

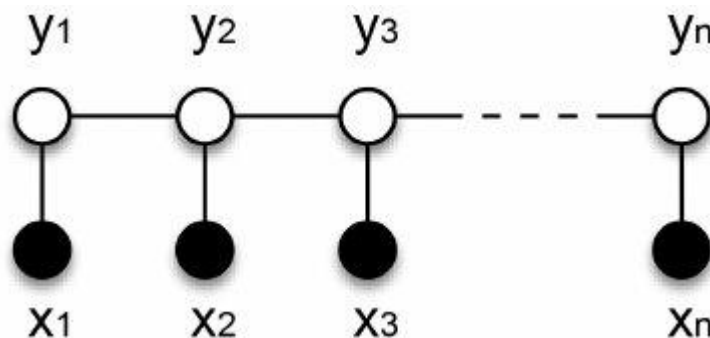
命名实体识别实践(一)-CRF

条件随机场-CRF

CRF，英文全称为Conditional Random Field, 中文名为条件随机场，是给定一组输入随机变量条件下另一组输出随机变量的条件概率分布模型，其特点是假设输出随机变量构成马尔可夫（Markov）随机场。

较为简单的条件随机场是定义在线性链上的条件随机场，称为线性链条件随机场（linear chain conditional random field）。

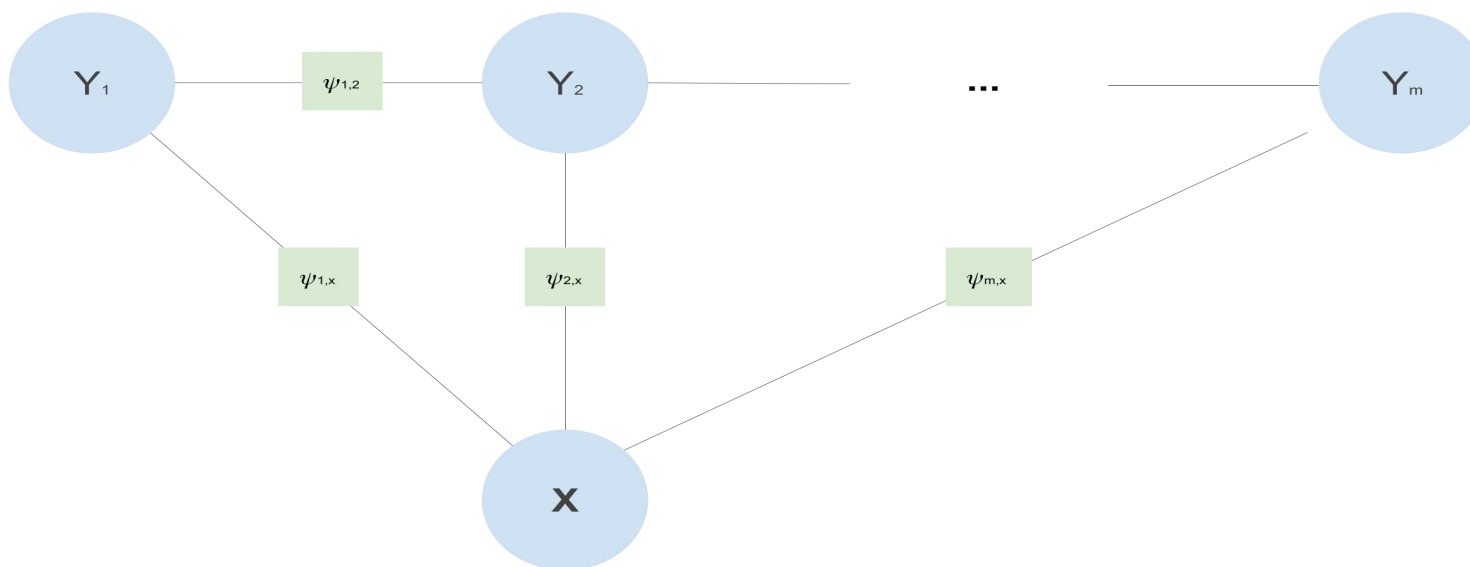
线性链条件随机场可以用于序列标注等问题，需要解决的命名实体识别(NER)任务正好可通过序列标注方法解决。



条件随机场-CRF

在条件概率模型 $P(Y|X)$ 中， Y 是输出变量，表示标记序列（或状态序列）， x 是输入变量，表示需要标注的观测序列。

- 训练时，利用训练数据集通过极大似然估计或正则化的极大似然估计得到条件概率模型 $p(Y|X)$ ；
- 预测时，对于给定的输入序列 x ，求出条件概率 $p(y|x)$ 最大的输出序列 y



实践1：基于CRF++实现NER

简介

CRF++是著名的条件随机场的开源工具，也是目前综合性能最佳的CRF工具，采用C++语言编写而成。其最重要的功能我认为是采用了特征模板。这样就可以自动生成一系列的特征函数，而不用我们自己生成特征函数，我们要做的就是寻找特征，比如词性等。

CRF++: Yet Another CRF toolkit

Introduction

CRF++ is a simple, customizable, and open source implementation of [Conditional Random Fields \(CRFs\)](#) for segmenting/labeling sequential data. CRF++ is designed for generic purpose and will be applied to a variety of NLP tasks, such as Na Chunking.

Table of contents













- Features
- News
- Download
- Source
- Binary package for MS-Windows
- Installation
- Usage
- Training and Test file formats
- Preparing feature templates
- Training (encoding)
- Testing (decoding)
- Case studies
- Useful Tips
- To do
- Links

官网地址：<http://taku910.github.io/crfpp/>

实践1： 基于CRF++实现NER

安装

CRF++的安装可分为Windows环境和Linux环境下的安装。关于Linux环境下的安装，可以参考文章：[CRFPP/CRF++编译安装与部署](#)。在Windows中CRF++不需要安装，下载解压CRF++0.58文件即可以使用

 chinese	2022/2/23 12:06	文件夹	
 doc	2022/2/23 10:23	文件夹	
 example	2022/2/23 10:23	文件夹	
 sdk	2022/2/23 10:23	文件夹	
 AUTHORS	2013/2/12 23:40	文件	1 KB
 BSD	2013/2/12 23:40	文件	2 KB
 COPYING	2013/2/12 23:40	文件	1 KB
 crf_learn.exe	2013/2/12 23:40	应用程序	50 KB
 crf_test.exe	2013/2/12 23:40	应用程序	50 KB
 LGPL	2013/2/12 23:40	文件	26 KB
 libcrfpp.dll	2013/2/12 23:40	应用程序扩展	330 KB
 README	2013/2/12 23:40	文件	1 KB

实践1： 基于CRF++实现NER

语料

在训练之前需要将标注数据转化为CRF++训练格式文件：

分两列，第一列是字符，第二例是对应的标签，中间用\t分割。

比如标注方案采用BISO， 效果如下：

浙	B_company
商	I_company
银	I_company
行	I_company
企	O
业	O
信	O
贷	O
部	O
叶	B_name
老	I_name
桂	I_name
博	O
士	O

实践1：基于CRF++实现NER

模板

模板是使用CRF++的关键，它能帮助我们自动生成一系列的特征函数，而不用我们自己生成特征函数，而特征函数正是CRF算法的核心概念之一。一个简单的模板文件如下：

```
# Unigram
U00:%x[-2,0]
U01:%x[0,1]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-2,0]/%x[-1,0]/%x[0,0]
U06:%x[-1,0]/%x[0,0]/%x[1,0]
U07:%x[0,0]/%x[1,0]/%x[2,0]
U08:%x[-1,0]/%x[0,0]
U09:%x[0,0]/%x[1,0]

# Bigram
B
```

实践1：基于CRF++实现NER

模板

在这里，我们需要好好理解下模板文件的规则。T**:%x[#,#]中的T表示模板类型，两个"#"分别表示相对的行偏移与列偏移。一共有两种模板：

played VBD O
on IN O
Monday NNP O
((O
home NN O << CURRENT TOKEN
team NN O
in IN O
CAPS NNP O
)) O
:: O



那么%x[#,#]的对应规则如下：

template	expanded feature
%x[0,0]	home
%x[0,1]	NN
%x[-1,0]	(
%x[-2,1]	NNP
%x[0,0]/%x[0,1]	home/NN
ABC%x[0,1]123	ABCNN123

以 “U01:%x[0,1]” 为例，它在该语料中生成的示例函数如下：

```
func1 = if (output = O and feature="U01:NN") return 1 else return 0
```


实践1：基于CRF++实现NER

训练

```
crf_learn -f 3 -c 4.0 -m 100 template train.data crf_model > train.rst
```

其中，template为模板文件，train.data为训练语料，-t表示可以得到一个model文件和一个model.txt文件，其他可选参数说明如下：

-f, -freq=INT 使用属性的出现次数不少于INT(默认为1)

-m, -maxiter=INT 设置INT为LBFGS的最大迭代次数 (默认10k)

-c, -cost=FLOAT 设置FLOAT为代价参数，过大会过度拟合 (默认1.0)

-e, -eta=FLOAT 设置终止标准FLOAT(默认0.0001)

-C, -convert 将文本模式转为二进制模式

-t, -textmodel 为调试建立文本模型文件

-a, -algorithm=(CRF|MIRA) 选择训练算法，默认为CRF-L2

-p, -thread=INT 线程数(默认1)，利用多个CPU减少训练时间

-H, -shrinking-size=INT 设置INT为最适宜的迭代变量次数 (默认20)

-v, -version 显示版本号并退出

-h, -help 显示帮助并退出

输出信息

- iter：迭代次数。当迭代次数达到maxiter时，迭代终止
- terr：标记错误率
- serr：句子错误率
- obj：当前对象的值。当这个值收敛到一个确定值的时候，训练完成
- diff：与上一个对象值之间的相对差。当此值低于eta时，训练完成

实践1：基于CRF++实现NER

预测

在训练完模型后，我们可以使用训练好的模型对新数据进行预测，预测命令格式如下：

```
crf_test -m crf_model test.data > test.rstt
```

-m model表示使用我们刚刚训练好的model模型，预测的数据文件为test.data> test.rstt 表示将预测后的数据写入到test.rstt 中。

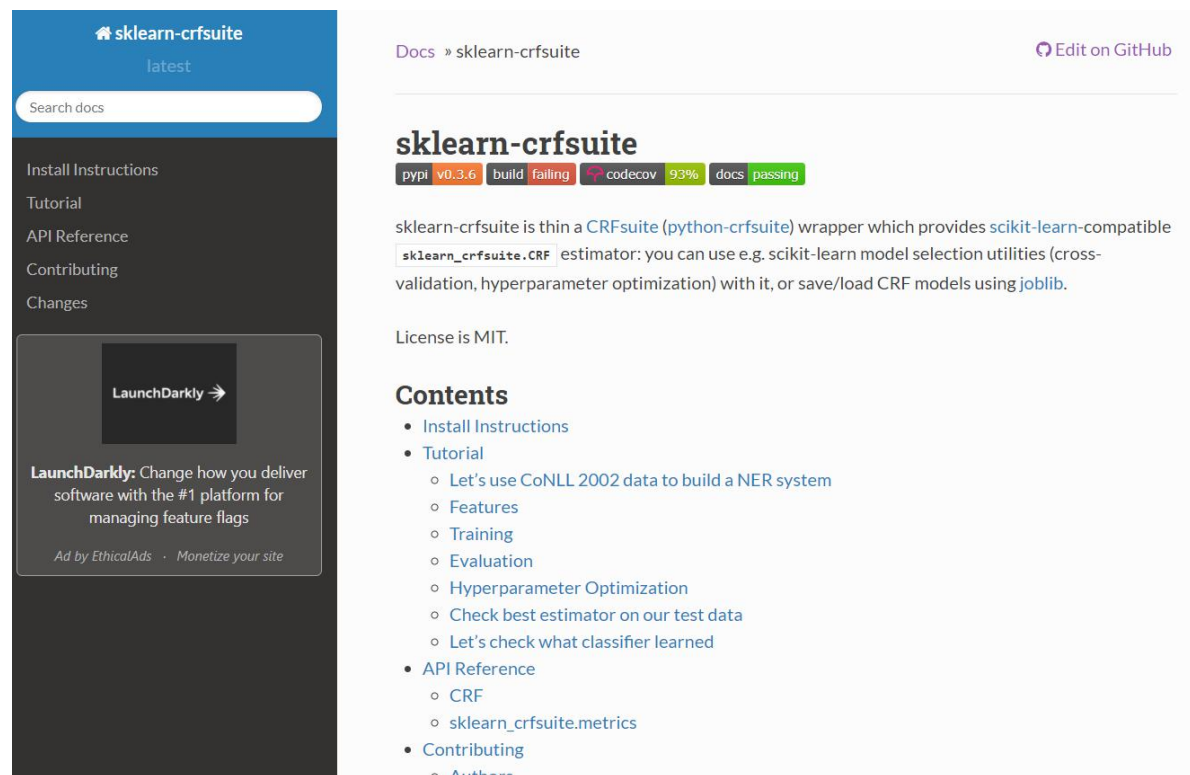
```
177208 家 0 0
177209 庭 0 0
177210 观 0 0
177211 念 0 0
177212 的 0 0
177213 希 B-LOC B-LOC
177214 腊 I-LOC I-LOC
177215 人 0 0
177216 ， 0 0
177217 对 0 0
177218 瓦 B-PER B-LOC
177219 西 I-PER I-LOC
177220 里 I-PER I-LOC
177221 斯 I-PER I-LOC
177222 幸 0 I-LOC
177223 福 0 I-LOC
177224 的 0 0
177225 家 0 0
177226 庭 0 0
177227 充 0 0
177228 满 0 0
177229 赞 0 0
177230 誉 0 0
177231 。 0 0
177232
```

实践2：基于sklearn_crfsuite实现NER

简介

sklearn-crfsuite是基于CRFsuite库的一款轻量级的CRF库。该库兼容sklearn的算法，因此可以结合sklearn库的算法设计实体识别系统。sklearn-crfsuite不仅提供了条件随机场的训练和预测方法还提供了评测方法。

<https://sklearn-crfsuite.readthedocs.io/en/latest/#>



sklearn-crfsuite

latest

Search docs

Install Instructions

Tutorial

API Reference

Contributing

Changes

LaunchDarkly ➔

LaunchDarkly: Change how you deliver software with the #1 platform for managing feature flags

Ad by EthicalAds · Monetize your site

Docs » sklearn-crfsuite

Edit on GitHub

sklearn-crfsuite

pypi v0.3.6 build failing codecov 93% docs passing

sklearn-crfsuite is thin a [CRFsuite](#) ([python-crfsuite](#)) wrapper which provides [scikit-learn](#)-compatible `sklearn_crfsuite.CRF` estimator: you can use e.g. [scikit-learn](#) model selection utilities (cross-validation, hyperparameter optimization) with it, or save/load CRF models using [joblib](#).

License is MIT.

Contents

- [Install Instructions](#)
- [Tutorial](#)
 - [Let's use CoNLL 2002 data to build a NER system](#)
 - [Features](#)
 - [Training](#)
 - [Evaluation](#)
 - [Hyperparameter Optimization](#)
 - [Check best estimator on our test data](#)
 - [Let's check what classifier learned](#)
- [API Reference](#)
 - [CRF](#)
 - [sklearn_crfsuite.metrics](#)
- [Contributing](#)
 - [Authors](#)

安装： `pip install sklearn-crfsuite`

实践2：基于sklearn_crfsuite实现NER

使用

```
def word2features(sent, i):
    """抽取单个字的特征"""
    word = sent[i]
    prev_word = "<s>" if i == 0 else sent[i-1]
    next_word = "</s>" if i == (len(sent)-1) else sent[i+1]
    # 使用的特征：
    # 前一个词，当前词，后一个词，
    # 前一个词+当前词， 当前词+后一个词
    features = {
        'w': word,
        'w-1': prev_word,
        'w+1': next_word,
        'w-1:w': prev_word+word,
        'w:w+1': word+next_word,
        'bias': 1
    }
    return features
```

```
crf_model =
sklearn_crfsuite.CRF(algorithm='lbfgs',c1=0.25,c2=0.018,m
ax_iterations=100,

all_possible_transitions=True,verbose=True)
crf_model.fit(X_train, y_train)
```

参考资料

条件随机场CRF及CRF++安装与使用

<https://www.biaodianfu.com/crf.html>

使用CRF++实现命名实体识别(NER)

<https://www.cnblogs.com/jclian91/p/10795413.html>

利用crf++进行实体识别

<https://www.jianshu.com/p/f5868fdd96d2>