Embedding Matching 源码阅读报告

Junfeng Hu*
15210240075
School of Computer Science
Fudan University
No.825, Zhangheng Road, Shanghai
15210240075@fudan.edu.cn

Contents

1 代码概况

《Accurate Linear-Time Chinese Word Segmentation via Embedding Matching》论文代码基于 Python 实现。

code 文件夹是代码实现，working_data 文件夹包含 PKU 数据集，以及 perl score 脚本。

在 code 文件夹下一共有以下几个文件：

- config.txt 实验配置文件

*http://junfenglx.github.io/

Table 1: PKU 数据集实验重现结果

指标	大小
F MEASURE	0.950
OOV Rate	0.058
OOV Recall Rate	0.756
IV Recall Rate	0.957

- predict.py 使用已有模型对预料库进行分词
- script2.py 训练模型脚本，重现实验
- seg2.py 算法实现文件
- word2vec2.py gensim 的 word2vec 实现的修改版本。

该代码依赖环境如下：

- Linux like OS
- Python 2.7, not Python 3.X
- Numpy >= 1.9
- gensim == 0.10.3

2 实验重现

在 code 文件夹下运行命令：

```
python script2.py config.txt
```

训练过程大概需要 1 个小时左右。config.txt 文件中指定迭代 10 次，当然也可以增大和减小迭代次数。

最终对测试集分词的结果如表 1

和其论文中得出的结果基本一致。

3 源码分析

code 文件夹下 seg2.py 是包含论文算法实现代码的文件。

其主要包含类 Seger，该类是算法实现类，其继承 gensim 的 Word2Vec 类。

该类主要方法见图 1

算法相关的函数主要有三个：

1. predict_single_position
2. train_gold_per_sentence
3. predict_sentence_greedy

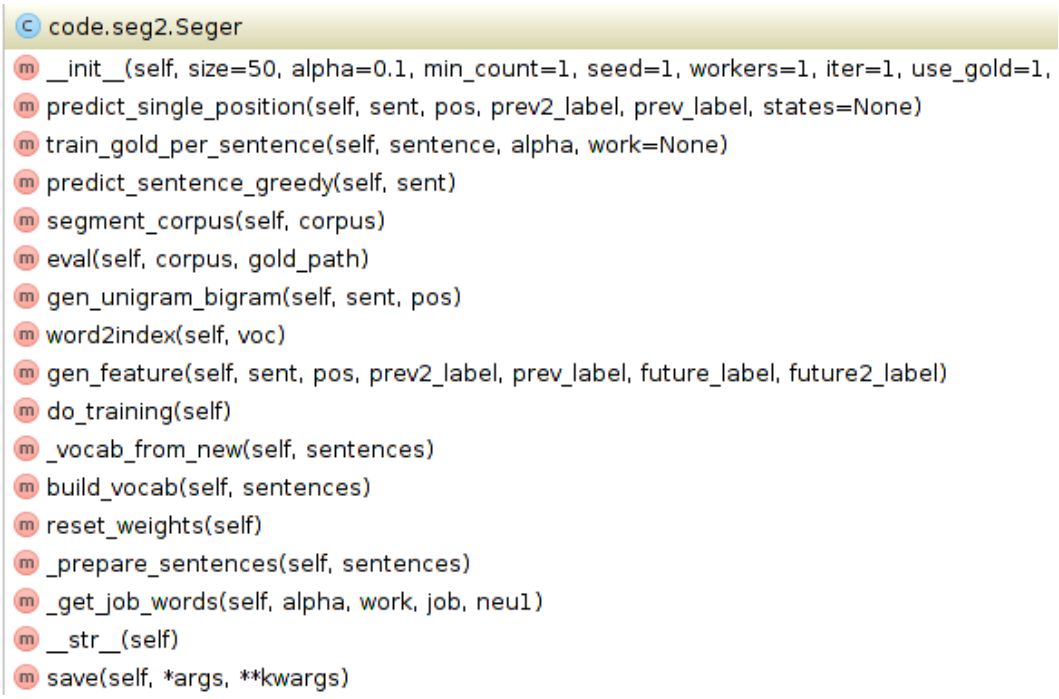

Figure 1: Seger class diagram

3.1 predict_single_position

predict_single_position 函数的 python doc 见 Code 1

predict_single_position

```
def predict_single_position(self, sent, pos,
                            prev2_label,
                            prev_label, states=None):
    """
    predict a character's label
    :param sent: the sentence
    :param pos: the character position
    :param prev2_label: second previous label
    :param prev_label: first previous label
    :param states: the previous iter states vector
    :return: softmax_score2, feature_index_list,
     pred_index_list2, feature_vec, pred_matrix2
    """
```

Code 1: predict_single_position 方法

predict_single_position 主要代码片段见 Code 2

predict_single_position 关键代码

```
pred_tuple = tuple([self.su_prefix + varient + u
    for varient in self.state_varient])
if pred_tuple[0] in self.vocab and pred_tuple[1] in self.vocab:
    pass
else:
    pred_tuple = None
    if self.train_mode:
        print 'Unknown candidate! Should NOT happen during training!'
        assert False

# constant $LABEL0, $LABEL1
# for unknown words in test
pred_tuple2 = tuple([self.label0_as_vocab, self.label1_as_vocab])

softmax_score = None
if pred_tuple:
    pred_index_list = [self.vocab[pred].index for pred in pred_tuple]
    pred_matrix = self.syn1neg[pred_index_list]

    if block is not None:
        pred_matrix = multiply(block, pred_matrix)
    elif self.drop_out:
        pred_matrix = (1 - self.dropout_rate) * pred_matrix

    raw_score = exp(dot(feature_vec, pred_matrix.T))
    softmax_score = raw_score / sum(raw_score)

pred_index_list2 = [self.vocab[pred].index for pred in pred_tuple2]
pred_matrix2 = self.syn1neg[pred_index_list2]

if block is not None:
    pred_matrix2 = multiply(block, pred_matrix2)
elif self.drop_out:
    # should be pred_matrix2
    pred_matrix2 = (1 - self.dropout_rate) * pred_matrix2

raw_score2 = exp(dot(feature_vec, pred_matrix2.T))
softmax_score2 = raw_score2 / sum(raw_score2)
# print pred_matrix2.shape, pred_matrix.shape
if pred_tuple:
    softmax_score2 = np_append(softmax_score2, softmax_score)
    pred_index_list2.extend(pred_index_list)
    pred_matrix2 = np_append(pred_matrix2, pred_matrix, axis=0)
```

Code 2: predict_single_position 关键代码

3.2 train_gold_per_sentence

train_gold_per_sentence 函数的 python doc 见 Code 3

train_gold_per_sentence

```
def train_gold_per_sentence(self, sentence, alpha, work=None):
    """
    :param sentence: the segmented sentence
    :param alpha: the learning rate
    :param work: self.layer1_size vector,
     initialized with zero, not use
    :return: words count_sum, train error_sum for report
    """
```

Code 3: train_gold_per_sentence 方法

train_gold_per_sentence 主要代码片段见 Code 4

train_gold_per_sentence 关键代码

```
prev2_label, prev_label = 0, 0
for pos in range(count_sum):

    softmax_score, feature_index_list, pred_index_list,\
    feature_vec, pred_matrix = self.predict_single_position(
            sentence, pos, prev2_label, prev_label,
            states=label_list)

    true_label = label_list[pos]
    if true_label == 0:
        gold_score = [1.0, 0.0, 1.0, 0.0]
    elif true_label == 1:
        gold_score = [0.0, 1.0, 0.0, 1.0]

    error_array = gold_score - softmax_score
    error_sum += sum(abs(error_array)) / len(error_array)
    gb = error_array * alpha
    neu1e = zeros(self.non_fixed_param)
    neu1e += dot(gb, pred_matrix[:, 0:self.non_fixed_param])

    if self.l2_rate:
        # l2 regularization
        self.syn1neg[pred_index_list] -=\
            alpha * self.l2_rate * self.syn1neg[pred_index_list]
        self.syn0[feature_index_list] -=\
            alpha * self.l2_rate * self.syn0[feature_index_list]

    # weight update
    # important code snippet
    # gb: list of length is 4
    self.syn1neg[pred_index_list] += outer(gb, feature_vec)
    self.syn0[feature_index_list] +=\
        neu1e.reshape(len(feature_index_list),
                      len(neu1e) / len(feature_index_list))

    softmax_score = softmax_score[-2:]
    if softmax_score[1] > 0.5:
        label = 1
    else:
        label = 0
    prev2_label = prev_label
    prev_label = label
    if self.use_gold:
        prev_label = true_label
```

Code 4: predict_single_position 关键代码

3.3 predict_sentence_greedy

predict_sentence_greedy 函数的 python doc 见 Code 5

predict_sentence_greedy

```
def predict_sentence_greedy(self, sent):
    """
    greedy predict sentence, used for test data
    :param sent: the sentence
    :return: segmented sentence, list of words
    """
```

Code 5: predict_sentence_greedy 方法

predict_sentence_greedy 主要代码片段见 Code 6

predict_sentence_greedy 关键代码

```
prev2_label, prev_label = 0, 0
for p, c in enumerate(old_sentence):
    # char is still the char from original sentence,
    #  for correct eval
    if p == 0:
        target = 0
    else:
        score_list, _, _, _, _ =\
            self.predict_single_position(
                sentence, p, prev2_label,
                    prev_label, states=states)

        if self.binary_pred:
            score_list = score_list[:2]
        elif self.hybrid_pred:
            old_char = old_sentence[p]
            if old_char in self.vocab and\
                            self.vocab[old_char].count\
                            > self.hybrid_threshold:
                score_list = score_list[-2:]
            else:
                # score_list = score_list[:2]
                x, y = score_list[:2], score_list[-2:]
                score_list = [(x[i] + y[i]) / 2.0 for i in range(2)]
        else:
            score_list = score_list[-2:]

        # transform score to binary target
        if score_list[1] > 0.5:
            target = 1
        else:
            target = 0
    # update the label in the current iter
    states[p] = target

    prev2_label = prev_label
    prev_label = target
```

Code 6: predict_sentence_greedy 关键代码

3.4 Call Hierarchy

predict_single_position 函数由 Code 7所列出的几个函数调用。

Call Hierarchy

```
segment_corpus(model, corpus, threashold=0) (code.predict)
Seger.train_gold_per_sentence(self, sentence, alpha, work=None) \
    (Seger in code.seg2)
Seger.predict_sentence_greedy(self, sent) (Seger in code.seg2)
```

Code 7: Call Hierarchy

4 算法改进

[1] 由从左到右单向模型改为双向模型。分词时同时考虑左边和右边字的状态。

双向具体实现过程主要有两种方法可以实现：

- ensemble two models
- using the labels of right characters

下面分别叙述。

4.1 Ensemble Method

该方式实现比较简单。就是从左向右训练出一个模型，然后从右往左训练一个模型，将这两个模型做一个 ensemble：对于每个字，将两个模型得出的两个 score 相加取平均数。根据平均后的 score 来确定每个字的 label。该方法没有改动训练过程，仅仅是整合两个不同模型的测试结果，实现相对简单。代码实现为 code 文件夹下的 predict_bidirect.py 文件。其主要实现见 Code 8

[1] 改进代码地址：https://github.com/junfenglx/emws/tree/dev

bidirection greedy predict

```
count = 0
seg_corpus = []

for sent_no, sent in enumerate(test_corpus):
    if sent_no % 100 == 0:
        print 'num of sentence segmented:', sent_no, '...'

    tokens = []
    if sent:
        forward_scores = segment_sentence(forward_model, sent)
        back_scores = segment_sentence(back_model, sent[::-1])
        back_scores = back_scores[::-1]

        # calculates scores
        # back_scores need right shit one to align forward_scores
        scores = np.zeros_like(forward_scores)
        scores[0] = forward_scores[0]
        for i in range(1, len(forward_scores)):
            scores[i] = (forward_scores[i] + back_scores[i-1]) / 2

        for pos, score in enumerate(scores):
            if score > 0.5:
                tokens[-1] += sent[pos]
            else:
                tokens.append(sent[pos])
        count += len(sent)
        """
        print(sent)
        print(forward_scores)
        print(back_scores)
        print(' '.join(tokens))
        """
    seg_corpus.append(tokens)
```

Code 8: bidirection greedy predict

4.2 使用右边字状态

增加一个配置选项：

```
no_right_action_feature = False
```

当其设置为 False 时，算法使用右边两个字的状态特征，反之则不使用右边字状态。

由于训练时从左向右扫描汉字，所以右边两个字的当前 label 无法预先知道，因此需要寻找一种机制来获取右边两个字的 label。

尝试了两种方案：

1. train 时，左边状态使用当前迭代，右边使用 gold standard labels
2. train 时，左右全部使用 gold standard labels，即使用 use_gold=1 开启该参数

测试时，由于并不知道 gold standard labels，使用多次迭代来计算 test sentences labels，初始时设置所有 labels 为 1, 表示未分词，经过多次迭代，得到 test sentences 的最终 labels。关键代码实现见 Code 9

iterate greedy predict

```
# initialize states vector
states = np.ones(len(sentence) + 2, dtype=np.int8)
states[0] = 0
states[-1] = 0
states[-2] = 0
if self.no_right_action_feature:
    do_greedy_predict()
else:
    for _ in range(self.iter):
        do_greedy_predict()
```

Code 9: iterate greedy predict

第一种训练方案，测试 score 达到 0.949, 其在第 5 次迭代后就达到了该水平。

第二种训练方案，测试 score 达到 0.948。

而当使用已有模型对 test 的输出的 labels 作为右边两字的 labels 时 (已有模型的 score 为 0.95)，两种训练方式 score 都能达到 0.951。这和整合两个模型取平均概率的 score 是一样的。