

## 自然语言通顺判定 文档：

### 运行环境：

python 3.6 tensorflow 1.12.0






输入：一句话（中文，可包括标点）

输出：判断这句话是否通顺的分类（0：通顺，1：不通顺）

### 任务描述：

自然语言通顺与否的判定，即给定一个句子，要求判定所给的句子是否通顺。

### 问题分析：

Model	URL	Score	CoLA
Microsoft D365 AI & MSR AI		81.9	61.5
BERT: 24-layers, 1024-hidden, 16-head		80.4	60.5
GPT on STILTs		76.9	47.2
Singletask Pretrain Transformer		72.8	45.4
BiLSTM+ELMo+Attn		70.5	36.0
BiLSTM+ELMo+Attn		68.9	18.9

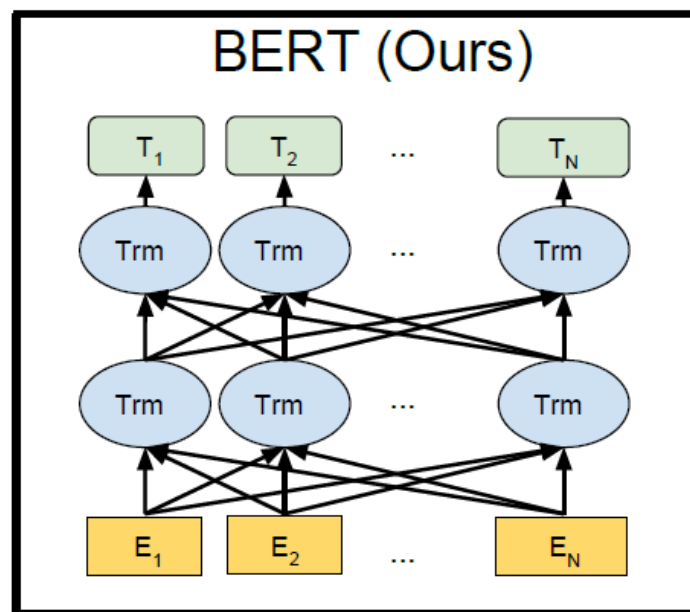
### GLUE 排行榜

在 GLUE 中有一个自然语言处理任务 CoLA，也是判断自然语言是否通顺，我们可以看到在 cola 上 Google 最新发布的基于 transformer 和 mask 的 embedding 取得了第二的成绩，所以在本次的任务中使用 bert 也是合适的，但是原数据集中噪音很多，在经过基本的降噪过程之后（去掉一些非中文的训练样本，乱码等等），还是保留有大量噪音，可能会对之后的结果造成影响。

## 技术路线：

使用基于 bert 的自然语言通顺判定：

Bert：



Bert 是 google 新提出的一种 embedding 方法，基于 pre-train 好的模型，在下游的 nlp 任务中只需要 fine-tuning 少部分轮数即可获得很好的效果，其具体思路是：

- 采取新的预训练的目标函数：the “masked language model” (MLM) 随机 mask 输入中的一些 tokens，然后在预训练中对它们进行预测。这样做的好处是学习到的表征能够融合两个方向上的 context。这个做法我觉得非常像 skip-gram。过去的同类算法在这里有所欠缺，比如上文提到的 ELMo，它用的是两个单向的 LSTM 然后把结果拼接起来；还有 OpenAI GPT，虽然它一样使用了 transformer，但是只利用了一个方向的注意力机制，本质上也一样是单项的语言模型。
- 增加句子级别的任务：“next sentence prediction”  
作者认为很多 NLP 任务比如 QA 和 NLI 都需要对两个句子之间关系的理解，而语言模型不能很好的直接产生这种理解。为了理解句子关系，作者同时 pre-train 了一个“next sentence prediction”任务。具体做法是随机替换一些句子，然后利用上一句进行 IsNext/NotNext 的预测。

在本次试验中使用的是最近 bert 预训练出的模型 **Bert-Base Chinese**，基于中文的 bert 与训练模型。

我们只需修改其 run\_classifier.py 文件，将其输入 example 修改为 texta 为训练样本，textb None（无需 textb），label 在训练集中为训练样本的 label，在测试集中为 0（这样测试出来的即为 0 的概率），然后开始 fine-tuning 即可，**A 榜中的模型 fine-tuning 了 25 个 epoch，B 榜中的为 3 个 epoch**，然后加入一些人工的先验知识（trick），如：含有较多非中文编码的句子应该不通顺，然后设置阈值即可生成最终的测试结果。

在 Bert 的实现中, 其实是在 bert 的输出加入了一个 dropout 之后接入一个 softmax 层, 其输出为需要的 label 的概率。

```
class HandleNLPProcessor(DataProcessor):
    """Processor for the NLP data set (GLUE version)."""
    def get_train_examples(self, data_dir):
        return self.read_data(data_dir, "train.txt")

    def get_test_examples(self, data_dir):
        return self.read_data(data_dir, "test_v3.txt")

    def get_labels(self):
        """See base class."""
        return ["0", "1"]

    def read_data(self, data_dir, is_train):
        file_path = os.path.join(data_dir, is_train)
        if "train" in is_train:
            with open(file_path, 'r', encoding="utf-8") as f:
                reader = f.readlines()
                examples = []
                for index, line in enumerate(reader):
                    guid = 'train-%d' % index
                    split_line = line.strip().split('\t')
                    text_a = tokenization.convert_to_unicode(split_line[1])
                    label = split_line[2]
                    examples.append(InputExample(guid=guid, text_a=text_a,
                                                  text_b=None, label=label))
            else:
                with open(file_path, 'r', encoding="utf-8") as f:
                    reader = f.readlines()
                    examples = []
                    for index, line in enumerate(reader):
                        guid = 'test-%d' % index
                        split_line = line.strip().split('\t')
                        text_a = tokenization.convert_to_unicode(split_line[1])
                        label = "0"
                        examples.append(InputExample(guid=guid, text_a=text_a,
                                                  text_b=None, label=label))
        return examples
```

修改的输入输出

运行情况:

12	MG1833038	0.868375	49	2019-01-09 14:06:32
----	-----------	----------	----	---------------------

A 榜

排名	学号	F1 Score	有效提交次数	提交时间
1	MG1833038	0.830161	41	2019-01-15 13:51:49

B 榜

注: 提交的代码中为 run\_classifier.py, 训练方法在 run.txt 中第一行为 train, 第二行为 test, 最终结果为 result.txt, 最后提交的结果由 posthandler.py 生成