

ディープラーニングのフレームワークを 使った自然言語処理研究の実装の知見

寺西 裕紀

奈良先端科学技術大学院大学 自然言語処理学研究室

2017年9月1日(金)

第15回SVM勉強会

目次

- 自己紹介
- 機械学習は難しい
- 実装の工夫
 - Logging
 - Application
 - Preprocessing
 - Dataset
 - Loader
 - Model
 - Training
- 実例
- おまけ

- 内容
 - 個人的なノウハウ・提案・チュートリアル的な内容
 - DL+NLPをやり始めたときに知っていたかったこと
- 資料：
<https://github.com/chantera/svm2017>

自己紹介

- Background

- 大学(商学部) → 社会人(Web/アプリエンジニア)
→ NAIST松本研 (M2)
- 機械学習・深層学習 / NLP 歴：ちょうど1年ぐらい

- 研究内容

- 並列句解析: <https://www.slideshare.net/HirokiTeranishi1/ss-78087806>

- 会社

- TeraBytes inc. <https://terabytes.jp/>
 - 事業内容: 受託開発, コンサルティング



- 趣味

- ボクシング

機械学習は難しい

- 学習しない！ → 原因の切り分けが難しい
 - データが悪い
 - データの量が悪い・質が悪い
 - 前処理に問題がある
 - モデルが悪い
 - バグがある（ある程度動いてしまう）
 - モデルそのものが悪い（表現力 ↔ 汎化性能）
 - 学習方法が悪い
 - 最適化手法の選択が悪い, 学習率などの設定が悪い
 - 参考: <https://www.slideshare.net/takahirokubo7792/ss-65413290>
- 対応方法：Goodfellow本 (Chapter 11) が詳しい
 - URL : <http://www.deeplearningbook.org/contents/guidelines.html>

実装で回避

- 前提・目的：adhocな実装を避ける
 - adhocな実装はバグを生みやすい
 - モデル実装以外のバグで悩みたくない, バグを減らしたい
 - adhocな実装は学習がうまくいかない原因を探るのが大変
 - 原因：設定(データ, モデル, 学習方法), 実装(前処理, モデル実装, その他)
 - 研究的には設定の部分に時間を注ぎ込みたい
- 仕組み化・ライブラリ化
 - ディープラーニング / NLPに使える汎用的な仕組みを作る
 - 再利用なコードにまとめる
 - デバッグ用の処理を入れる, テストを書く
 - バグの減少, 問題の特定が容易, 開発の効率化
- 設計の見直し ← 今日の話のメイン

Example: BiLSTM POS tagger

```

1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 from argparse import ArgumentParser
5 from collections import defaultdict
6 import logging
7
8 import chainer
9 from chainer import functions as F, links as L
10 import numpy as np
11
12 class Model(chainer.Chain):
13     ...
14
15 def read_conll(file):
16     ...
17
18 def main(train_file,
19          test_file,
20          n_epochs=20,
21          batch_size=10,
22          lr=0.01,
23          gpu=-1):
24     train_sentences = read_conll(train_file)
25     n_train_samples = len(train_sentences)
26     test_sentences = read_conll(test_file)
27     n_test_samples = len(test_sentences)
28
29     word2id = defaultdict(lambda: len(word2id))
30     char2id = defaultdict(lambda: len(char2id))
31     tag2id = defaultdict(lambda: len(tag2id))
32
33     unknown = "<UNKNOWN>"
34     unknown_word_id = word2id[unknown]
35     unknown_char_id = char2id[unknown]
36
37     train_words = []
38     train_chars = []
39     train_tags = []
40     for tokens in train_sentences:
41         words = []
42         chars = []
43         tags = []
44         for token in tokens:
45             words.append(word2id[token['form'].lower()])
46             chars.append(np.array([char2id[char]
47                                   for char in token['form']], np.int32))
48             tags.append(tag2id[token['postag'].lower()])
49         train_words.append(np.array(words, np.int32))
50         train_chars.append(chars)
51         train_tags.append(np.array(tags, np.int32))
52
53     test_words = []
54     test_chars = []
55     test_tags = []
56     for tokens in test_sentences:
57         words = []
58         chars = []
59         tags = []
60         for token in tokens:
61             words.append(word2id.get(token['form'].lower(),
62                                     unknown_word_id))
63             chars.append(np.array([char2id.get(char, unknown_char_id)
64                                   for char in token['form']], np.int32))
65             tags.append(tag2id[token['postag'].lower()])
66         test_words.append(np.array(words, np.int32))
67         test_chars.append(chars)
68         test_tags.append(np.array(tags, np.int32))
69
70     model = Model(word_vocab_size=len(word2id),
71                  word_embed_size=100,
72                  char_vocab_size=len(char2id),
73                  char_embed_size=30,
74                  n_labels=len(tag2id))
75     model.train(train_words, train_chars)
76
77     if gpu > -1:
78         chainer.cuda.get_device_from_id(gpu).use()
79         model.to_gpu()
80
81     optimizer = chainer.optimizers.Adam(lr)
82     optimizer.setup(model)
83     optimizer.add_hook(chainer.optimizer.GradientClipping(5.0))
84
85     indices = np.arange(n_train_samples)
86     for epoch in range(n_epochs):
87         iteration, epoch_loss, epoch_accuracy = 0, 0.0, 0.0
88         chainer.config.train = True
89         chainer.config.enable_backprop = True
90         np.random.shuffle(indices)
91         for i in range(0, n_train_samples, batch_size):
92             model.cleargrads()
93             words = np.take(train_words, indices[i:i + batch_size])
94             chars = np.take(train_chars, indices[i:i + batch_size])
95             tags = np.take(train_tags, indices[i:i + batch_size])
96             ys = model(words, chars)
97             ts = F.softmax_cross_entropy(ys, tags, ignore_label=-1)
98             loss = F.softmax_cross_entropy(ys, ts, ignore_label=-1)
99             loss.backward()
100             optimizer.update()
101             accuracy = F.accuracy(ys, ts, ignore_label=-1)
102             epoch_loss += loss.data
103             epoch_accuracy += accuracy.data
104             iteration += 1
105         logging.info('[train epoch {:02d}] loss: {:.6f}, accuracy: {:.6f}'
106                     .format(epoch + 1,
107                             epoch_loss / iteration,
108                             epoch_accuracy / iteration))
109
110     model.cleargrads()
111     iteration, epoch_loss, epoch_accuracy = 0, 0.0, 0.0
112     chainer.config.train = False
113     chainer.config.enable_backprop = False
114     for i in range(0, n_test_samples, batch_size):
115         words = test_words[i:i + batch_size]
116         chars = test_chars[i:i + batch_size]
117         tags = test_tags[i:i + batch_size]
118         ys = model(words, chars)
119         ts = F.softmax_cross_entropy(ys, tags, ignore_label=-1)
120         loss = F.softmax_cross_entropy(ys, ts, ignore_label=-1)
121         accuracy = F.accuracy(ys, ts, ignore_label=-1)
122         epoch_loss += loss.data
123         epoch_accuracy += accuracy.data
124         iteration += 1
125     logging.info('[test epoch {:02d}] loss: {:.6f}, accuracy: {:.6f}'
126                 .format(epoch + 1,
127                         epoch_loss / iteration,
128                         epoch_accuracy / iteration))
129
130 if __name__ == "__main__":
131     logging.basicConfig(level=logging.DEBUG)
132     chainer.config.debug = False
133     chainer.config.type_check = False
134     parser = ArgumentParser()
135     parser.add_argument('--trainfile', type=str, required=True)
136     parser.add_argument('--testfile', type=str)
137     parser.add_argument('--epoch', type=int, default=20)
138     parser.add_argument('--batchsize', type=int, default=10)
139     parser.add_argument('--lr', type=float, default=0.01)
140     parser.add_argument('--gpu', type=int, default=-1)
141     args = parser.parse_args()
142     main(args.trainfile, args.testfile, args.epoch, args.batchsize, args.lr)

```

Logging

- 原因特定のためには最も重要
- 機能の作り込み
 - ファイルと標準（エラー）出力への書き込み
 - 異なるログレベルを設定可能にする
 - ログファイルのユーティリティ
- フォーマットの見直し
 - 日付+タイムゾーン, プロセスごとの識別子, メッセージ
 - ミリ秒まで出す：正確な時間の把握, 簡易な時間計測
 - 識別子：同一ファイルに同時に書き込んだ場合のメッセージの識別
- 実験設定を残す
 - OS・実行環境, コマンドライン引数, モデルアーキテクチャ, ハイパーパラメータ

Logging

※Pythonの場合

- 標準ライブラリのロガーを拡張
- モデルのアーキテクチャは `__repr__` を実装して表現
- コマンドライン引数や `main()` への引数は薄いラッパーを使って呼び出し側でロギングすると楽
- `main()` は `try-except` でラップして例外はログへ

```
138 DATE_FORMAT = "%Y-%m-%d %H:%M:%S.%f %Z"
139
140
141 class AppLogger(Logger):
142     FORMAT = "%(asctime)-15s\t%(accessid)s\t[%](levelname)s\t%(message)s"
143     _config = {
144         'level': INFO,
145         'verbosity': TRACE,
146         'filelog': True,
147         'logdir': None,
148         'filename': "%Y%m%d.log",
149         'filemode': 'a',
150         'fileprefix': '',
151         'filesuffix': '',
152         'fmt': FORMAT,
153         'datefmt': DATE_FORMAT,
154         'mkdir': False,
155     }
156
157     @classmethod
158     def configure(cls, **kwargs):
159         cls._config.update(kwargs)
160
161     def initialize(self):
162         super(AppLogger, self).initialize()
163         config = AppLogger._config
164         now = datetime.now(tzlocal())
165         self._accessid = uuid.uuid4().hex[:6]
166         self._uniqueid = "UNIQUEID"
167         self._accesssec = now
168         self._accesstime = now.strftime(config['datefmt'])
169
170         if len(self.handlers) == 0:
171             if config['filelog']:
172                 self._add_file_handler(config)
173
174             stream_handler = logging.StreamHandler()
175             stream_handler.setLevel(config['verbosity'])
176             stream_handler.setFormatter(
177                 ColoredFormatter(config['fmt'], config['datefmt']))
178             self.addHandler(stream_handler)
179
180         message = "LOG Start with ACCESSID=[%s] UNIQUEID=[%s] ACCESTIME=[%s]"
181         self.info(message % (self._accessid, self._uniqueid, self._accesstime))
```


Logging

```

hiroki $ python example/tagging-final.py --trainfile ~/Desktop/NLP/data/ptb-sd3.3.0/dep/wsj_train.conll --testfile ~/Desktop/NLP/data/ptb-sd3.3.0/dep/wsj_test.conll --epoch 50 --batchsize 32
2017-08-31 10:44:12.206 JST      2ecee2 [info] LOG Start with ACCESSID=[2ecee2] UNIQUEID=[UNIQID] ACESSTIME=[2017-08-31 10:44:12.206496 JST]
2017-08-31 10:44:12.207 JST      2ecee2 [trace] posix.uname_result(sysname='Darwin', nodename='Hirokis-MacBook-Air.local', release='15.6.0', version='Darwin Kernel Version 15.6.0: Sun Jun  4 21:43:07 PDT 2017; root:xnu-3248.70.3~1/RELEASE_ARM_T8020', machine='arm64t8020')
2017-08-31 10:44:12.207 JST      2ecee2 [trace] sys.argv: ['example/tagging-final.py', '--trainfile', '/Users/hiroki/Desktop/NLP/data/ptb-sd3.3.0/dep/wsj_train.conll', '--testfile', '/Users/hiroki/Desktop/NLP/data/ptb-sd3.3.0/dep/wsj_test.conll', '--epoch', '50', '--batchsize', '32']
2017-08-31 10:44:12.207 JST      2ecee2 [trace] app._config: {'debug': False, 'logdir': '/Users/hiroki/Desktop/svm2017/example/logs', 'loglevel': 'info', 'logoption': 'a', 'quiet': False}
2017-08-31 10:44:12.207 JST      2ecee2 [info] *** [START] ***
2017-08-31 10:44:12.207 JST      2ecee2 [debug] App._process(self) called - command: <function train at 0x10e5d48c8>, args: {'batch_size': 32, 'n_epoch': 50, 'gpu': -1, 'lr': 0.01, 'test_file': '/Users/hiroki/Desktop/NLP/data/ptb-sd3.3.0/dep/wsj_test.conll', 'train_file': '/Users/hiroki/Desktop/NLP/data/ptb-sd3.3.0/dep/wsj_train.conll'}
2017-08-31 10:44:12.512 JST      2ecee2 [info] initialized model: {'dropout': 0.5, 'lstm_hidden_size': 200, 'n_lstm_layers': 1, 'char_filter_size': 30, 'char_window_size': 3, 'n_labels': 45, 'char_embed_size': 30, 'char_vocab_size': 78, 'word_vocab_size': 7305, 'self': {...}, '__class__': <class '__main__.Model'>, 'links': (<chainer.links.connection.embed_id.EmbedID object at 0x10fc44dd8>, {'dropout': 0.5, 'window_size': 3, 'out_size': 30, 'embed_size': 30, 'vocab_size': 78, 'self': {...}, '__class__': <class '__main__.CharCNN'>}, <chainer.links.connection.n_step_lstm.NStepBiLSTM object at 0x10fc44e10>, <chainer.links.connection.linear.Linear object at 0x10fca6a20>)}
2017-08-31 10:46:40.329 JST      2ecee2 [info] [train epoch 01] loss: 1.399560, accuracy: 0.594988
2017-08-31 10:46:42.383 JST      2ecee2 [info] [test epoch 01] loss: 0.772760, accuracy: 0.778271
2017-08-31 10:48:15.499 JST      2ecee2 [info] [train epoch 02] loss: 0.713188, accuracy: 0.784788
2017-08-31 10:48:17.590 JST      2ecee2 [info] [test epoch 02] loss: 0.490854, accuracy: 0.869005
2017-08-31 10:49:47.006 JST      2ecee2 [info] [train epoch 03] loss: 0.515316, accuracy: 0.844857
2017-08-31 10:49:48.949 JST      2ecee2 [info] [test epoch 03] loss: 0.416239, accuracy: 0.889443
2017-08-31 10:51:22.413 JST      2ecee2 [info] [train epoch 04] loss: 0.410470, accuracy: 0.876689
2017-08-31 10:51:24.432 JST      2ecee2 [info] [test epoch 04] loss: 0.386856, accuracy: 0.902937
AC2017-08-31 10:51:26.110 JST      2ecee2 [warn] Signal(2) received: The program /Users/hiroki/Desktop/teras/teras/app/_init_.py will be closed
2017-08-31 10:51:26.206 JST      2ecee2 [info] LOG End with ACCESSID=[2ecee2] UNIQUEID=[UNIQID] ACESSTIME=[2017-08-31 10:44:12.206496 JST] PROCESSTIME=[433.997841000]

hiroki $

```

- ログから実験の再現ができそう

Application

- 統合的な役割
 - コマンドライン引数の処理
 - 設定ファイルの読み書き
 - ロガーの初期化
 - 環境設定の切り替え
 - デバッグモードの切り替え
 - 命令の実行
 - 例外ハンドリング
 - シグナルハンドリング

```
33
34 class App(AppBase):
35     DEFAULT_APP_NAME = "teras.app"
36     DEFAULT_CONFIG_FILE = "~/.teras.conf"
37     app_name = ''
38     _argparser = ConfigArgParser(DEFAULT_CONFIG_FILE)
39     verbose = True
40
41     ...
42
43     @classmethod
44     def run(cls, command=None):
45         cls.configure()
46         command, command_args, common_args \
47             = cls._parse_args(command)
48         try:
49             def handler(signum, frame):
50                 raise SystemExit("Signal(%d) received: "
51                                   "The program %s will be closed"
52                                   % (signum, __file__))
53             signal.signal(signal.SIGINT, handler)
54             signal.signal(signal.SIGTERM, handler)
55             os.umask(0)
56
57             self = cls._get_instance()
58             self._initialize(command, command_args, common_args)
59             self._preprocess()
60             self._process()
61             self._postprocess()
62             self._finalize()
63         except Exception:
64             logging.e("Exception occurred during execution:",
65                       exc_info=True, stack_info=cls.debug)
66         except SystemExit as e:
67             logging.w(e)
68         finally:
69             logging.getLogger().finalize()
70             sys.exit(0)
```

Application

```
239   
240 if __name__ == "__main__":  
241     logging.AppLogger.configure(mkdir=True)  
242     App.add_command('train', train, {  
243         'train_file': arg('--trainfile', type=str, required=True),  
244         'test_file': arg('--testfile', type=str),  
245         'n_epoch': arg('--epoch', type=int, default=20),  
246         'batch_size': arg('--batchsize', type=int, default=10),  
247         'lr': arg('--lr', type=float, default=0.01),  
248         'gpu': arg('--gpu', type=int, default=-1),  
249     })  
250     App.run()
```

- 実行したい関数をラップするだけ

Preprocessing

- 汎用的な前処理
 - 表層形の処理
 - lowercase化, 数字のトークン化
 - 単語のID化, 未知語処理
 - IDから単語のlookup
 - パディング
 - pretrained embeddingの読み込み
 - 2種類のファイル形式に対応
 - 辞書・ベクトル同一ファイル
 - 辞書・ベクトル別ファイル
 - ヘッダー有無: (単語数, 次元数)
 - データの正規化

```
318 class EmbeddingPreprocessor(Preprocessor):
319     def __init__(self,
320                 embed_file=None,
321                 embed_size=50,
322                 unknown="<UNK>",
323                 pad=None,
324                 tokenizer=split,
325                 initializer=standard_normal,
326                 preprocess=lower,
327                 embed_dtype=np.float32):
328         self.embed_dtype = embed_dtype
329         self._init_embeddings(embed_file, embed_size)
330         super(EmbeddingPreprocessor, self).__init__(
331             unknown, pad, tokenizer, preprocess,
332             _get_int_type(embed_dtype))
333         self._initializer = initializer
334         if not self.use_pretrained and self._pad_id >= 0:
335             self.get_embeddings()[self._pad_id] = 0
336
337     def _init_embeddings(self, embed_file, embed_size):
338         if embed_file is not None:
339             vocab_file = None
340             if isinstance(embed_file, (list, tuple)):
341                 embed_file, vocab_file = embed_file
342             vocabulary, embeddings = \
343                 load_embeddings(embed_file,
344                               vocab_file,
345                               self.embed_dtype)
346             self.use_pretrained = True
347         elif embed_size is not None:
348             ...
349         ...
350     ...
351
```

Dataset

- 汎用的なミニバッチの処理
 - samplesのバッチイテレーション
 - samplesのシャッフル
 - sampleの属性ごとのバッチ化
 - [(word, pos, label), ...]
で初期化したデータを
(words, chars, labels)に変換
 - なるべく余計な処理はさせない

```
7
8 class Dataset(Sequence):
9     """Immutable Class"""
10
11     def __init__(self, *samples):
12         self._n_cols = len(samples)
13         if self._n_cols == 0:
14             self._columns = ([],)
15             self._n_cols = 1
16         elif self._n_cols == 1:
17             if isinstance(samples[0], (list, tuple)):
18                 columns = tuple(
19                     np.array(column)
20                     if isinstance(column[0], np.ndarray) else column
21                     for column in map(tuple, zip(*samples[0])))
22                 self._columns = columns
23                 self._n_cols = len(columns)
24             else:
25                 self._columns = (samples[0],)
26         elif self._n_cols > 1:
27             self._columns = samples
28         self._samples = list(map(tuple, zip(*self._columns)))
29         self._len = len(self._samples)
30         self._indices = np.arange(self._len)
31
32     def __iter__(self):
33         return (sample for sample in self._samples)
34
35     def batch(self, size, shuffle=False, colwise=True):
36         if shuffle:
37             np.random.shuffle(self._indices)
38         if colwise:
39             return self._get_col_iterator(size)
40         else:
41             return self._get_row_iterator(size)
42
```

Loader

- Loader
= FileReader
+ Preprocessing
+ Dataset
- 実装・機能
 - readerからitemsをイテレート
 - filterとmapを継承してitemを変換 (preprocessorを呼ぶ)

```
12 class CorpusLoader(Loader):
13     @abstractmethod
14     def map(self, item):
15         words, labels = zip(*[(token['form'],
16                               self.tag_map.add(token['postag']))
17                               for token in item])
18         sample = (self._word_transform(words, as_one=True),
19                  self.get_processor('char')
20                    .fit_transform(words, as_one=False),
21                  np.array(labels, dtype=np.int32))
22         return sample
23
24     def filter(self, item):
25         return True
26
27     def load(self, file, train=False, size=None):
28         self._train = train
29         self._reader.set_file(file)
30         if size is None:
31             samples = [self.map(item) for item in self._reader
32                        if self.filter(item)]
33         else:
34             samples = []
35             for item in self._reader:
36                 if len(samples) >= size:
37                     break
38                 if not self.filter(item):
39                     continue
40                 samples.append(self.map(item))
41
42         return Dataset(samples)
```

Model

- よくある実装

```
55   
56 class Model:  
57   
58     def forward(xs, ts):  
59         ...  
60         ys = self.linear(hs)  
61         loss = F.softmax_cross_entropy(ys, ts)  
62         return ys, loss  
63   
64   
65 ...  
66 ys, loss = model.forward(xs, ts)  
67 accuracy = F.accuracy(ys, ts)  
68 loss.backward()  
69 
```

Model

- モデルのforward計算でlossを計算させない
 - predict時にはlossの計算は必要ない
 - 本来predict時には正解データは必要ない
 - forwardメソッドの引数に正解データを入れたくない
- モデルと学習の分離
 - Modelの外でloss関数を設定
 - loss関数は タスク依存 > モデル依存
 - 例) 多クラス分類ならモデルはラベルのスコア(logit)を返せばよい
 - 呼び出し側でクロスエントロピーなりヒンジなりを選ぶ
- モデルに学習/予測の処理をなるべく意識させない
 - せいぜいdropoutぐらい

Model

- よくある実装

```
1
2 class MLP(chainer.Chain):
3
4     def __init__(self, n_units, n_out, dropout=0.5):
5         super(MLP, self).__init__()
6         with self.init_scope():
7             self.l1 = L.Linear(None, n_units)
8             self.l2 = L.Linear(None, n_units)
9             self.l3 = L.Linear(None, n_out)
10            self.dropout_ratio = dropout
11
12    def __call__(self, x):
13        h1 = F.dropout(F.relu(self.l1(x)), self.dropout_ratio)
14        h2 = F.dropout(F.relu(self.l2(h1)), self.dropout_ratio)
15        return self.l3(h2)
16
```

- 先行研究と同じモデルなのに再現できない、なぜ？

Model

- 重みの初期化を明示的に指定しているか？
 - 使っているクラスの重みの初期化を正確に理解しているか
 - normal, uniform, zeros, lecun, gloriot (xavier), he
 - デフォルトの初期化が適さないケースもある
 - activation関数によって適した初期化が違う,
LSTMのforget gateのbiasを1にするかどうか
- 関数の実装は把握しているか？
 - dropoutのpredict時のscalingは？, LSTMのpeepholeは？
- フレームワークのソースコードは必ず読む
 - 先行研究を再現する場合は公開されているコードで使っている
フレームワークと自分が使うフレームワークの両方
- 研究の場合はなるべく $Wx + b$ レベルで自分で書く

Training

- 毎回書いていないか？
 - epoch / batch のイテレーション
 - trainフラグの管理
 - backwardの呼び出し, 重みの更新
 - loss, accuracyの計測/出力

```
196 indices = np.arange(n_train_samples)
197 for epoch in range(n_epoch):
198     iteration, epoch_loss, epoch_accuracy = 0, 0.0, 0.0
199     chainer.config.train = True
200     chainer.config.enable_backprop = True
201     np.random.shuffle(indices)
202     for i in range(0, n_train_samples, batch_size):
203         model.cleargrads()
204         words = np.take(train_words, indices[i: i + batch_size])
205         chars = np.take(train_chars, indices[i: i + batch_size])
206         tags = np.take(train_tags, indices[i: i + batch_size])
207         ys = model(words, chars)
208         ys = F.concat(ys, axis=0)
209         ts = F.concat(tags, axis=0)
210         if gpu >= 0:
211             ts.to_gpu()
212         loss = F.softmax_cross_entropy(ys, ts, ignore_label=-1)
213         loss.backward()
214         optimizer.update()
215         accuracy = F.accuracy(ys, ts, ignore_label=-1)
216         epoch_loss += loss.data
217         epoch_accuracy += accuracy.data
218         iteration += 1
219     logging.info('[train epoch {:02d}] loss: {:.6f}, accuracy: {:.6f}'
220                  .format(epoch + 1,
221                          epoch_loss / iteration,
222                          epoch_accuracy / iteration))
223
```

Training

- 設計方針：モデルと学習の分離
 - Trainerクラスにはforward関数, loss関数, accuracy関数(必要であれば)を渡す
 - trainerは学習時にforward → loss (& accuracy) → backward & update, テスト時はforward → accuracy を呼び出す
- タスク・実験設定に固有な処理をループに入りたい
 - 例) テスト時のdecoding + evaluation
 - Trainerクラスをオブザーバーパターンで書く (Keras参考)
 - ループ内でイベント処理：self.notify(BATCH_END, data)
- forward, backward以外の処理が重いと話にならない
 - ループ部分はC, C++, Cythonで書く

Refactoring: Model

```

69
70 class Model(chainer.Chain):
71
72     def __init__(self,
73                 word_embedding,
74                 char_embedding,
75                 n_labels,
76                 char_window_size=3,
77                 char_filter_size=30,
78                 n_blstm_layers=1,
79                 lstm_hidden_size=200,
80                 dropout=0.5):
81         """
82         @TODO: logging parameters initialization
83         """
84         super().__init__()
85         if isinstance(word_embedding, np.ndarray):
86             word_vocab_size, word_embed_size = word_embedding.shape
87         else:
88             word_vocab_size, word_embed_size = word_embedding
89             word_embedding = None
90         self.dropout = dropout
91         with self.init_scope():
92             self.word_embed = L.EmbedID(word_vocab_size, word_embed_size,
93                                         word_embedding)
94             self.char_cnn = CharCNN(char_embedding,
95                                    char_filter_size, char_window_size,
96                                    dropout)
97             # @TODO: should specify parameters initialization
98             self.blstm = L.NStepBiLSTM(n_blstm_layers,
99                                       word_embed_size + char_filter_size,
100                                       lstm_hidden_size, dropout)
101             self.linear = L.Linear(lstm_hidden_size * 2, n_labels,
102                                   initialW=initializers.GlorotUniform(),
103                                   initial_bias=0)
104         self._desc = {} # model description
105

```

```

106
107     def __call__(self, word_seqs, char_seqs):
108         indices = [0]
109         xs = []
110         for word_seq, char_seq in zip(word_seqs, char_seqs):
111             x_word = self.word_embed(self.xp.array(word_seq))
112             x_char = self.char_cnn(char_seq)
113             x = F.concat((x_word, x_char))
114             indices.append(indices[-1] + x.shape[0])
115             xs.append(F.dropout(x, self.dropout))
116         hy, cy, ys = self.blstm(hx=None, cx=None, xs=xs)
117         ys = F.concat(ys, axis=0)
118         ys = self.linear(F.dropout(ys, self.dropout))
119         ys = F.split_axis(ys, indices[1:-1], axis=0)
120         return ys
121
122     def __repr__(self):
123         return repr(self._desc)
124

```

Refactoring: Loader

```
135 class DataLoader(dataset.loader.CorpusLoader):
136
137     def __init__(self,
138                 word_embed_size=100,
139                 char_embed_size=30,
140                 word_embed_file=None,
141                 word_preprocess=lambda x: x.lower(),
142                 word_unknown="<UNKNOWN>",
143                 embed_dtype='float32'):
144         super(DataLoader, self).__init__(reader=io.reader.ConllReader())
145         self.use_pretrained = word_embed_file is not None
146         self.add_processor(
147             'word', embed_file=word_embed_file, embed_size=word_embed_size,
148             embed_dtype=embed_dtype,
149             initializer=Uniform(scale=np.sqrt(3 / word_embed_size)),
150             preprocess=word_preprocess, unknown=word_unknown)
151         self.add_processor(
152             'char', embed_file=None, embed_size=char_embed_size,
153             embed_dtype=embed_dtype,
154             initializer=Uniform(scale=np.sqrt(3 / char_embed_size)),
155             preprocess=lambda x: x.lower())
156         self.tag_map = preprocessing.text.Vocab()
157
158     def map(self, item):
159         # item -> (words, chars, labels)
160         words, labels = zip(*[(token['form'],
161                               self.tag_map.add(token['postag']))
162                               for token in item[1:]]) # skip root token
163         sample = (self._word_transform(words, as_one=True),
164                 self.get_processor('char')
165                     .fit_transform(words, as_one=False),
166                 np.array(labels, dtype=np.int32))
167         return sample
168
169     def load(self, file, train=False, size=None):
170         if train and not self.use_pretrained:
171             # assign an index if the given word is not in vocabulary
172             self._word_transform = self.get_processor('word').fit_transform
173         else:
174             # return the unknown word index if the word is not in vocabulary
175             self._word_transform = self.get_processor('word').transform
176         return super(DataLoader, self).load(file, train, size)
```

Refactoring: Training

```
179
180 def train(
181     train_file,
182     test_file,
183     word_embed_file=None,
184     word_embed_size=100,
185     char_embed_size=30,
186     n_epoch=20,
187     batch_size=10,
188     lr=0.01):
189     logging.info('initialize DataLoader with word_embed_file={}, '
190                 'word_embed_size={}, and char_embed_size={}'
191                 .format(word_embed_file, word_embed_size, char_embed_size))
192     loader = DataLoader(word_embed_size, char_embed_size, word_embed_file)
193     logging.info('load train dataset from {}'.format(train_file))
194     train_dataset = loader.load(train_file, train=True)
195     logging.info('load test dataset from {}'.format(test_file))
196     test_dataset = loader.load(test_file, train=False)
197
198     model = Model(word_embedding=loader.get_embeddings('word'),
199                  char_embedding=loader.get_embeddings('char'),
200                  n_labels=len(loader.tag_map))
201     logging.info('initialized model: {}'.format(model))
202
203     optimizer = chainer.optimizers.Adam(lr)
204     optimizer.setup(model)
205     optimizer.add_hook(chainer.optimizer.GradientClipping(5.0))
206
207     trainer = training.Trainer(optimizer, model,
208                                loss_func=lambda ys, ts:
209                                    F.softmax_cross_entropy(
210                                        F.concat(ys, axis=0), F.concat(ts, axis=0)),
211                                accuracy_func=lambda ys, ts:
212                                    F.accuracy(
213                                        F.concat(ys, axis=0), F.concat(ts, axis=0)))
214     trainer.configure(framework_utils.config)
215
216     trainer.fit(train_dataset, None, batch_size=batch_size,
217                epochs=n_epoch, validation_data=test_dataset,
218                verbose=App.verbose)
219
```

Example: BiaffineParser

- BiaffineParser (Dozat+, 2016)
 - Chainer / PyTorch実装
<https://github.com/chantera/biaffineparser>
 - コマンドライン引数で backendを'chainer'か 'pytorch'を指定
 - 提案した設計の利点
 - フレームワークに依存しない
- Modelクラスだけフレームワークで実装してtrainerに modelとoptimizerとロス関数を渡すだけ

```
63 |  
64 | model = model_cls(  
65 |     embeddings=(loader.get_embeddings('word'),  
66 |                 loader.get_embeddings('pos')),  
67 |     n_labels=len(loader.label_map),  
68 |     **model_params,  
69 | )  
70 |  
71 | if gpu >= 0:  
72 |     framework_utils.set_model_to_device(model, device_id=gpu)  
73 |  
74 | if backend == 'chainer':  
75 |     optimizer = chainer.optimizers.Adam(  
76 |         alpha=lr, beta1=0.9, beta2=0.9, eps=1e-08)  
77 |     optimizer.setup(model)  
78 |     optimizer.add_hook(chainer.optimizer.GradientClipping(5.0))  
79 | elif backend == 'pytorch':  
80 |     optimizer = torch.optim.Adam(model.parameters(),  
81 |                                   lr=lr, betas=(0.9, 0.9), eps=1e-08)  
82 |     torch.nn.utils.clip_grad_norm(model.parameters(), max_norm=5.0)  
83 |  
84 | Log.i('optimizer: Adam(alpha={}, beta1=0.9, '  
85 |       'beta2=0.9, eps=1e-08), grad_clip=5.0'.format(lr))  
86 |  
87 | parser = models.BiaffineParser(model)  
88 |  
89 | trainer = Trainer(optimizer, parser, loss_func=parser.compute_loss,  
90 |                   accuracy_func=parser.compute_accuracy)  
91 | trainer.configure(framework_utils.config)  
92 | if backend == 'pytorch':  
93 |     trainer.add_hook(Event.EPOCH_TRAIN_BEGIN, lambda data: model.train())  
94 |     trainer.add_hook(Event.EPOCH_VALIDATE_BEGIN, lambda data: model.eval())  
95 | if test_dataset:  
96 |     trainer.attach_callback(  
97 |         utils.Evaluator(parser,  
98 |                         pos_map=loader.get_processor('pos').vocabulary,  
99 |                         ignore_punct=True))  
100 |  
101 | trainer.fit(train_dataset, None,  
102 |            batch_size=batch_size,  
103 |            epochs=n_epoch,  
104 |            validation_data=test_dataset,  
105 |            verbose=App.verbose)
```


実装力を高めるために...

- ソースコードを読む
 - Chainer, Keras, DyNet
- きれいなコードを書く
 - コーディング規約を読む
 - [Google C++ Style Guide](#)
 - [PEP8](#)
 - linterを使ってコード解析/整形する : cpplint, flake8
- 勉強
 - [リーダブルコード](#)
 - [デザインパターン](#)
 - [PofEAA](#)
 - [Effective C++](#) / [Effective Java](#)

おまけ



himako
@himkt

フォローする



インターンの人、NAISTは山奥で暇なのでみんなディープラーニングフレームワークを作りがちみたいな話を聞いてなるほどとなった

4:53 - 2017年8月4日

- 研究するなら結局はフレームワークのソースコードを読んで理解しなければならない → 仕様が変わるし面倒
- オレオレフレームワークを作ると幸せになれる
- フレームワーク実装の難しさの8割はtensor計算
 - tensor計算をライブラリに任せるならすぐ作れる
 - CPU/GPUの演算が同じインタフェースでできればもっと楽
 - numpy/cupy, arrayfire, etc...

おまけ



Yuya Unno
@unnonouno

フォロー中



自動微分のおかげで、微分なんてしなくていい人生なんだと思ったら、作る側だった。。。

6:17 - 2017年8月21日

- フレームワーク実装の面倒くささの8割は微分の実装
- 自分の研究で使うときに必要な関数だけ書けば楽
 - みんなに使ってもらうフレームワークは目指さない
- フレームワーク雑感
 - Chainer: プロトタイピング
 - DyNet: アルゴリズムをゴリゴリ書くときに使う
 - Tensorflow: プロダクション
 - PyTorch: 洗練されてないのでコードが汚くなる, 今後に期待

Appendix

- 発表資料 : <https://github.com/chantera/svm2017>
- ライブラリ : <https://github.com/chantera/teras>
 - 自分の研究用なのでAPIはちょくちょく変わります, forkして使ってください
 - 開発計画 : Cython実装, Transition Systemの汎用的なクラス(Stateなど)

```
1 teras
2 |__init__.py
3 |__app__
4 |__init__.py
5 |__argparse.py
6 |__base__
7 |__init__.py
8 |__event.py
9 |__text.py
10 |__dataset__
11 |__init__.py
12 |__dataset.py
13 |__download.py
14 |__loader.py
15 |__mnist.py
16 |__framework__
17 |__init__.py
18 |__chainer__
19 |__init__.py
20 |__callbacks.py
21 |__functions.py
22 |__model.py
23 |__optimizers.py
24 |__dynet__
25 |__init__.py
26 |__functions.py
27 |__model.py
28 |__pytorch__
29 |__init__.py
30 |__callbacks.py
31 |__model.py
32 |__io__
33 |__init__.py
34 |__reader.py
35 |__logging__
36 |__init__.py
37 |__nlp__
38 |__transition__
39 |__arc_standard.py
40 |__state.py
41 |__preprocessing__
42 |__init__.py
43 |__text.py
44 |__training__
45 |__init__.py
46 |__callbacks.py
47 |__event.py
48 |__trainer.py
49 |__utils__
50 |__init__.py
51 |__builtin.py
52 |__git.py
53 |__progressbar.py
54
55 14 directories, 38 files
```