



北京大学
PEKING UNIVERSITY

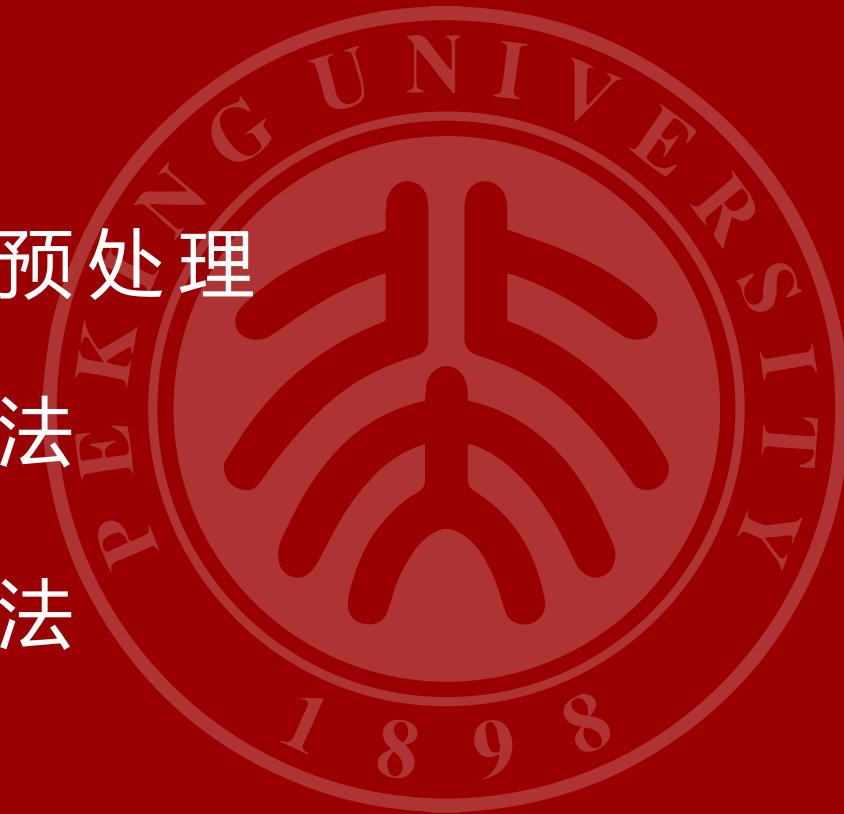
自然语言处理项目汇报

组员：罗旭坤 罗登 马心睿
日期：2021年5月7日

目录

CONTENTS

- 01 问题描述
- 02 数据分析与预处理
- 03 机器学习方法
- 04 深度学习方法
- 05 总结与展望



问题描述

- 设计模型，对SIGHAN 2005第二届中文分词任务中的PKU数据集进行分词

训练集	测试集	Type数	Token
19056	1944	55303	1,109,947

- 提供了**training.txt**、**test.txt**和**vocab.txt**文件，全角字符，utf-8编码
- 词与词、词与标点之间由两个空格隔开

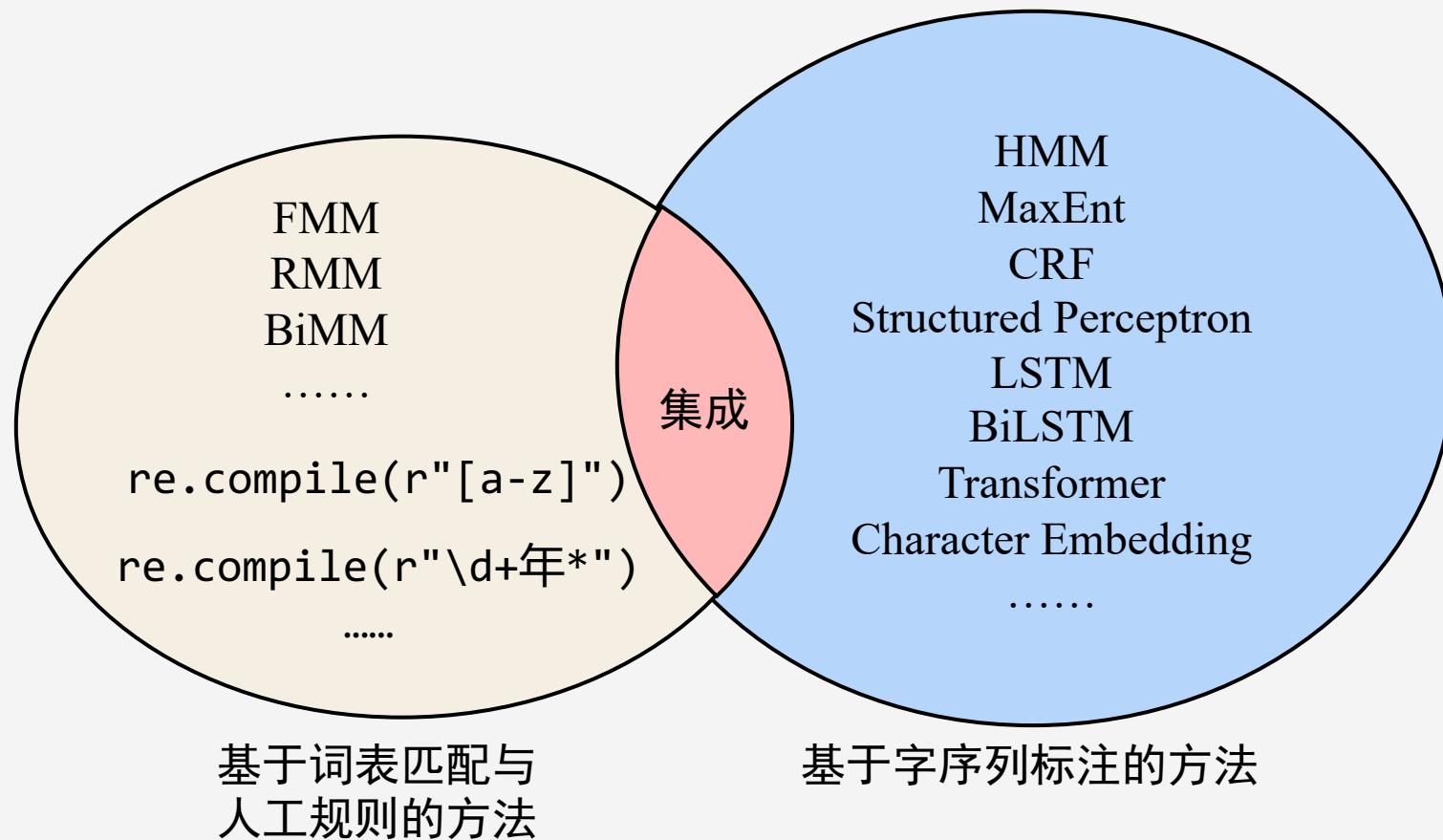
共同 创造 美好 的 新 世 纪 —— 二〇〇一 年 新 年 贺 词

- 评测指标：Precision, Recall, F1, IV Recall, OOV Recall

```
$ perl scripts/score training_vocab.txt test.txt predict.txt
```

研究思路

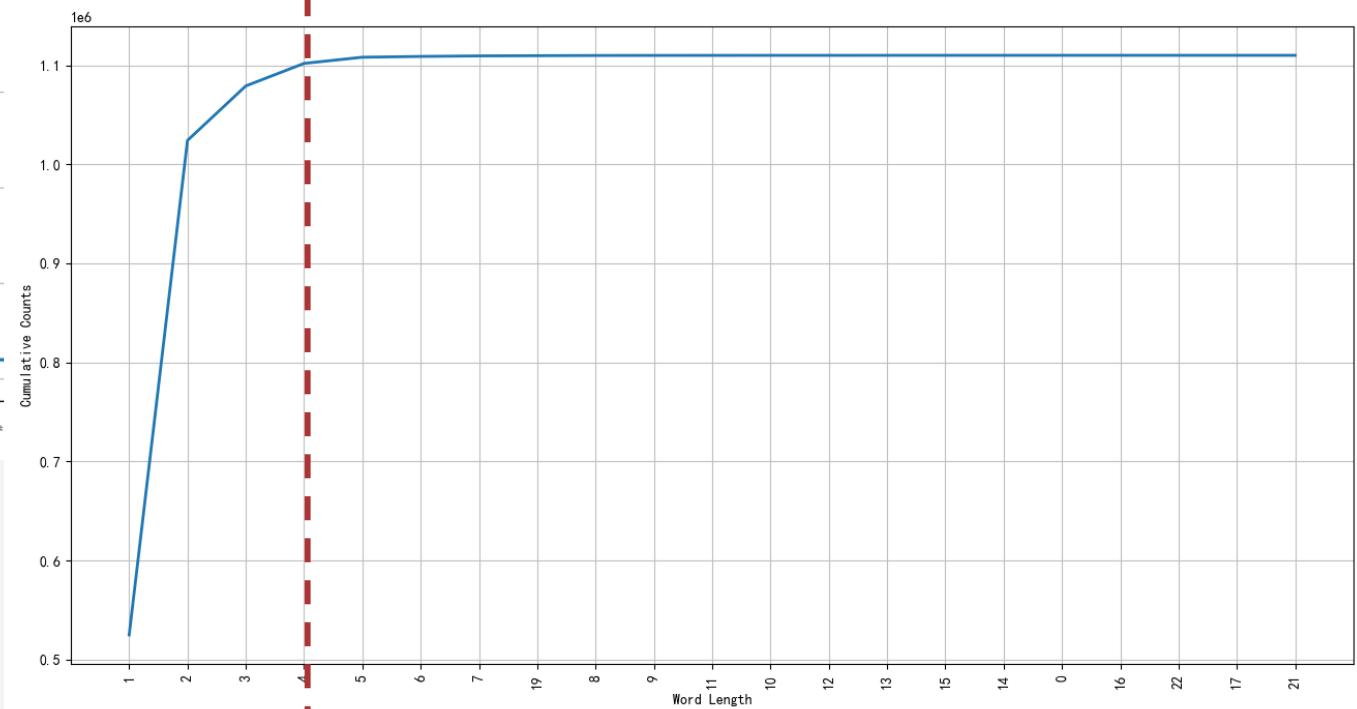
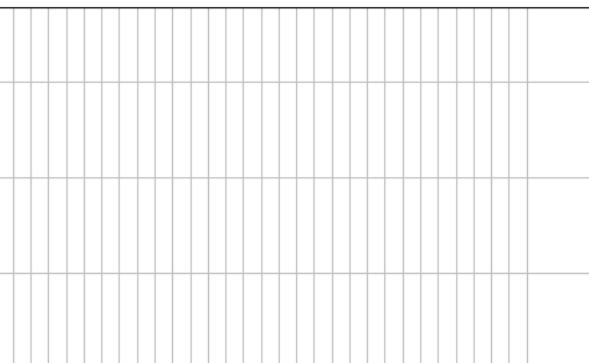
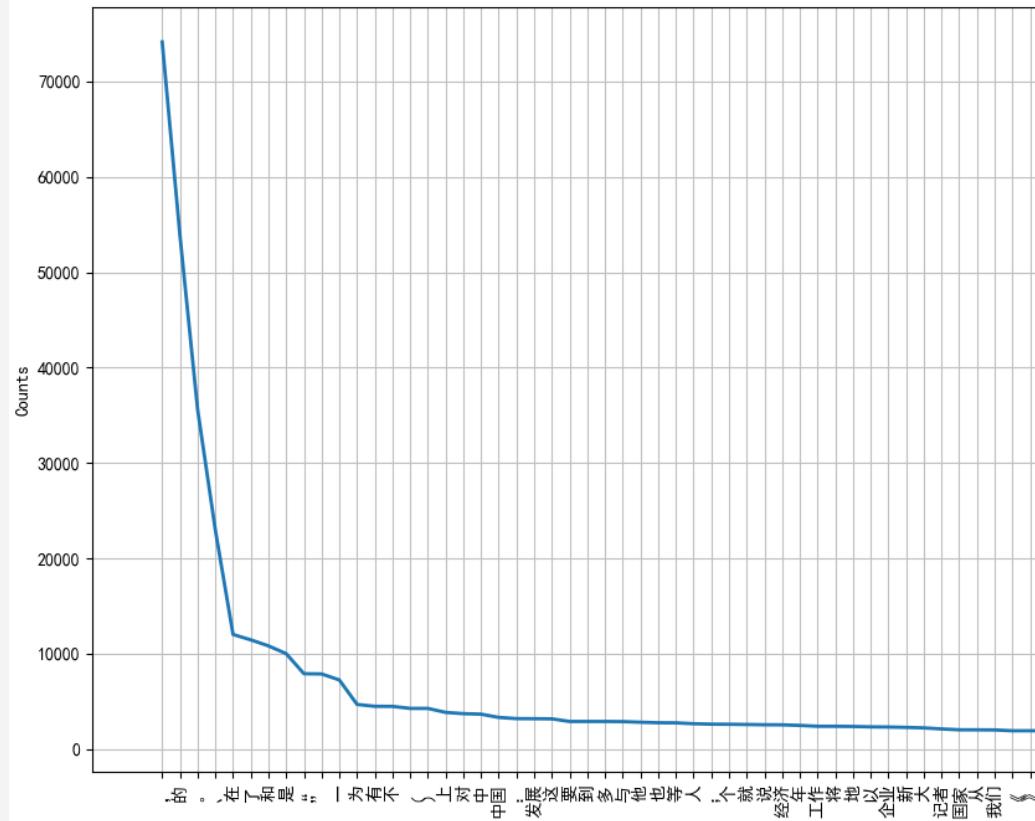
- 基于词表匹配与规则的方法，直接切分（最大匹配，正则表达式……）
- 字序列标注法，然后采用HMM、MaxEnt、CRF、语言模型……



迈向充满希望的新世纪
B E B E B E S S B E
.....

4 tag 标注示例

探索性分析



最大匹配算法



- FMM，扫描句子，切词，查找词表，未匹配时移除右侧字符
- RMM，扫描句子，切词，查找词表，未匹配时移除左侧字符

max len=4

共同创造美好的新世纪——二〇〇一年新年贺词

共同创造美好

```
def max_forward(Line: str, max_len = 4):  
    res = []  
    n = len(Line)  
    idx = 0  
    while idx < n:  
        lens = max_len  
        while lens > 0:  
            sub = Line[idx: idx + lens]  
            print(sub)  
            if sub in vocab:  
                res.append(sub)  
                idx += lens  
                lens = max_len  
            else:  
                lens -= 1  
            if lens == 0:  
                idx += 1  
    return res
```

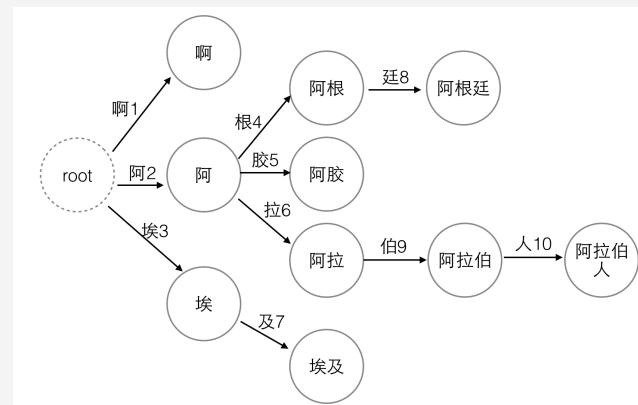
最大匹配算法

基于词典的最大匹配算法的特点：

- 实现简单
- 速度快，效率高
- 效果很大取决于字典的质量
- 新词发现能力几乎为0

词表的实现可以是：

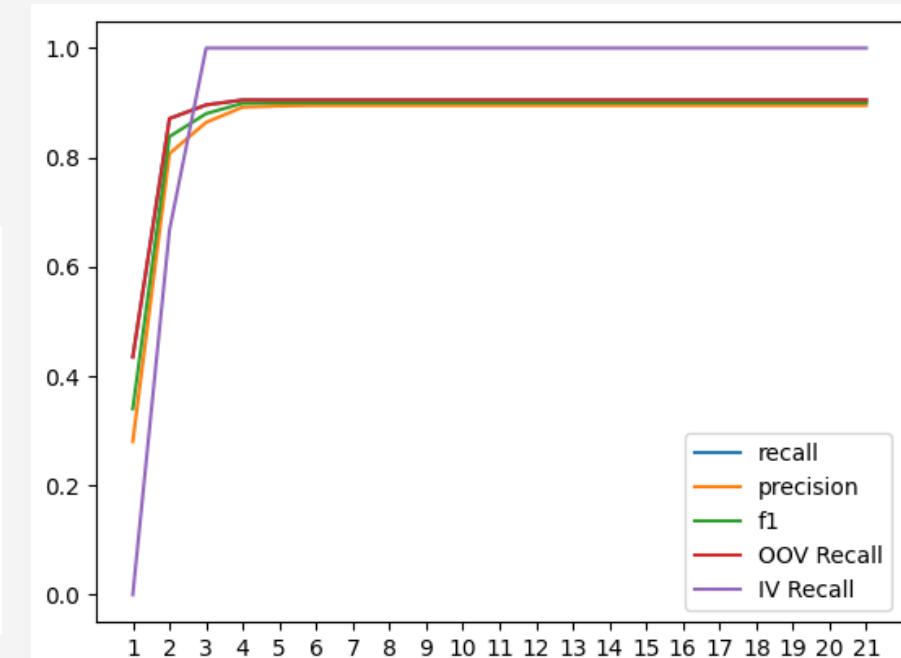
- Set $O(n \log N)$
- Hash $O(n)$
- Trie $O(n)$
- DAT (Double Array Trie)
- 前缀字典 (jieba)



Trie树

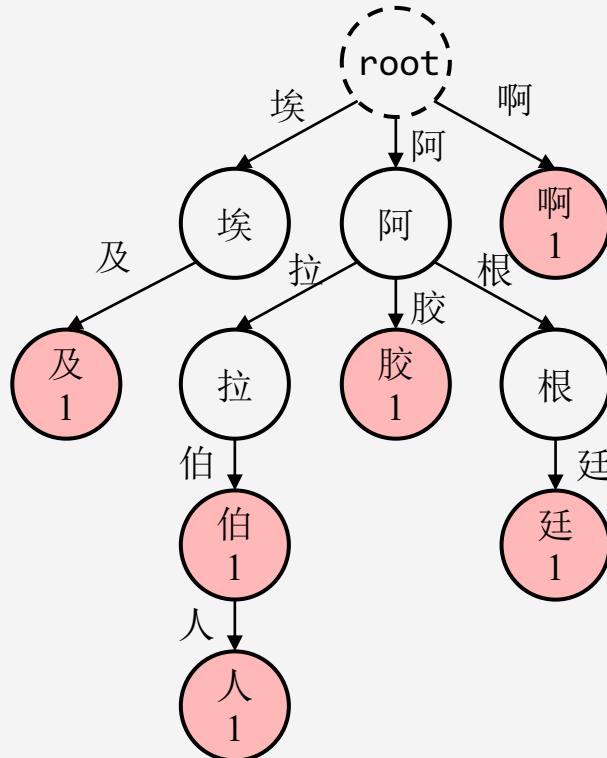
RECALL: 0.905
PRECISION: 0.892
F1 : 0.899

OOV Rate: 0.058
OOV Recall: 0.000
IV Recall: 0.960



最大长度的影响

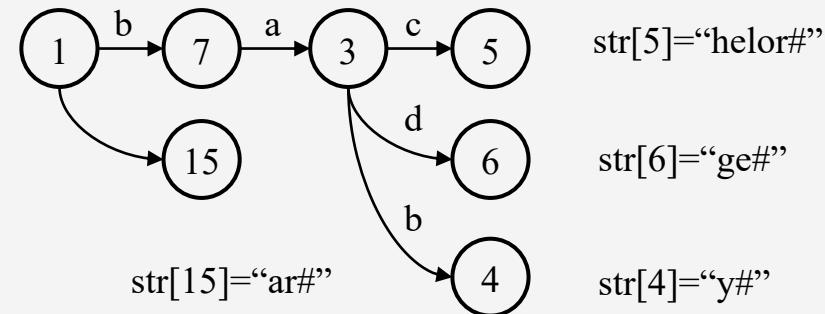
Input: 埃及, 阿根廷, 阿拉伯, 阿胶, 啊, 阿拉伯人



Trie的特点：

- 查找效率高
- 支持前缀计数
- 内存开销比较大

Double Array Trie



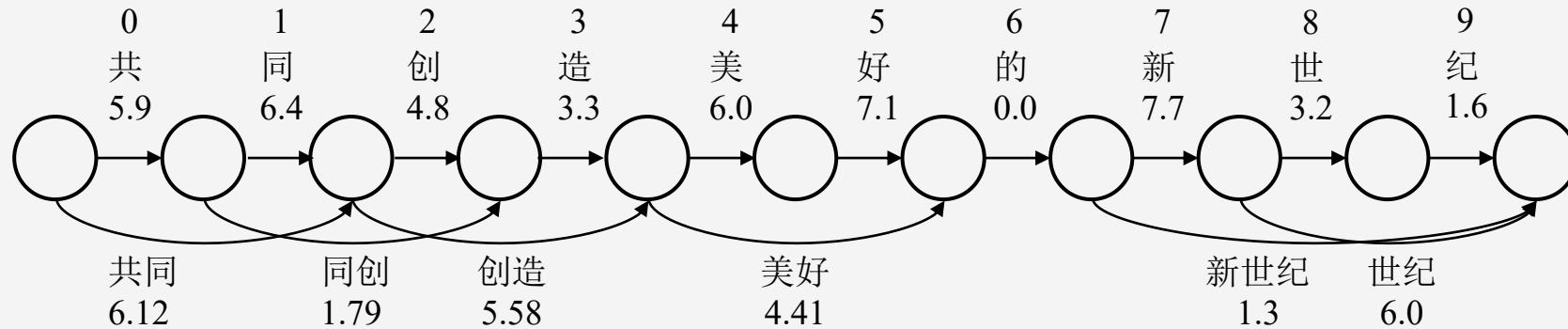
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BASE	4	0	1	-15	-1	-12	1	0	0	0	0	0	0	-9
CHECK	0	0	7	3	3	3	1	0	0	0	0	0	0	1
TAIL	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	h	e	1	o	r	#	?	?	a	r	#	g	e	#
													y	#

Trie树示例

[1] 力扣208 实现 Trie <https://leetcode-cn.com/problems/implement-trie-prefix-tree/>

[2] Aoe, J. I., Morimoto, K., & Sato, T. (1992). An efficient implementation of trie structures. Software: Practice and Experience, 22(9), 695-721.

Input: 共同创造美好的新世纪



0: [0, 1]	共, 共同
1: [1, 2]	同, 同创
2: [2, 3]	创, 创造
3: [3]	造
4: [4, 5]	美, 美好
5: [5]	好
6: [6]	的
7: [7, 9]	新, 新世纪
8: [8, 9]	世, 世纪
9: [9]	纪



动态规划
逆向计算最短路径

RECALL: 0.924
PRECISION: 0.881
F1 : 0.902

OOV Rate: 0.058
OOV Recall: 0.193
IV Recall: 0.968

词表匹配法效果对比



序号	匹配方法	maxlen	P	R	F1	Roov	RIV
1	FMM	4	0.890	0.903	0.896	0.000	0.958
2	FMM	5	0.892	0.903	0.897	0.000	0.958
3	RMM	4	0.892	0.905	0.899	0.000	0.960
4	RMM	5	0.894	0.905	0.900	0.000	0.960
5	DAG	-	0.881	0.924	0.902	0.193	0.968

- 基于词图的匹配方法，可以视为对最大匹配的改进，在其上考虑了词频与概率信息
- 使用动态规划算法得到概率最大的路径序列，兼具效果与速度
- DAG图上的词频概率的计算可以是多样的

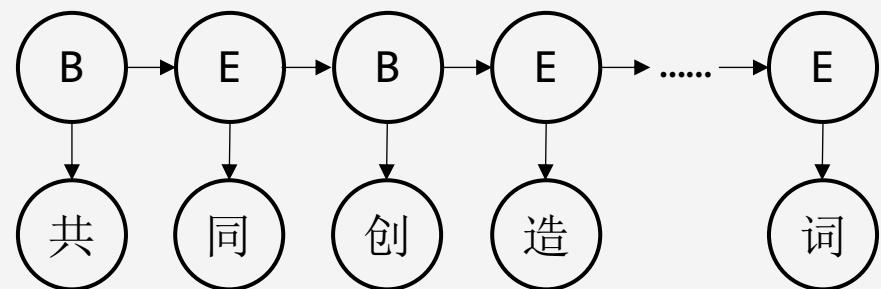
基于隐马尔可夫模型的中文分词：

- 观测序列 V 为输入句子，隐状态 Q 为BMSE标注信息
 - HMM的监督训练即为词（字）频统计，一阶HMM等价于Bigram Model
 - 通过Viterbi解码得到最可能的状态转移序列

$$\delta_t(i) = \max_{1 \leq j \leq N} [\delta_{t-1}(j)a_{ji}]b_i(o_t), i = 1, 2, \dots, N$$

$$\Psi_t(i) = \arg \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}], i = 1, 2, \dots, N$$

Viterbi解码递推公式



RECALL: 0.745
PRECISION: 0.715
F1 : 0.730

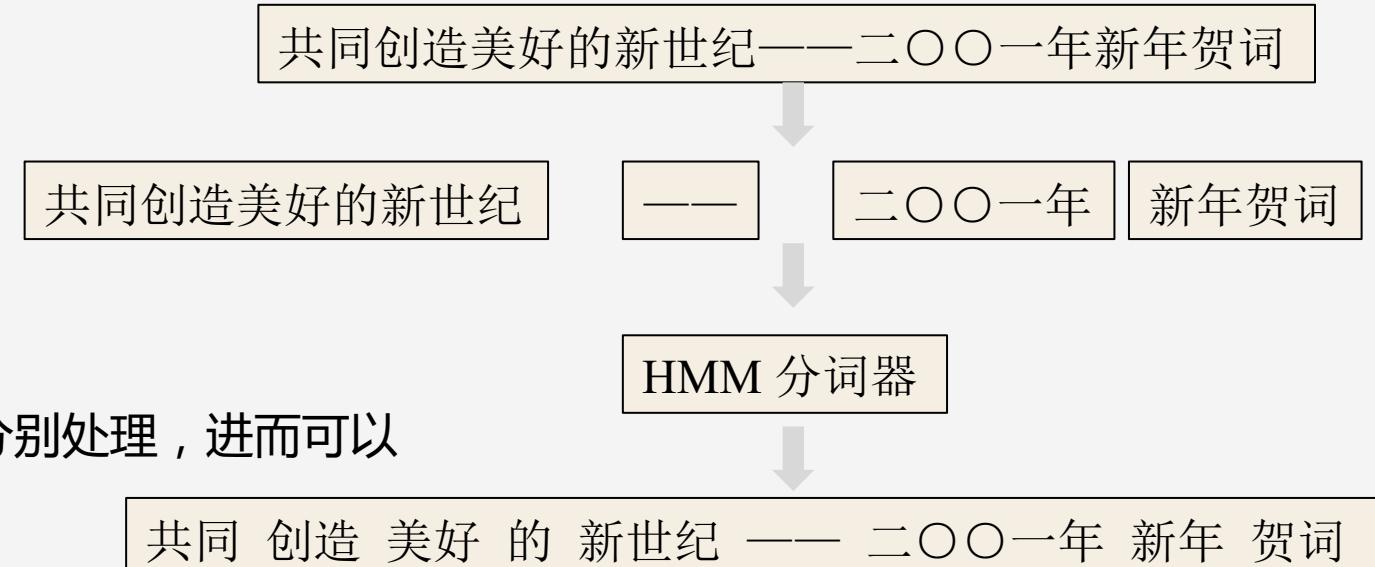
OOV Rate: 0.058
OOV Recall: 0.315
IV Recall: 0.771

HMM示例

HMM分词可以做的改进

- 一般不使用numpy实现，因为emition的维度难以确定
 - 对概率取log运算后存储，将乘法变成加法，避免精度损失
 - 对状态转移进行限定，只考虑合法状态

```
PrevStatus = {  
    'B': 'ES',  
    'M': 'MB',  
    'S': 'SE',  
    'E': 'BM'  
}
```



- 可以将一个句子分为多个chunk（段），分别处理，进而可以引入人工规则
 - 优先使用词表匹配，仅将HMM用于未登录词识别

HMM				P	R	F1	Roov	RIV
序号	人工规则	chunk	词表匹配					
1	否	否	否	0.715	0.745	0.730	0.315	0.771
2	否	是	否	0.760	0.784	0.772	0.436	0.805
3	是	否	否	0.716	0.746	0.731	0.318	0.772
4	是	是	否	0.716	0.746	0.731	0.318	0.772
5	否	是	是	0.887	0.866	0.877	0.434	0.893

- 词表匹配法F1值最高，使用HMM后有所降低，但是OOV Recall有显著提升
- 仅用HMM效果比较差，在IV词识别上很低
- chunk处理也能提升效果，但是注意人工规则不要太多

TNT - A Statistical Part-of-Speech Tagger



TNT			P	R	F1	Roov	RIV
序号	模型	chunk					
1	HMM	否	0.715	0.745	0.730	0.315	0.771
2	HMM	是	0.760	0.784	0.772	0.436	0.805
3	TNT	否	0.767	0.767	0.767	0.341	0.793
4	TNT	是	0.776	0.785	0.780	0.417	0.807
5	snownlp[1]	是	0.893	0.897	0.895	0.325	0.932

- TNT基于2阶HMM，提出Trigrams'n Tags标注方案，对条件概率进行平滑处理

$$p(t_3|t_2, t_1) = \lambda_1 \hat{P}(t_3) + \lambda_1 \hat{P}(t_3|t_2) + \lambda_1 \hat{P}(t_3|t_2, t_1)$$

[1] SNOWNLP <https://github.com/isnowfy/snownlp>

[2] TNT — A Statistical Part-of-Speech Tagger <https://arxiv.org/pdf/cs/0003055.pdf>

最大熵原理

- 在学习概率模型时，所有可能的概率模型中，**熵最大的模型是最好的模型**
- 最大熵原理是统计学习的一般性原理**，最大熵模型是该原理在分类问题上的应用
- 最大熵模型采用改进的迭代尺度法或拟牛顿法、梯度下降法等进行优化

$$\max_{P \in \mathcal{C}} H(P) = - \sum_{x,y} = \tilde{P}(x)P(y|x) \log P(y|x)$$

$$s.t. \quad E_P(f_i) = E_{\tilde{P}}(f_i), i = 1, 2, \dots, n$$

$$\sum_y P(y|x) = 1$$

$$f(x, y) = \begin{cases} 1, & x \text{ co-occur with } y \\ 0, & \text{otherwise} \end{cases}$$

特征函数

$$E_{\tilde{P}}(f) = \sum_{x,y} \tilde{P}(x,y)f(x,y)$$

关于经验分布的期望

$$E_P(f) = \sum_{x,y} \tilde{P}(x)P(y|x)f(x,y)$$

关于模型分布的期望

$$\mathcal{C} \equiv \{P \in \mathcal{P} | E_P(f_i) = E_{\tilde{P}}(f_i), \quad i = 1, 2, \dots, n\}$$

$$H(P) = - \sum_{x,y} \tilde{P}(x)P(y|x) \log P(y|x)$$

条件熵

最大熵算法 Max Entropy

$$P_w(y|x) = \frac{1}{Z_w(x)} \exp\left(\sum_i w_i f_i(x, y)\right)$$

$$Z_w(x) = \sum_y \exp\left(\sum_i w_i f_i(x, y)\right)$$

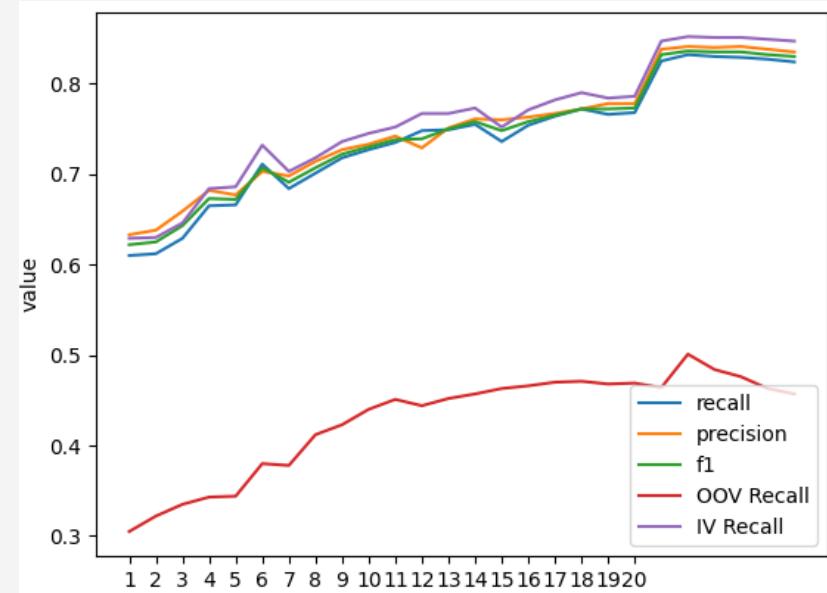
w 为最大熵模型的参数

$f_i(x, y)$ 为特征函数

$Z_w(x)$ 为归一化因子

基于张乐博士的maxent工具包实现分词

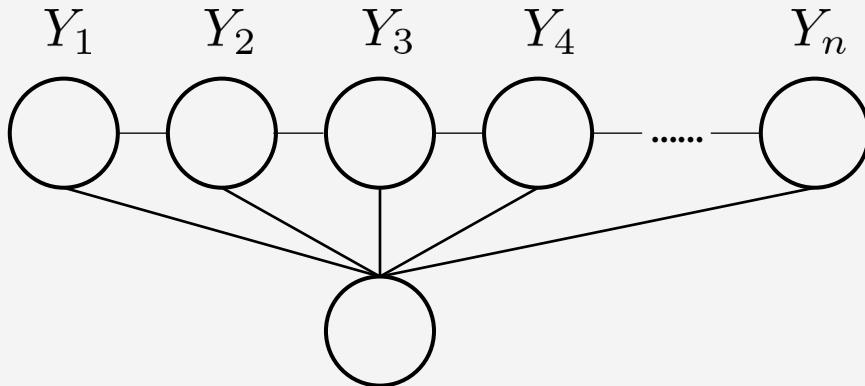
- maxent : <https://github.com/lzhang10/maxent>



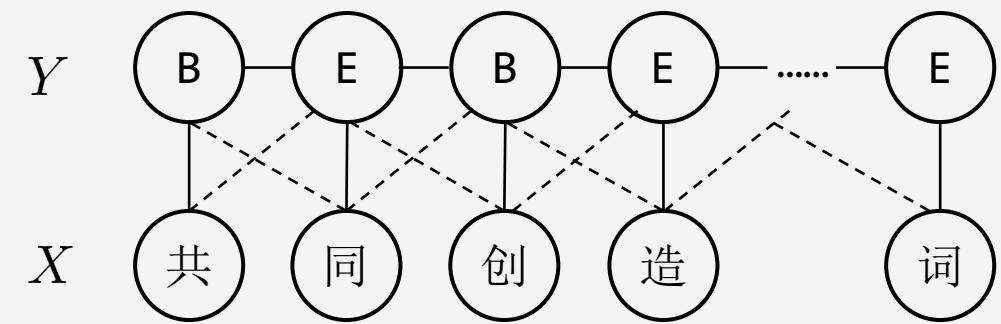
RECALL: 0.832
PRECISION: 0.841
F1 : 0.836

OOV Rate: 0.058
OOV Recall: 0.501
IV Recall: 0.852

- CRF是一种无向图模型，常用于解决序列标注问题
- 与HMM不同，CRF是一种判别模型，建模 $P(Y|X)$
- 摒弃了HMM中两个不合理的假设，齐次马尔可夫假设和条件独立假设



$$X = X_1, \dots, X_{n-1}, X_n$$



linear CRF示例

满足马尔可夫性 $P(Y_i|X, Y_1, Y_2, \dots, Y_n) = P(Y_i|X, Y_{i-1}, Y_{i+1})$

参数化表示

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i,k} \lambda_k t_k(y_{i-1}, y_i, x, i) + \sum_{i,l} \mu_l s_l(y_i, x, i)\right)$$

$$Z(x) = \sum_y \exp\left(\sum_{i,k} \lambda_k t_k(y_{i-1}, y_i, x, i) + \sum_{i,l} \mu_l s_l(y_i, x, i)\right)$$

学习

$$L(w) = \log \prod_{x,y} P_w(y|x)^{\bar{P}(x,y)} = \sum_{x,y} = \bar{P}(x,y) \log P_w(y|x)$$

Viterbi解码

$$\delta_{i+1}(l) = \max_{1 \leq j \leq m} \left\{ \delta_t(j) + \sum_{k=1}^K w_k f_k(y_i = j, y_{i+1} = l, x, i) \right\}, l = 1, 2, \dots, m$$

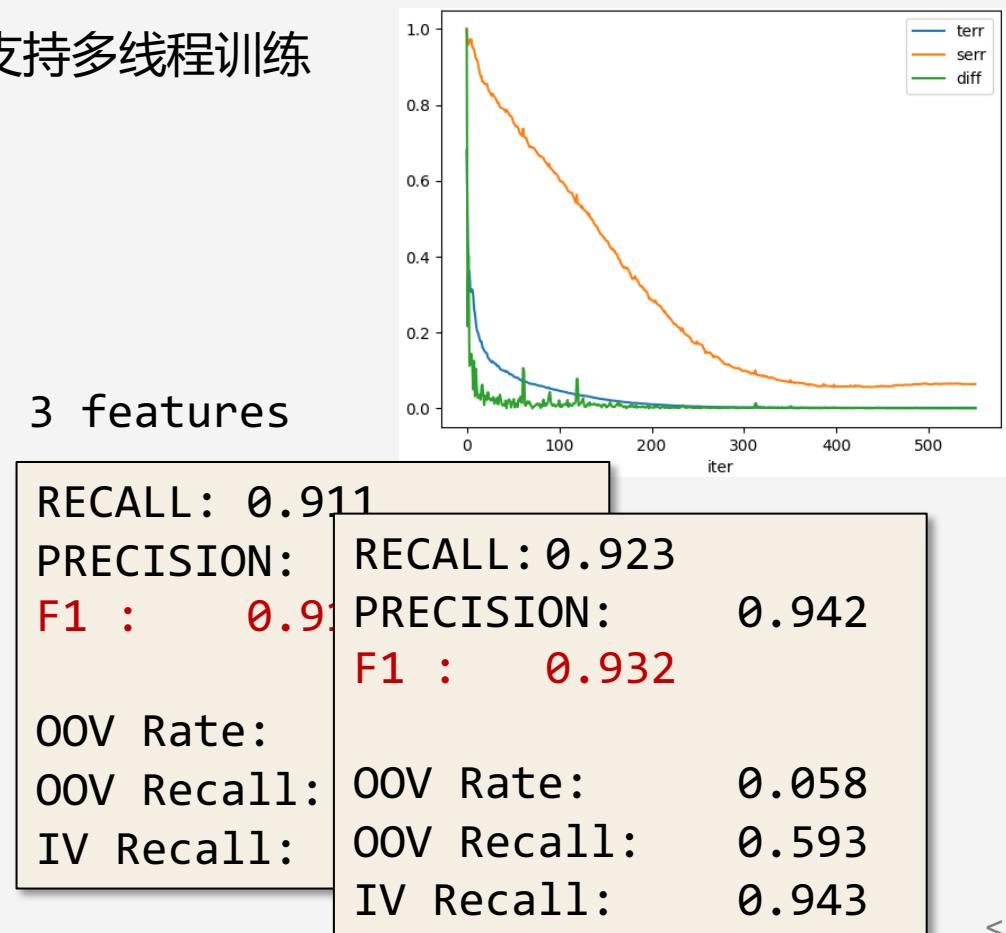
$$\Psi_{i+1}(l) \arg \max_{1 \leq j \leq m} \left\{ \delta_i(l) \right\} + \sum_{k=1}^K w_k f_k(y_i = j, y_{i+1} = l, x, i), l = 1, 2, \dots, m$$

基于CRF++实现中文分词

- crf++ : <https://github.com/lzhang10/maxent>
- CRF++中使用template文件来指定特征，并自动生成出满足要求的特征函数
- 根据用户指定特征函数进行训练CRF模型，C++编写，支持多线程训练

```
# Unigram
U00:%x[-2,0]      当前词的左侧两个词
U01:%x[-1,0]      当前词的左侧一个词
U02:%x[0,0]        当前词
U03:%x[1,0]        当前词的右侧一个词
U04:%x[2,0]
U05:%x[-2,0]/%x[-1,0]
U06:%x[-1,0]/%x[0,0]      BiGram
U07:%x[0,0]/%x[1,0]
U08:%x[1,0]/%x[2,0]
U09:%x[-2,0]/%x[-1,0]/%x[0,0]
U10:%x[-1,0]/%x[0,0]/%x[1,0]      TiGram
U11:%x[0,0]/%x[1,0]/%x[2,0]
```

思想自由 兼容并包 template文件示例



PKUSEG: A Toolkit for Multi-Domain Chinese Word Segmentation

- pkuseg : <https://github.com/lancopku/pkuseg-python>
- 使用ADF（基于频率信息的自适应在线梯度更新）算法对CRF模型进行改进
- 出现频次越高的特征，应该越重要
- 在CRF的训练中，不同的参数使用不同的学习率进行训练，学习率可根据参数频率在线调整
- F1=88.0% (训练过程中F1达到了95%)

结构化感知机 Structured Perceptron

- CRF全局化地以最大熵准则建模概率 $P(Y|X)$ ，SP则是以最大熵准则建模一个score评分函数

$$\text{score函数} \quad S(Y, X) = \sum \alpha_s \Phi_s(Y, X)$$

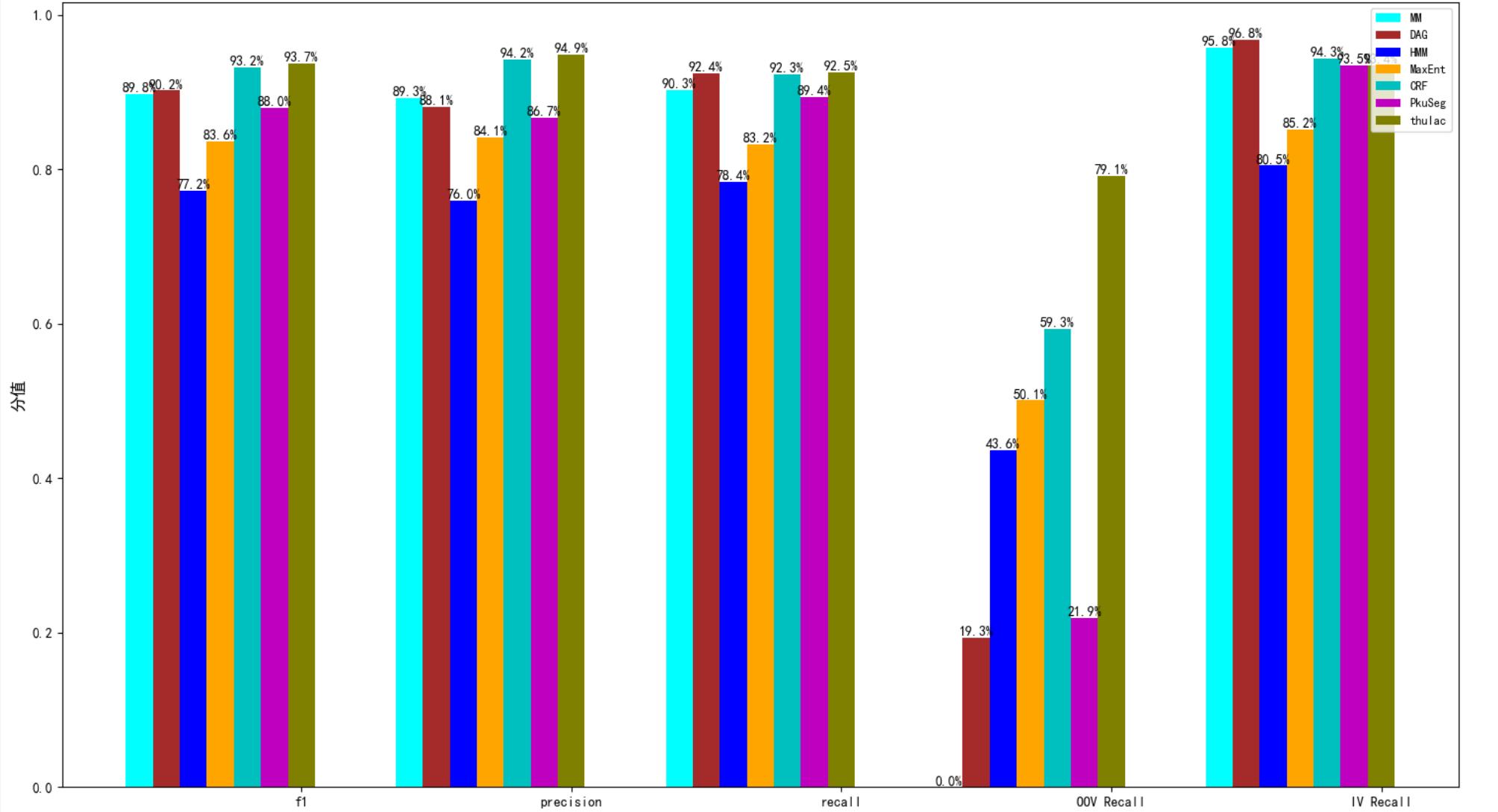
$$\text{本地函数的全局化表示} \quad \Phi_s(Y, X) = \sum_i^s \phi_s(h_i, y_i)$$

求解令score函数最大的Y序列 $\arg \max_Y S(Y, X)$

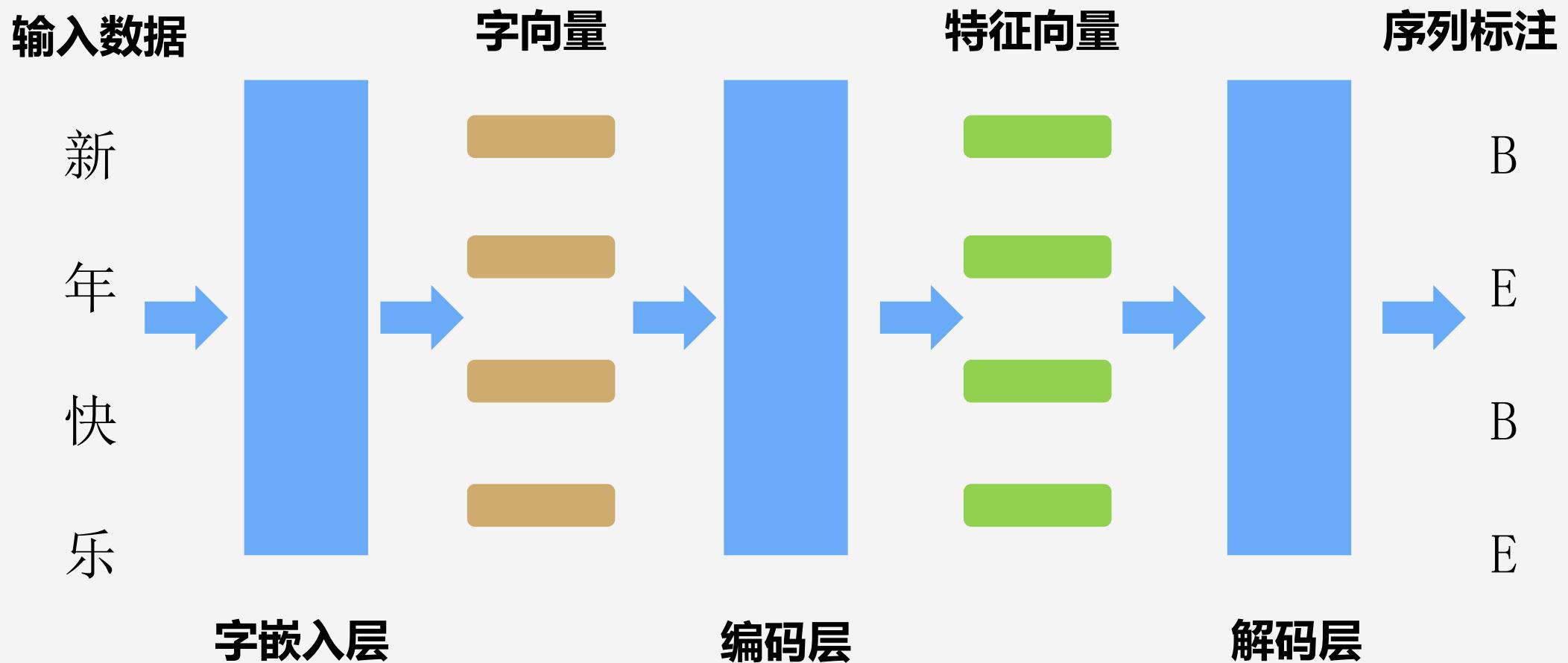
- 基于thulac : <http://thulac.thunlp.org/> 实现结构化感知机算法，F1=93.7%

方法对比

不同分词算法的对比



- 基于深度学习的分词系统的一种基本框架^[1]



• 数据预处理

- 普通字符处理：直接拆分

总理走上台 → 总理走上台

- 标点符号处理

- 单独的符号（如“。”，“”（“ 等）：使用**特殊符号 “[PUNC]”**统一代替

- 连续的符号（如“……”，“——”）：**合并后**使用**特殊符号 “[PUNC]”**统一代替

总理走上台来，亲切慰问我们…… → 总理走上台来 [PUNC] 亲切慰问我们 [PUNC]

- 数字处理

- 三类数字：全角、半角、中文

```
half_numbers = "1234567890"  
full_numbers = "一 二 三 四 五 六 七 八 九 。〇 十 百 万 亿"
```

- 分类合并后使用**特殊符号 “[NUM]”**统一代替

二〇〇〇年十二月三十一日 → [NUM] 年 [NUM] 月 [NUM] 日

- 字母处理：合并后使用**特殊符号 “[ALP]”**统一代替

• 字嵌入层

- GloVe: Global Vector for Word Representation*

- 输入为语料中字的共现频率矩阵
- 损失函数

$$J = \sum_{i,j=1}^V f(X_{ij}) (w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$
$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases}$$

- 参数设置

```
Vocab min count: 3, vector size: 300, iteration number: 1000, window size: 15
```

• 编码层

- LSTM *

Forget Gate: $f^{(t)} = \sigma(U^f x^{(t)} + W^f h^{(t-1)} + b^f)$

Input Gate: $i^{(t)} = \sigma(U^i x^{(t)} + W^i h^{(t-1)} + b^i)$

Output Gate: $o^{(t)} = \sigma(U^o x^{(t)} + W^o h^{(t-1)} + b^o)$

内部状态更新

$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$$

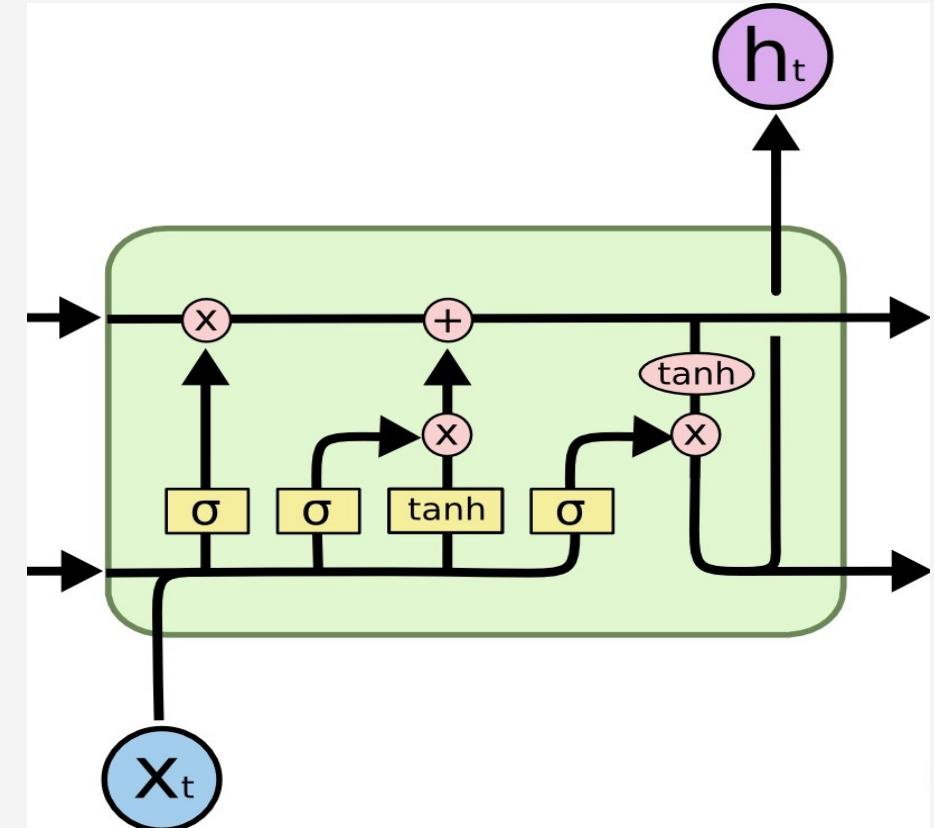
$$\tilde{c}^{(t)} = \tanh(Ux^{(t)} + Wh^{(t-1)} + b^f)$$

输出

$$h^{(t)} = o^{(t)} \circ \tanh(c^{(t)})$$

- 参数设置

Layers: 4, Hidden size: 300, Bidirectional: True, Dropout: 0.1



• 编码层

- Transformer *

Multi-Head Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

Add & Norm:

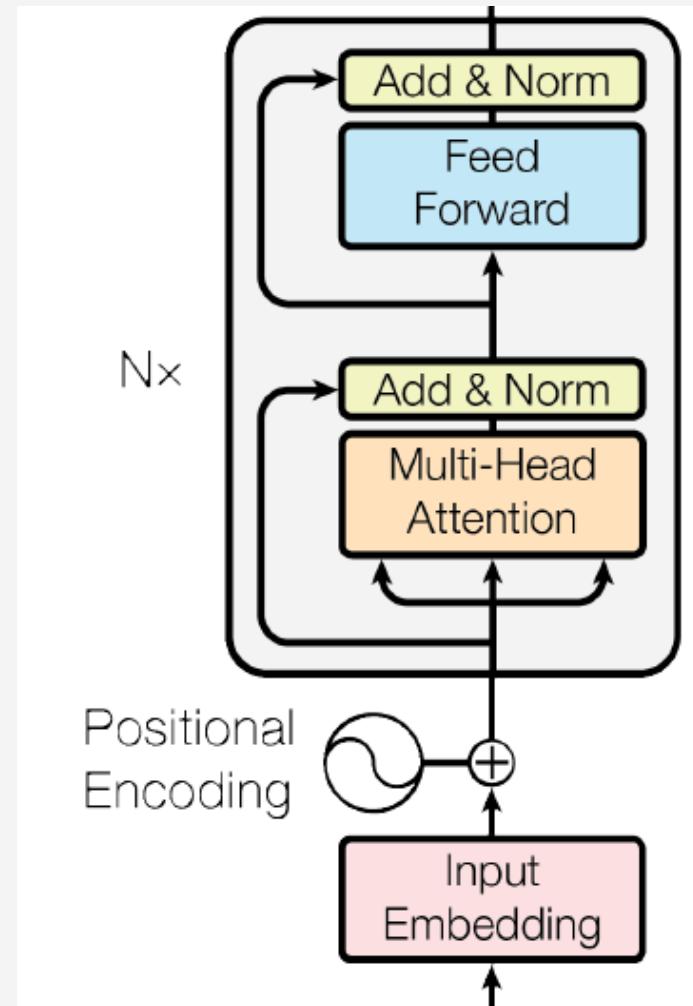
$$\tilde{x} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

Feed Forward:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- 参数设置

Layers: 4, Hidden size: 300, Heads number: 2, Dropout: 0.1



• 解码层

- Linear Layer + Softmax
 - 将编码层得到的特征输入全连接层
 - 经softmax后每个字对应输出**5维向量**，表示属于某类（BMESO）的概率
 - 损失函数：交叉熵
- CRF
 - 将编码层得到的特征输入CRF层，得到最大得分和相应序列
 - 损失函数：模型当前**最大得分**与**正确标注序列得分**之差

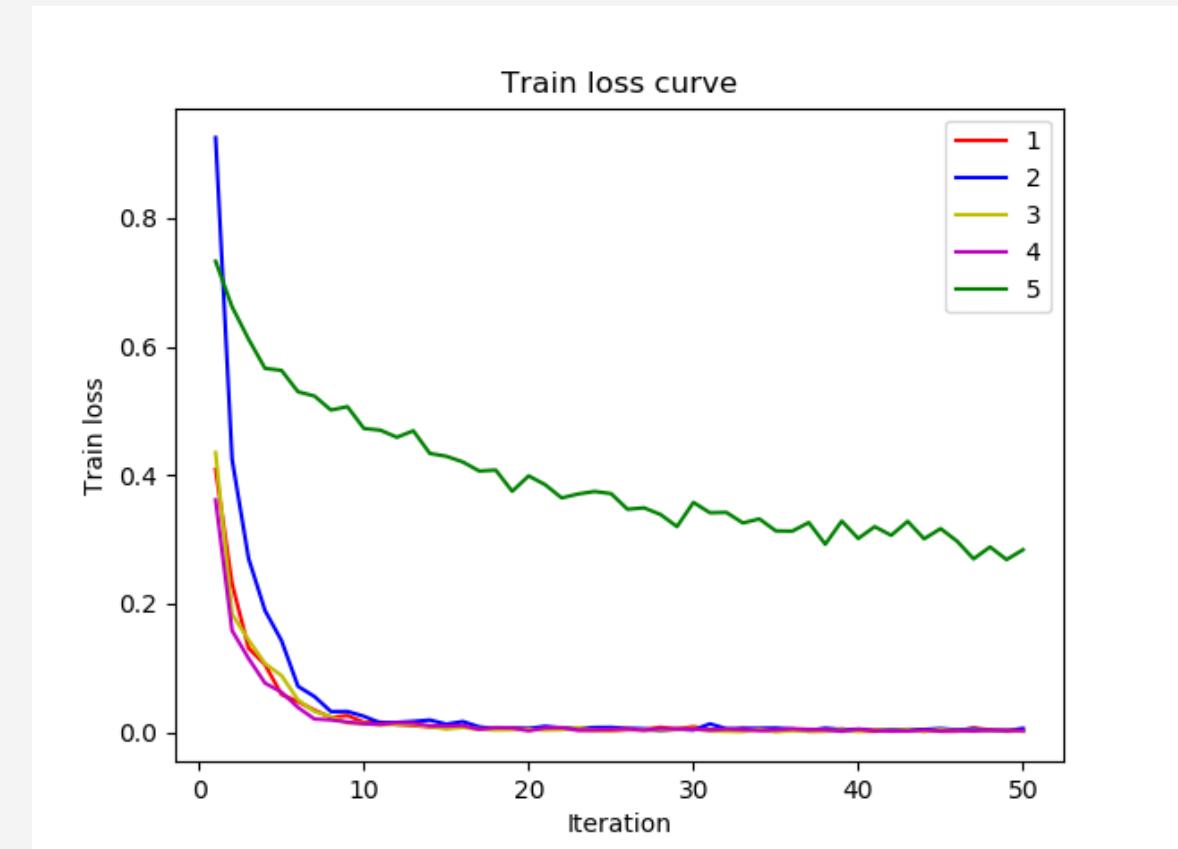
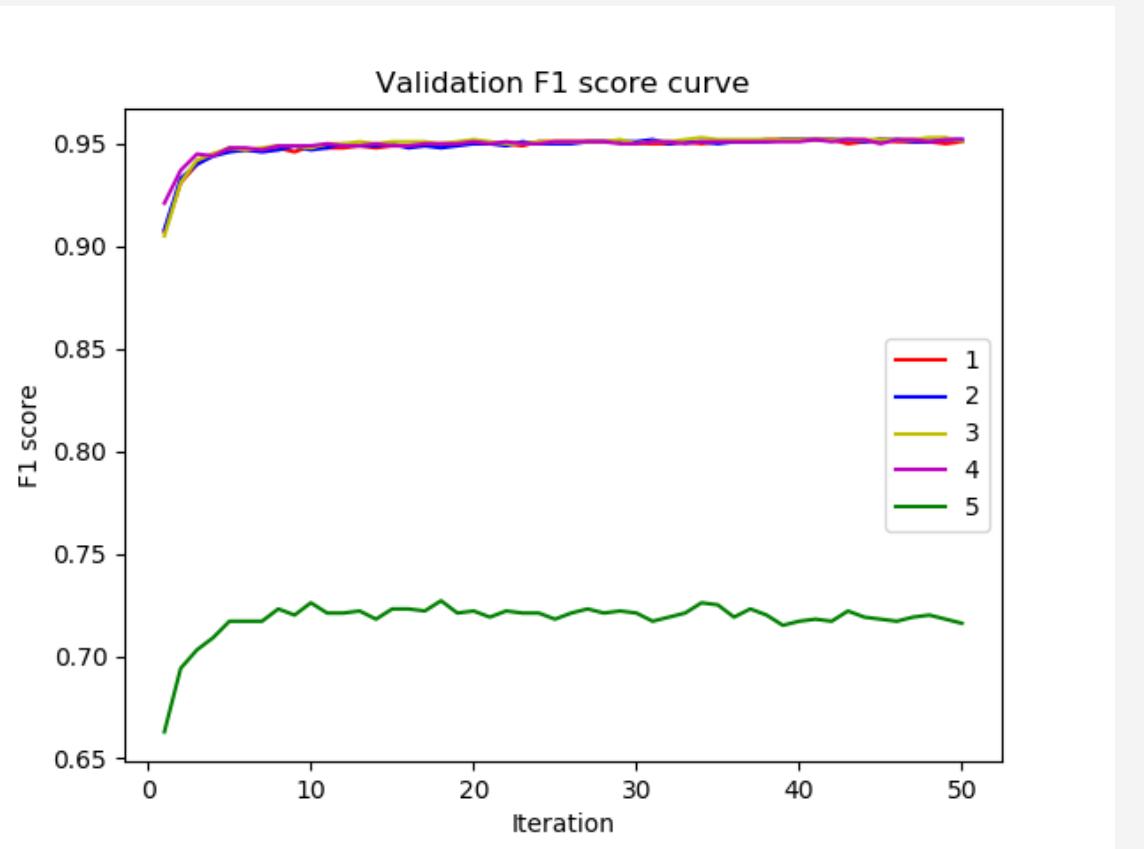
- 交叉验证结果对比 [1]

模型					P	R	F1
序号	预处理	字嵌入层	编码层	解码层			
1	否	Glove	LSTM	Linear [2]	0.952	0.952	0.952
2	否	Glove	LSTM	CRF	0.952	0.952	0.952
3	是	Glove	LSTM	Linear	0.953	0.952	0.953
4	是	Random	LSTM	Linear	0.951	0.952	0.952
5	是	Glove	Transformer	Linear	0.727	0.734	0.731

[1] : 五折交叉验证；最多训练50个epoch，直到模型收敛；优化器：Adam ($lr = 1e-4$)

[2] : Linear Layer + Softmax

• 交叉验证结果对比



- 测试集结果对比 [1]

模型					P	R	F1	Roov	RIV
序号	预处理	字嵌入层	编码层	解码层					
1	否	Glove	LSTM	Linear [2]	0.927	0.921	0.924	0.597	0.940
2	否	Glove	LSTM	CRF	0.928	0.923	0.926	0.629	0.941
3	是	Glove	LSTM	Linear	0.942	0.937	0.940	0.775	0.947
4	是	Random	LSTM	Linear	0.940	0.937	0.938	0.765	0.947
5 [3]	是	Glove	Trans[4]	Linear	0.641	0.646	0.644	0.409	0.660

[1] : 训练20个epoch

[2] : Linear Layer + Softmax

[3] : 分词任务可能不太需要Transformer模型所提供的全局信息

模型5可能的改进：切割样本，缩短长度；精调lr、hidden_size等参数；增大训练数据量...

[4] : Transformer

• 测试样例对比

共同 创造 美好 的 新 世 纪 —— 二〇〇一 年 新 年 贺 词

• 1 : 共同 创造 美好 的 新 世 纪 —— 二〇〇一 年 新 年 贺 词

• 2 : 共同 创造 美好 的 新 世 纪 —— 二〇〇一 年 新 年 贺 词

• 3 : 共同 创造 美好 的 新 世 纪 —— 二〇〇一 年 新 年 贺 词

• 4 : 共同 创造 美好 的 新 世 纪 —— 二〇〇一 年 新 年 贺 词

• 5 : 共 同 创造 美好 的 新 世 纪 —— 二〇〇一 年 新 年 贺 词

• 分析

- 数据预处理对特殊字符分词具有重要影响

模型				
序号	预处理	字嵌入层	编码层	解码层
1	否	Glove	LSTM	Linear
2	否	Glove	LSTM	CRF
3	是	Glove	LSTM	Linear
4	是	Random	LSTM	Linear
5	是	Glove	Trans	Linear

• 测试样例对比 中国 人民 进入 新 世 纪 的 主 要 任 务 , 就 是 继 续 推 进 现 代 化 (未完)

• 1: 中国 人民 进入 新 世 纪 的 主 要 任 务 , 就 是 继 续 推 进 现 代 化

• 2: 中国 人民 进入 新 世 纪 的 主 要 任 务 , 就 是 继 续 推 进 现 代 化

• 3: 中国 人民 进入 新 世 纪 的 主 要 任 务 , **就是** 继 续 推 进 现 代 化

• 4: 中国 人民 进入 新 世 纪 的 主 要 任 务 , 就 是 继 续 推 进 现 代 化

• 5: **中国人民进入** 新 世 纪 **的主 要 任 务** , 就 是 继 续 **推 进 现 代 化**

• 分析

- 对于长文本（测试集中最长含1019个字符），由于Transformer对整段文本进行self-attention，特征中含有**全局信息**过多，导致可能更需要**局部信息**的分词任务效果不好

模型				
序号	预处理	字嵌入层	编码层	解码层
1	否	Glove	LSTM	Linear
2	否	Glove	LSTM	CRF
3	是	Glove	LSTM	Linear
4	是	Random	LSTM	Linear
5	是	Glove	Trans	Linear

可以做的改进

- 对参数设置和模型组合进行更多的尝试，给出更详细的对比分析
- Transformer在本次分词任务的尝试上并没有成功，可以考虑改进训练方法，或者采用适合长文本处理的Transformer-XL
- 对于传统的分词方法，存在很多工程上可改进的空间，进一步实践

收获与成长

- 对各种较为知名的分词模型与算法（传统的或深度的）都做了一定的尝试，对分词问题也有了一个较为全面的认识
- 对比了深度学习和传统机器学习在分词问题上的特点，适用场景
- 通过实验，也验证了一些结论，并且发现了一些实践上有效的技巧



感谢您的关注

<https://github.com/RonDen/HanTokenization>