

# Beyond Accuracy: Behavioral Testing of NLP Models with CheckList

年份：2020

会议：ACL best paper

机构：微软

理解：传统模型测试使用held-out(留出法)分割的测试集进行单一指标的性能测试，存在如下问题：

1. 测试集与训练集有着相同的bias（领域性/分布），导致过高的估计模型
2. 测试指标单一，不能精确发现模型问题所在

于是，本文提出CheckList使用软件工程的方法进行NLP任务的测试，有如下好处：

1. 能够不区分NLP任务，不用针对性的使用某种测试指标进行测试
2. 能对NLP模型进行多种基础能力及不同类型的测试
3. 能够精确的发现模型在某种能力上存在的缺陷，并给出详细分析
4. 提供便捷的工具用以完成以上操作

CheckList可能存在如下问题：

1. 如何构造测试用例？对模型的多种基础能力及不同类型进行测试时，需要自行构造测试用例，测试用例构造策略的好坏，可能决定了测试的有效程度，所以如何构造测试用例存在较强的主观性，因人而异
2. 当前CheckList中很多内置方法和模型对其他语言（如中文）的支持性可能没有那么完善

## CheckList

首先，checklist是一种针对NLP模型的测试工具，如何安装及使用见如下链接：

<https://github.com/marcotcr/checklist>

## CheckList的多种能力测试

用户通过checklist提供的便捷构造数据的模版，针对某种能力构造专门的测试数据，然后对该能力进行测试

- Vocabulary: 是否掌握词汇及词性
- Robustness: 是否能应对拼写错误和噪音
- Taxonomy: 同义词、反义词理解能力
- NER: 是否能识别相关命名实体
- Coreference: 指代关系
- SRL(Semantic role labeling): 是否能理解各种角色

- Logic: 是否能处理连接词、对称性等
- ...

例如, 要对模型进行Vocabulary能力测试, 需要使用**checklist**提供的数据构造方法, 自主进行测试数据构造, 然后进行Vocabulary能力测试

## 每种能力的不同类型测试

---

针对某种能力, CheckList又做了三种不同类型的测试, 分别是:

- MFT: 最小功能测试
  - 利用上述构造好的测试数据进行常规单元测试
- INV: 不变性测试
  - 对数据添加较小扰动, 再次进行测试, 期望结果保持不变 (过敏感测试)
- DIR: 定向期望测试
  - 对数据添加较小扰动, 再次进行测试, 期望结果改变 (过稳定测试)

例如, 对模型进行Vocabulary能力测试的时候, 可分为进行上述三种类型的测试:

- 直接使用构造好的测试数据进行MFT测试
- 对测试数据使用Checklist提供的方法替换一些单词 (替换规则要自主设计), 期望结果不变, 进行INV测试
- 对测试数据使用Checklist提供的便捷方法替换一些单词 (替换规则要自主设计), 期望结果改变, 进行DIR测试

## 如何使用CheckList

---

进行CheckList对NLP模型进行测试的必要条件:

1. NLP模型
2. 构造CheckList测试数据
3. 编写CheckList测试代码

### 1. 从头开始对情感分类模型进行CheckList测试

我们仅对情感分类模型进行Negative能力的MFT测试

```
1  """Negative否定词能 - MFT测试"""
2  import numpy as np
3  import random
4  import checklist
5  from checklist.editor import Editor
6  from checklist.test_types import MFT
```


```

7 from checklist.pred_wrapper import PredictorWrapper
8
9 """利用模版构造少量测试数据"""
10 editor = Editor()
11
12 # 模版字典
13 pos = ['好', '伟大', '优秀', '完美', '友善']
14 neg = ['坏', '猥琐', '傻傻', '笨笨', '愚蠢']
15 something = ['男人', '女人', '狗', '猪', '学生', '教授', '领导']
16
17 # 模版数据初始化
18 ret = editor.template('这不是一个{pos}的{something}', pos=pos,
19 something=something, labels=0, save=True, nsamples=20)
20 ret += editor.template('这不是一个{neg}的{something}', neg=neg,
21 something=something, labels=1, save=True, nsamples=20)
22
23 # MFT测试模块初始化
24 test = MFT(ret.data, labels=ret.labels, name='Simple
25 negation', capability='Negation', description='Very simple negations.')
26
27 """加载情感分类模型(用随机函数充当随机情感分类模型)"""
28 # 情感分类随机模型
29 def predict_proba(inputs):
30     p1 = np.array([ random.random() for x in inputs]).reshape(-1, 1)
31     p0 = 1- p1
32     return np.hstack((p0, p1))
33
34 # 初始化checklist模型容器
35 wrapped_pp = PredictorWrapper.wrap_softmax(predict_proba)
36
37 # 开始测试
38 test.run(wrapped_pp, overwrite=True)
39 test.visual_summary()

```

Predicting 40 examples

#### Test Summary

Test [MFT] on [NEGATION] Simple negation
Desc. Very simple negations.
Result <b>FAILURE RATE ON ALL CASES</b> 14/40=35.0% 
<b>FILTER TEST CASES</b> <input type="text" value="Input free text and enter"/>

#### Examples Failed cases only

> 这不是一个友善的女人	Expect 0   Pred 1 (0.73)	✗
> 这不是一个优秀的狗	Expect 0   Pred 1 (0.91)	✗
> 这不是一个友善的领导	Expect 0   Pred 1 (0.71)	✗
> 这不是一个完美的教授	Expect 0   Pred 1 (0.58)	✗
> 这不是一个伟大的狗	Expect 0   Pred 1 (0.54)	✗
> 这不是一个完美的女人	Expect 0   Pred 1 (0.84)	✗
> 这不是一个好的女人	Expect 0   Pred 1 (0.91)	✗

```

1  """我们可以把测试数据/测试结果存储起来，方便下次调用"""
2  # 存储测试数据
3  test.to_raw_file('raw_test_file.txt')
4  # 存储测试结果
5  docs = open('raw_test_file.txt').read().splitlines()
6  preds = predict_proba(docs)
7  f = open('/tmp/softmax_preds.txt', 'w')
8  for p in preds:
9      f.write('%f %f\n' % tuple(p))
10 f.close()

```

## 2. 官方情感分析模型CheckList DEMO

官方已经针对情感分类任务构造好了CheckList各种测试数据，并使用BERT等模型预测好了结果，所以官方的Demo直接读取文件并且展示即可

```

1  import checklist
2  from checklist.test_suite import TestSuite
3
4  # 加载测试数据
5  suite_path = 'sentiment_suite.pkl'
6  suite = TestSuite.from_file(suite_path)
7  # 加载存储好的测试结果
8  pred_path = 'bert'
9  suite.run_from_file(pred_path, overwrite=True)
10 # 展示checklist结果
11 suite.visual_summary_table()

```

本次测试，针对基于BERT的情感分类模型进行了Vocabulary、Robustness等能力测试，并且针对某些能力做了不同类型的测试，其中百分比数值代表错误率。

Capabilities	Minimum Functionality Test failure rate % (over N tests)	INVariance Test failure rate % (over N tests)	DIRectional Expectation Test failure rate % (over N tests)
+ Vocabulary	100.0% (5)	10.2% (1)	0.8% (4)
+ Robustness		11.4% (5)	
+ NER		7.6% (3)	
+ Fairness		96.4% (4)	
+ Temporal	18.8% (1)		100.0% (1)
+ Negation	99.8% (9)		
+ SRL	100.0% (5)		

同样，具体能力测试可以点开查看详情，进行具体分析

MINIMUM FUNCTIONALITY TEST

	test name	failure rate
+	single positive words	0 / 34 = 0.0% <div></div>
+	single negative words	0 / 35 = 0.0% <div></div>
-	single neutral words	13 / 13 = 100.0% <div></div>
<div><div><div>Test Summary</div><div><div>Test [MFT] on [VOCABULARY] single neutral words</div><div>Desc. T0D0_DESCRIPTION</div><div>Result <b>FAILURE RATE ON ALL CASES</b> 13/13=100.0% <div></div></div><div><b>FILTER TEST CASES</b> <div>Input free text and enter</div></div></div><div><div>Examples <b>Failed cases only</b></div><div><div>&gt; commercial</div><div>Expect 1   Pred 0 (0.91) <div></div></div></div><div><div>&gt; British</div><div>Expect 1   Pred 2 (1.00) <div></div></div></div><div><div>&gt; private</div><div>Expect 1   Pred 2 (1.00) <div></div></div></div><div><div>&gt; find</div><div>Expect 1   Pred 2 (1.00) <div></div></div></div><div><div>&gt; international</div><div>Expect 1   Pred 2 (1.00) <div></div></div></div><div><div>&gt; Indian</div><div>Expect 1   Pred 2 (0.99) <div></div></div></div></div></div></div>		
+	Sentiment-laden words in context	0 / 500 = 0.0% <div></div>
+	neutral words in context	473 / 500 = 94.6% <div></div>