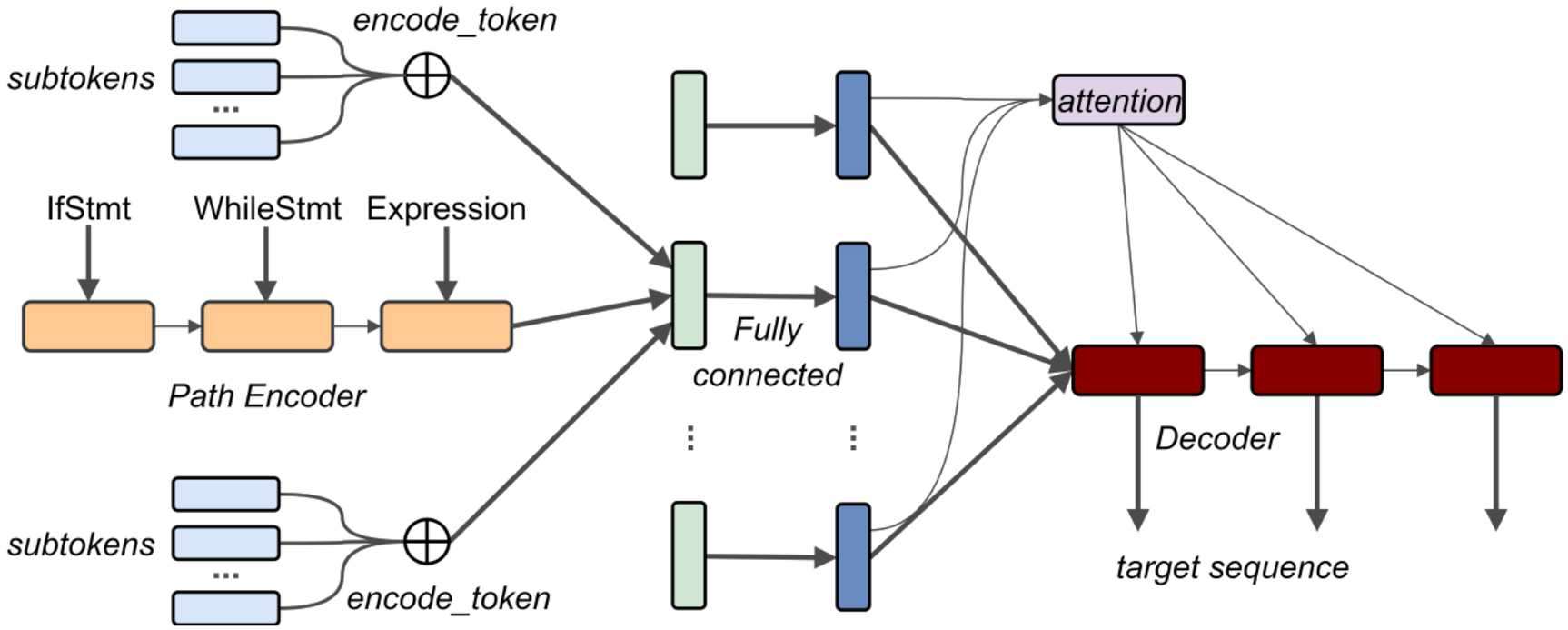


Figure 3: Our model encodes each path as a sequence of AST nodes, and averages the produced input vectors as the initial state of the decoder. The decoder generates an output sequence while attending over the encoded paths.

[Alon et. al. (2018a, 2018b)]

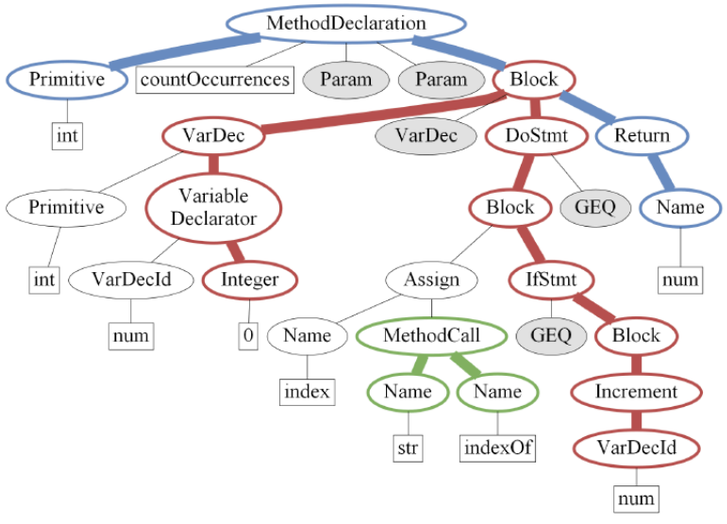
SUMMARY

```java
int countOccurrences(String str, char ch) {
    int num = 0;
    int index = -1;
    do {
        index = str.indexOf(ch, index + 1);
        if (index >= 0) {
            num++;
        }
    } while (index >= 0);
    return num;
}
```

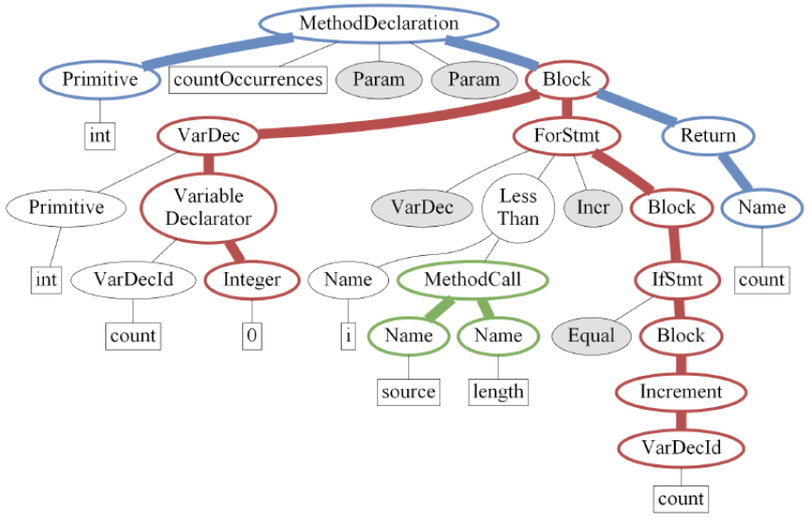
(a)

```java
int countOccurrences(String source, char value) {
    int count = 0;
    for (int i = 0; i < source.length(); i++) {
        if (source.charAt(i) == value) {
            count++;
        }
    }
    return count;
}
```

(b)



(c)



(d)

Figure 2: An example of two Java methods that have exactly the same functionality. Although having a different *sequential* (token-based) representation, considering syntactic patterns reveals recurring paths, which might differ only in single nodes (a `ForStmt` node instead of a `Do`-while node).

[Alon & Levy (2018)]

INDUSTRY TOOLS

1. Deepsoft

"DeepSoft, partly inspired by human memory, is built upon the powerful deep learning-based Long Short Term Memory architecture that is capable of learning long- term temporal dependencies that occur in software evolution.

DeepSoft provides a new approach for research into modelling of source code, risk prediction and mitigation, developer modelling, and automatically generating code patches from bug reports."

[Dam et. al. (2018)]

INDUSTRY TOOLS

2.DeepCoder

"The approach is to train a neural network to predict properties of the program that generated the outputs from the inputs.

We use the neural network's predictions to augment search techniques from the programming languages community, including enumerative search and an SMT-based solver.

Empirically, we show that our approach leads to an order of magnitude speedup over the strong non-augmented baselines and a Recurrent Neural Network approach, and that we are able to solve problems of difficulty comparable to the simplest problems on programming competition websites."

[Balog et. al. (2017)]

INDUSTRY TOOLS

3. DeepBugs

"extracts positive training examples from a code corpus, leverages simple program transformations to create negative training examples, trains a model to distinguish these two, and then uses the trained model for identifying programming mistakes in previously unseen code"

[Pradel (2018)]

INDUSTRY TOOLS

4. DeepRace

"DeepRace, a novel approach toward detecting data races in the source code. We build a deep neural network model to find data races instead of creating a data race detector manually.

Our model uses a one-layer convolutional neural network (CNN) with different window size to find data races method. Then we adopt the class activation map function with global average pooling to extract the weights of the last convolutional layer and backpropagate it with the input source code to extract the line of codes with a data race. Thus, the DeepRace model can detect the data race bugs on a file and line of code level."

[Tehrani et. al. (2019)]

SUMMARY

- Some interesting tools have been developed to apply machine learning on source code [Markovtsev & Kant (2017)].
- There have been proposed different approaches to model source code such as sequence tokens, trees & graphs [Brockschmidt et. al. (2018)].
- There have been studies to model source code change based on Tree2Tree models inspired from Seq2Seq model [Chakraborty et. al. (2018a, 2018b)].
- Bug fixing patches have also been generated using NMT models on source code [Michele (2018)].

CONCLUSIONS

- Machine learning on source code is a very promising area of research, however, we still have to see breakthroughs as we have seen in Natural Language Processing.
- This course has just introduced the field & a particular approach to the problem.
- Use of data from the open source repositories and machine learning we may see major developments in the software development process as we have seen in other industries.

REFERENCES

- Aho, Lam, Sethi & Ullman (2006), Compilers, principles, techniques & tools (Addison-Wesely).
- Markovtsev & Kant (2017), Topic modelling of public repositories at scales using names in source code, [arXiv:1704.00135v2].
- Sutskever et. al. (2014), Sequence to Sequence Learning with neural Networks, [arXiv: 14093215].
- Alon et. al. (2018a), code2seq: Generating Sequences from Structured Representations of Code, [arXiv: 1808.01400].
- Alon et. al. (2018b), code2vec, Learning distributed representation of code, ar[Xiv: 1803.09473].

REFERENCES

- Brockschmidt et. al. (2018), Code modelling with graph, [arXiv: 1805.08490].
- Chakraborty et. al. (2018a), Tree2Tree Neural Translation Model for Learning Source Code Changes, [arXiv:1810.00314].
- Chakraborty et. al. (2018b), CODIT: Code Editing with Tree-Based NeuralMachine Translation, arXiv: [arXiv:1810.00314]
- Dam et. al. (2016), DeepSoft: A vision for a deep model of software, [arXiv:1608.00092]
- Balog et. al. (2017), DeepCoder, Learning to write programs, [arXiv:1611.01989]

REFERENCE

- Pradel (2018), A Learning Approach to Name-based Bug Detection [arXiv:1805.11683].
- Tehrani et. al. (2019), DeepRace: Finding Data Race Bugs via Deep Learning [arXiv:1907.07110].

THANK YOU !