# NMT实验记录

by 王强

2016/10/13

---



# 一、实验环境

- OS: CentOS 6.5
- Deep Learning Framework: Theano 0.9.0
- GPU: GeForce GTX 1080, 8G
- Cuda: v8.0
- Cudnn: v7.5

---

# 二、语言方向

> - 中文 - 英文

# 三、实验数据

## 1. 来源

> - 训练集：排序过的数据 `01`~`07`
> - 校验集：`mt06`
> - 测试集：`mt04`、`mt05`、`mt08`

## 2. 处理

### (1) 转 UTF-8 编码

```
iconv -c -f gbk -t utf8  < 【输入文件】 > 【输出文件】
```

> - `-c` 作用是忽略无效字符，不加的话会报错【iconv: 未知 *** 处的非法输入序列】

### (2) 分词

> - 训练集：`泛化`、`不翻译`
> - 校验集：`泛化`、`不翻译`
> - 测试集：`泛化`、`不翻译`

### (3) 非法字符过滤

```
perl NiuTrans-clear.illegal.char.pl -src 【输入源语文件】 -tgt 【输入目标语文件】 -outSrc 【清洗后的源语文件】 -outTgt 【清洗后的目标语文件】
```

### (4) 长度比过滤

```
perl NiuTrans-length.ratio.filter.pl -src 【输入源语文件】 -tgt 【输入目标语文件】 -outSrc 【过滤后的源语文件】 -outTgt 【过滤后的目标语文件】 -lengthRestrict 【50】
```

> - `-lengthRestrict` 作用是限制句子的最大长度（原脚本是限制源语的最大长度，被我改成也包括目标语的最大长度），这里使用的参数是`50`

## 3. 统计

### (1) 句子数&词汇数

| 语种 | 训练集(句子数/词汇数/平均句长) | 校验集**mt06**(句子数/词汇数) | 测试集**mt04** | 测试集**mt05** | 测试集**mt08** |
|---|---|---|---|---|---|
| 中文 | 185W/3942W/21 | 1664/37136 | 1788/47735 | 1082/29271 | 1357/31505 |
| 英文 | 185W/4496W/24 | - | - | - | - |

### (2) 词典覆盖度

| 统计(3万词) | 训练集 | 校验集**mt06** | 测试集**mt04** | **mt05** | **mt08** |
|---|---|---|---|---|---|
| OOV句子比例 | 23.98%/14.67% | - | - | - | - |
| 词汇表覆盖度 | 98.16%/99.17% | - | - | - | - |
| 词汇表总大小 | 159784/150259 | - | - | - | - |

# 四、实验设置

## 1. SMT

| 参数名称 | 参数值 |
|---|---|
| 翻译规则长度 | `5-5` |
| 语言模型元数 | `5`，仅用训练数据目标语部分 |
| 调序样本 | `8000万` |
| MERT | |

## 2. NMT

| 参数名称 | 参数值 |
| --- | --- |
| 词汇表大小 | 源语`3万`，目标语`3万` |
| 词向量大小 | `500` |
| 递归隐藏层大小 | `1000` |
| Batch大小 | `80` |
| Dropout | embedding=`0.2` hidden=`0.2` source=`0.1` target=`0.1` |
| 学习率 | 0.0001 |
| 优化器 | AdaDelta |
| 梯度Clipping | 1 |
| L2 | 0 |
| Attention decay | 0 |
| 校验指标 | BLEU |
| 校验频率 | 1万（处理完1万个batch） |
| 校验起始次数 | 8万（前8万次更新过程中不进行校验） |
| 模型保存频率 | 3万 |

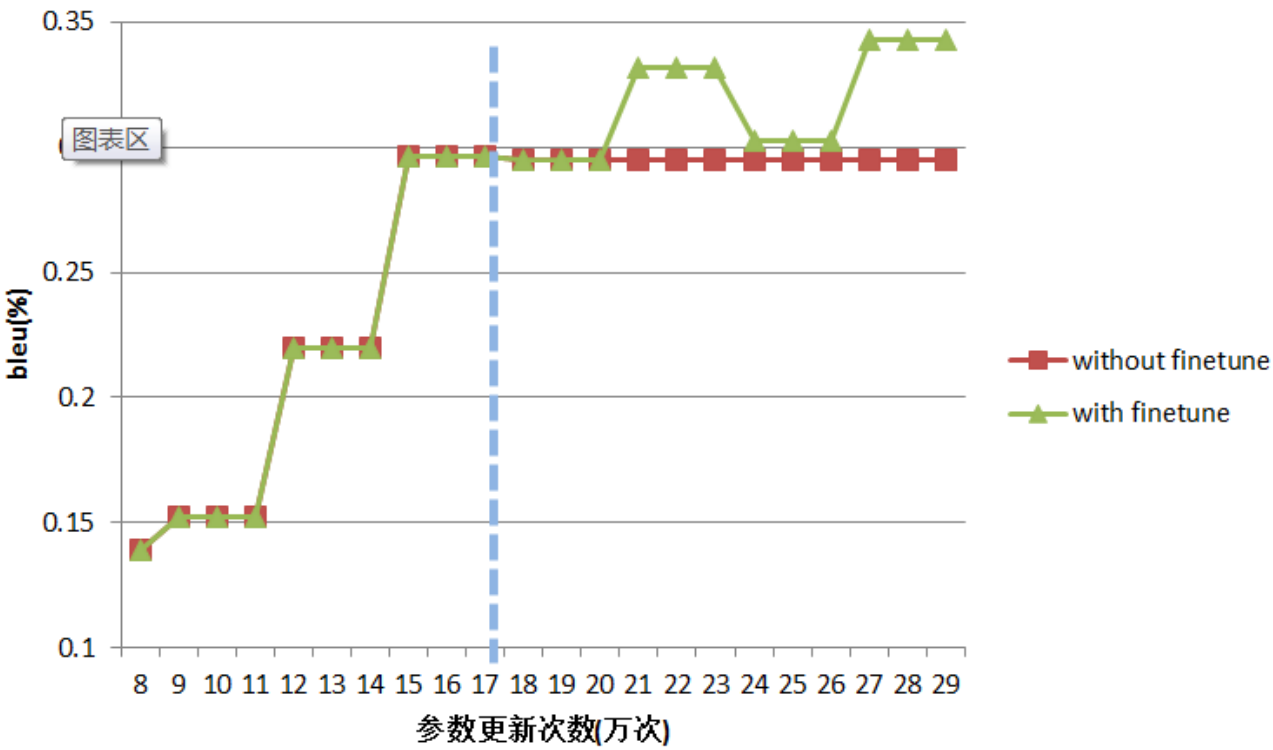
# 五、实验结果

## 1. 翻译性能

| 编号 | 名称 | **mt06** | **mt04** | **mt05** | **mt08** |
|---|---|---|---|---|---|
| 1 | NiuTrans(baseline) | 32.09 | 36.65 | 31.3 | 25.99 |
| 2 | NiuTrans + 线上语言模型 | 33.87(+1.78) | 38.18(+1.53) | 32.76(+1.46) | 27.86(+1.87) |
| 3 | NMT | 29.64 | 33.89 | 28.56 | 23.06 |
| 4 | 3 + finetune | 34.27(+2.1) | - | - | - |
| 5 | 4 + replace_unk | 34.92(+2.83) | 41.11(+4.46) | 34.3(+3) | 27.1(+1.11) |
| 6 | 5 + ensemble (4 models) | **37.32(+5.23)** | 43.46(+6.81) | **35.91(+4.61)** | 29.32(+4.33) |
| 7 | 4 - dropout 跟 4 比 | 33.70(-0.57) | - | - | - |
| 8 | 7 - dropout 跟 5 比 | 34.33(-0.59) | 39.78(-1.33) | 33.25(-1.05) | 25.86(-1.24) |
| 9* | 5 + avg ens(4 models) 跟 6 比 | 36.88(-0.44) | **43.60(+0.14)** | 35.74(-0.17) | **29.42(+0.1)** |

## 2. 现象

### (1) 训练过程中校验集**BLEU**变化情况



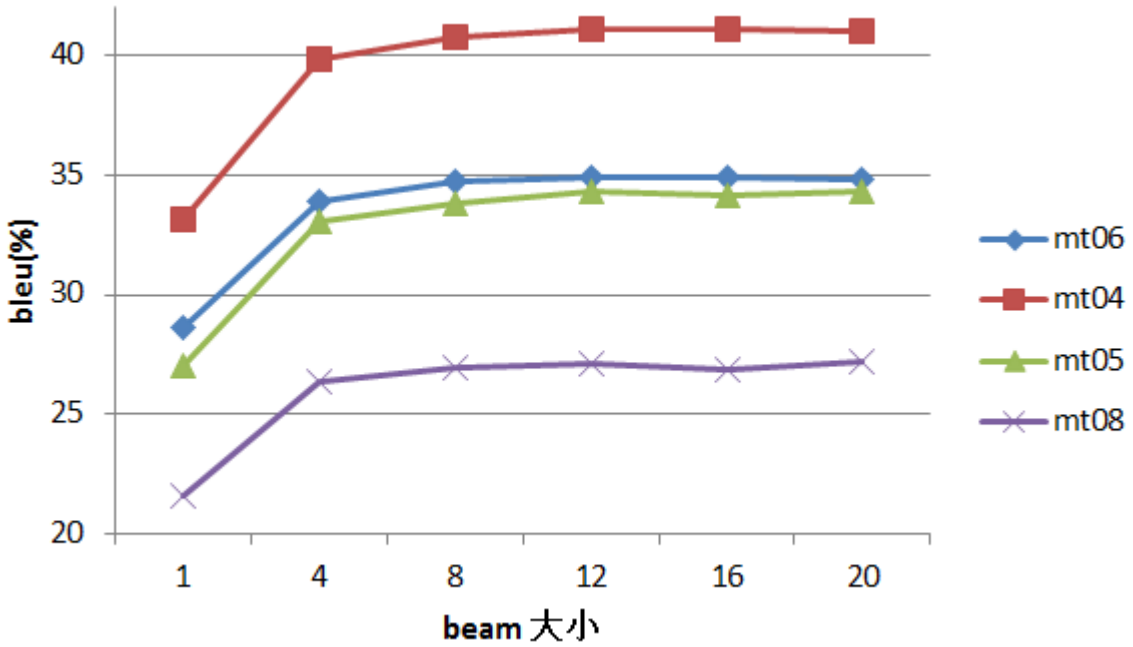

> - 在训练过程中，Bleu是阶跃式的变化，不连续
> - **Finetune**是提高性能的关键步骤，本实验中比未Finetune的NMT在检验集上Bleu提高4.6

| 更新次数(万次) | **w/o finetune (bleu)** | **w/ finetune (bleu)** |
|---|---|---|
| 8 | 0.1392 | 0.1392 |
| 9 | 0.1526 | 0.1526 |
| 10 | 0.1526 | 0.1526 |
| 11 | 0.1526 | 0.1526 |
| 12 | 0.2197 | 0.2197 |
| 13 | 0.2197 | 0.2197 |
| 14 | 0.2197 | 0.2197 |
| 15 | 0.2964 | 0.2964 |
| 16 | 0.2964 | 0.2964 |
| 17 | 0.2964 | 0.2964 |
| **18**(从这开始Finetune) | 0.2948 | 0.2948 |
| 19 | 0.2948 | 0.2948 |
| 20 | 0.2948 | 0.2948 |
| 21 | 0.2948 | 0.3322 |
| 22 | 0.2948 | 0.3322 |
| 23 | 0.2948 | 0.3322 |
| 24 | 0.2948 | 0.3027 |
| 25 | 0.2948 | 0.3027 |
| 26 | 0.2948 | 0.3027 |
| 27 | 0.2948 | 0.3427 |
| 28 | 0.2948 | 0.3427 |
| 29 | 0.2948 | 0.3427 |

## **(2)** 训练速度

> - 大约5个小时训练一个epoch
> - 一个epoch大约需要更新23000次（也就是包含大约23000 batch）
> - 一个batch包含80个样本
> - 平均每秒处理100个sample，翻译速度约**2000**词/秒

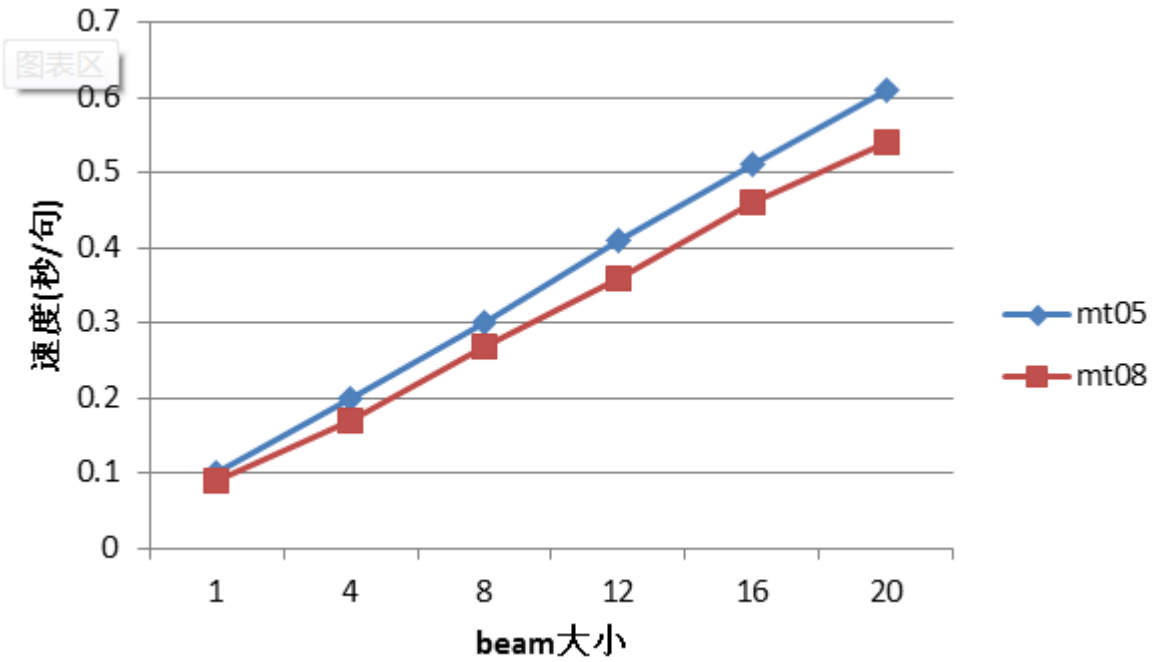
## **(3) Beam Size**对**BLEU**的影响




> - 随着beam size的增大，**BLEU**先增大，而后趋于稳定
> - 一般，beam size设置**8~20**之间
> - greedy search效果很差，比SMT的baseline还低

| Beam | mt06 | mt04 | mt05 | mt08 |
|---|---|---|---|---|
| 1 | 28.65 | 33.15 | 27.05 | 21.6 |
| 4 | 33.93 | 39.84 | 33.06 | 26.38 |
| 8 | 34.7 | 40.74 | 33.82 | 26.93 |
| 12 | 34.92 | 41.11 | 34.3 | 27.1 |
| 16 | 34.87 | 41.04 | 34.1 | 26.86 |
| 20 | 34.84 | 40.99 | 34.32 | 27.2 |

## (4) Beam Size对速度的影响(GPU)

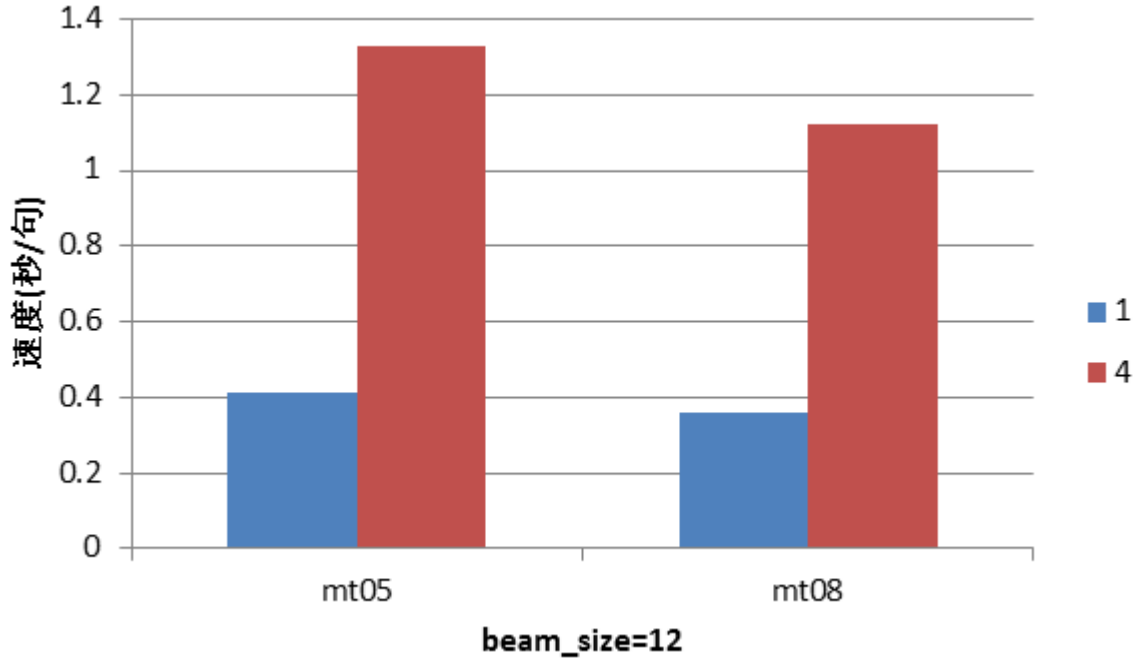



- 解码速度同beam size大小线性相关

| Beam | mt05 (秒/句) | mt08(秒/句) |
|---|---|---|
| 1 | 0.1 | 0.09 |
| 4 | 0.2 | 0.17 |
| 8 | 0.3 | 0.27 |
| 12 | 0.41 | 0.36 |
| 16 | 0.51 | 0.46 |
| 20 | 0.61 | 0.54 |

## (5) Ensemble Decoding对速度的影响

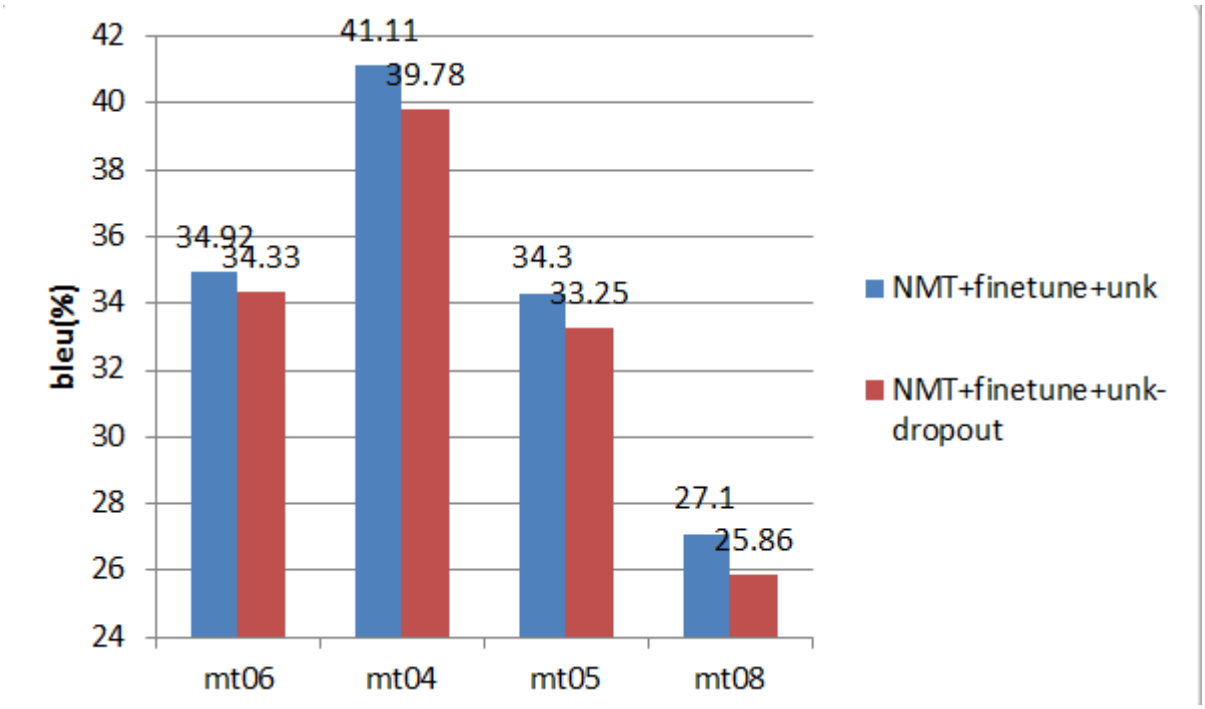
`beam_size`=12时，single model 和 ensemble model的速度比较




- 解码的单句耗时同ensemble的模型数目成正比

| Model Number | mt05(秒/句) | mt08(秒/句) |
|---|---|---|
| 1 | 0.41 | 0.36 |
| 4 | 1.33 | 1.12 |

## (6)Dropout的影响




- 不开Dropout在测试集上bleu稳定的下降1个点多

## (7) Parameter Ensemble VS. Probability Ensemble

`Parameter Ensemble`指把多个模型的参数取平均，得到的一个平均的模型，解码时只使用这一个平均模型

`Probability Ensemble`指多个模型一起参与解码，解码过程中使用多个模型分别给出概率值，对各个概率取平均（实际应用的是算术平均值），作为最终的概率

详见$5.1 行`6`和行`9`的对比

> - `Parameter Ensemble`能够获得与`Probability Ensemble`相当的性能
> - 但是由于只使用一个model，`Parameter Ensemble`解码时间同single model一样；而`Probability Ensemble`则是同集成的Model数线性相关
> - 推荐以后使用`Parameter Ensemble`

### (8) Parameter Ensemble 使用的模型数对BLEU的影响

| 模型数 | **mt06** | **mt04** | **mt05** | **mt08** |
|---|---|---|---|---|
| 0(baseline) | 34.92 | 41.11 | 34.30 | 27.10 |
| 2 | 35.27(+0.35) | 42.08(+0.97) | 34.16(-0.14) | 28.05(+0.95) |
| 4 | 36.88(+1.96) | 43.60(+2.49) | 35.74(+1.43) | 29.42(+2.32) |
| 6 | 37.40(+2.48) | 43.75(+2.64) | 35.90(+1.60) | 29.46(+2.36) |

> - 用Parameter Ensemble的方法做模型集成，一般使用4~8个模型就行

## 3. 不同的Segmentation

`w2w`表示源语是词，目标语是词

`c2w`表示源语是字，目标语是词

`sw2sw`表示源语是子词，目标语是子词

| 方法 | **mt06** | **mt04** | **mt05** | **mt08** |
|---|---|---|---|---|
| w2w + rep_unk (baseline) | 34.92 | 41.11 | 34.3 | 27.1 |
| c2w + rep_unk | 34.41(-0.51) | 39.39(-1.72) | 32.23(-2.07) | 26.36(-0.74) |
| sw2sw + rep_unk | | | | |

# 六、翻译结果分析

TO DO

# 七、总结

TO DO