面向移进-归约句法分析器的单模型系统融合算法*

马 骥，朱慕华，肖 桐，朱靖波
东北大学 自然语言处理实验室，辽宁 沈阳 110004
E-mail: {majineu, zhumuhua}@gmail.com; {xiaotong, zhujingbo}@mail.neu.edu.cn

摘 要：本文提出了一种面向移进-归约句法分析器的单模型系统融合算法。在训练阶段，该方法通过调整训练数据的分布，来构建用于融合的多个移进-归约句法分析器。在解码阶段，该方法首先使用各个移进-归约句法分析器对待分析的句子进行句法分析，然后利用一个线性模型对各句法分析器输出的句法树进行评分，从中选出得分最高的句法树作为最终结果。本文中的实验是在宾州英文树库上进行的。实验结果表明，本文中的方法能够显著改善基准系统的性能。

关键词：句法分析；系统融合；移进-归约句法分析器

Single-Model System Combination for Shift-Reduce Parser

Ma Ji, Zhu Muhua, Xiao Tong, Zhu Jingbo
Nature Language Processing Lab, Northeastern University, Shenyang 110004
E-mail: {majineu, zhumuhua}@gmail.com; {xiaotong, zhujingbo}@mail.neu.edu.cn

Abstract: This paper proposes a single-model system combination method for a shift-reduce parser. In the training step, a group of shift-reduce parsers are automatically constructed by varying the distribution of the training data. In the decoding step, our method first uses the parsers to parse the input sentence. Then it uses a linear model function to select the parse tree with the highest score as the final result. We conduct our experiment on the Engligh Penn Treebank. Experimental results show that our method leads to significant improvements over the baseline system.

Keywords: parsing; system combination; shift-reduce parser

1 引言

系统融合的主要目的在于通过融合多套不同的系统来得到更好的整体性能。基于系统融合的方法在近年来已经受到广泛的关注，这类方法已经广泛应用于与自然语言处理领域相关的任务中，例如文本分类[1]，机器翻译[2][3] 和句法分析 [4][5]等。一般而言，基于系统融合的方法要解决的两个核心问题为：如何构建多套用于融合的子系统[1]；如何将各子系统的输出进行融合从而得到最终结果。对于问题一，一种方法是用不同的模型来构建各个子系统。本文称这种方法为**基于多模型的系统融合**。另一种方法是用同一个模型来构建不同的子系统。本文称这种方法为**基于单模型的系统融合**。多模型系统融合的缺点在于开发不同模型需要付出较高的时间与人力。而单模型系统融合开发代价相对较低。因此，本文主要研究面向移进-归约句法分析器的单模型系统融合技术。在训练阶段，本文使用基于 AdaBoost[6]的子系统生成算法，来构建多个移进-归约句法分析器。在解码阶段，本文主要考虑两类特征——子系统置信度和扩展动作序列置信度（第 4 章将详细介绍）——并使用这两类特征的线性组合对各子系统的输出进行融合。本作者在宾夕法尼亚大学提供的英文树库（Penn English Treebank）上进行实验。实验结果表明，本文中的方法能够显著提高移进-归约句法分析器的性能。Henderson 和 Brill[7]对面向 Collins Parser[8]的单模型系统融合方法进行过研

* 基金项目：国家自然科学基金资助项目（60873091，61073140）；中央高校基本科研业务费专项资金、高等学校博士学科点专项科研基金资助（20100042110031）。

[1] 为描述方便,本文称参与整合的各系统为子系统。

究，然而关于移进-归约句法分析器，尚没有此类研究。移进-归约句法分析器的特点在于其速度快，实现简单。因此通过系统融合来提高移进-归约句法分析器的性能是一件有意义的工作。

2 移进-归约句法分析法

移进-归约句法分析法涉及的两个主要数据结构为：输入队列 Q 和分析栈 S。队列 Q 中的元素是输入句子中尚未被处理的 <词,词性>对序列。分析栈 S 中保存着输入串中已经被处理的部分所对应的句法树片段。移进-归约句法分析法自左向右地扫描待分析的句子，并在扫描过程中执行下列动作之一。移进：将 Q 的队首元素压入栈 S，并从 Q 中删除该元素;一元/二元-X-归约：生成一个新的元素，该元素的非终结符类型为 X，弹出栈顶的一个/两个元素，并将弹出的元素作为新生成的元素的子树。最后，将新生成的元素插入到栈顶。当 Q 为空时，如果 S 中只有一个元素，则将该元素作为对输入句子的分析结果返回。如果 S 中包含多于一个元素，则报告对输入句子分析失败。基于移进-归约句法分析法的首个句法分析器是由 Sagae 和 Lavie[9]实现的，其基本思想是从 S 和 Q 的状态中抽取特征向量，然后使用一个分类器根据特征向量来选择要执行的动作。因此，整个句法分析过程可以看成是使用分类器进行一系列移进-归约动作的决策过程，而训练移进-归约句法分析器则主要是训练该分类器。用于训练分类器的训练数据是一组<特征向量,动作>对，称为分类样本。Sagae 和 Lavie 在文献[9] 中实现的移进-归约句法分析器每一步只选择一个动作来执行，因此一个输入句子仅对应唯一一组动作序列，这种贪婪的策略限制了移进-归约分析法的搜索空间，并且如果某一步选择了错误的动作，则该错误将影响对后续动作的选择。为了克服该缺点，Sagae 和 Lavie [10]使用了最优优先（best-first search）搜索算法对他们前期的工作进行改进。在分析过程中，分析器每次可以选择尽可能多的动作来对当前 S 和 Q 的状态进行扩展，并将动作执行之后的结果状态存入到一个全局优先级队列中。分析器使用基于最大熵模型的分类器对每个动作进行评分，优先级队列中的每个状态的得分是从初始状态到该状态所执行的动作序列的得分的乘积。该方法扩大了分析器的搜索范围，从而提高了分析器的性能。本文所研究的系统融合方法也是基于 Sagae 和 Lavie 在文献 [10]中提出的移进-归约句法分析器，具体方法将在余下的章节中进行详细讨论，其中第三章主要介绍了子系统构建算法，第四章主要介绍子系统输出融合算法，第五章介绍实验，第六章和第七章则分别为相关工作和对本文的总结。

3 单模型系统融合的子系统构建

本章介绍基于 AdaBoost 的子系统生成算法，该算法的主要思想是通过更新训练数据的分布，来构建各子系统。在对该算法的具体细节进行描述之前，首先介绍该算法所涉及几个主要符号的含义。本文用 P_{sen} 表示训练语料中的句子的分布，P_{smp} 表示从训练语料中抽取的分类样本的分布，SR 表示移进-归约句法分析器。SR 在训练语料中的第 i 个句子上取得的 F1-score 用 $score(i)$表示。SR 在整个训练语料中取得的平均性能用 r 来表示。

子系统生成算法通过迭代的更新 P_{sen} 和 P_{smp}，来构建各个子系统，其（第 t 轮）迭代的基本过程为：算法首先根据当前分类样本的分布 P_{smp}^{t} 来训练一个最大熵分类器，从而构建一个移进-归约句法分析器 SR_t，然后使用 SR_t 对训练语料中的句子进行句法分析。根据 SR_t 在训练语料中的每个句子上取得的分数 $score_t(i)$，以及 SR_t 在整个训练语料上取得的平均性能 r_t 来更新训练语料中的句子的权重[1]，从而得到训练语料的句子的新的分布 P_{sen}^{t+1}。由于训练一个移进-归约句法分析器的主要目的是训练该分析器中用于对移进-归约动作进行决策的分类器。因此，算法根据分布 P_{sen}^{t+1}，调整从语料中抽取的分类样本权重，从而得到新的分类样本的分布 P_{smp}^{t+1}。图 1 描述了子系统构建

[1] 本文用 $p(i)$表示第 i 个样本的权重，P 表示分布。样本的分布可以通过对每个样本的权重进行归一化而得到。

算法的流程，3.1 和 3.2 节将详细讨论 P_{sen} 和 P_{smp} 的更新过程。3.3 节还将介绍如何修改最大熵模型的学习算法，使新的学习算法（*WeightedMELearn*）能够考虑每个样本的权重。

输入：一组已标注的句子$\{(s_i, t_i)\}$（训练集），其中 s_i 表示第 i 个句子，t_i 为其句法树

输出：一组移进-归约句法分析器$\{SR_1, SR_2, \ldots SR_T\}$

初始化：

1. 从已标注的句子中抽取分类样本

2. 将 P_{sen} 和 P_{smp} 设置成均匀分布

For $t = 1, \ldots, T$

1. 调用方法 *WeightedMELearn*，该方法根据分类样本的当前分布 P'_{smp} 来构建一个移进-归约句法分析器 SR_t

2. 计算 SR_t 在训练集上取得的性能：

$$r_t = \sum_{i=1}^{m} p'_{sen}(i) score_t(i) \tag{1}$$

3. 计算步长因子

$$\beta_t = \frac{1}{2} \ln \frac{(1+r_t)}{(1-r_t)} \tag{2}$$

4. 更新训练集中的句子的分布 P_{sen}

$$p'^{+1}_{sen}(i) = \frac{p'_{sen}(i) e^{\beta_t * error_t(i)}}{Z_t} \tag{3}$$

5. 根据 P_{sen} 更新 P_{smp}（3.2 节介绍两种不同的更新方法）

EndFor

图 1　基于 AdaBoost 的子系统构建算法

3.1　更新 P_{sen}

本小结主要讨论如何调整训练语料中句子的分布 P_{sen}（图 1 中步骤（4））。其具体过程如下：假设在第 t 轮迭代中，算法首先使用在该轮构建的移进-归约句法分析器 SR_t 对训练语料中的句子进行句法分析，SR_t 会在一部分句子中取得相对较低的 F1-score。为了让下一轮构建的句法分析器能够更好地处理这部分句子，更新句子权重的准则为：SR_t 在某句子中的得分越低，该句子的权重增长幅度就越大。公式（3）给出了更新第 i 个句子的权重的计算方法，其中 Z_t 为归一化因子。由公式（3）可知，权重更新的幅度主要由两部分决定：损失因子 $error_t(i)$，和步长因子 β_t。本文定义 SR_t 在第 i 个句子上的性能损失 $error_t(i)$ 为，

$$error_t(i) = 1 - score_t(i) \tag{4}$$

其中。由公式（4）可知，SR_t 在该句子上的得分越低，损失因子就越大，因此对该句子的权重的增大幅度就越大。

步长因子 β_t 的计算主要基于以下原则：对于性能相对较高的句法分析器仍然无法正确处理的句子应该给予更多的重视。定义 SR_t 在整个训练语料上取得的平均性能 r_t 为其在训练语料的每个句子上得分的期望，如公式（1），然后根据 r_t 的值，计算 β_t，如公式（2）所示。由公式（2）可知，SR_t 的平均性能越高，则步长因子 β_t 越大，因此，对于那些不能被 SR_t 正确处理的句子的更新幅度就越大。

3.2　根据 P_{sen}，更新 P_{smp}

假设 s_j 为训练语料中第 j 个句子并且 SR_t 在该句子上的性能损失大于 0，即 $error_t(j) > 0$。对于如何更新从 s_j 中抽取的分类样本的权重（注：对从训练语料的每个句子中抽取的全部分类样本

的权重进行归一化即可得到分类样本的分布 P_{smp}），本文使用了两种不同的方案。

方案一，从 s_j 中抽取的全部分类样本的权重都将被更新，方案一设置这些分类样本的权重与 s_j 的权重相同。方案二仅更新 SR_t 对 s_j 进行句法分析过程中的第一个错误决策所对应的分类样本的权重，并设置该分类样本的权重与 s_j 的权重相同。方案二与 Collins 和 Roark 在文献[11]中提出的'early update'类似，使用这种方案主要是由于移进-归约句法分析法的每一步决策都会对后续动作的决策产生影响，第一个错误决策的出现往往会导致更多后续的错误决策。因此，句法分析过程中的第一个决策错误所对应的分类样本应该受到更多的重视。

3.3 训练加权样本——WeightedMELearn

本文通过重新定义训练样本的经验分布从而使最大熵模型的学习算法能够根据训练样本的权重对模型的参数做相应的调整。令$\{(x_i, y_i)\}$为一组独立同分布的训练样本，其中 x_i 表示第 i 个样本的特征向量，y_i 为该样本的类别。本文定义训练样本的**加权经验分布** $\tilde{p}_w$ 为：

$$\tilde{p}_w(x_i, y_i) \equiv \frac{count(x_i, y_i) * w_{(x_i, y_i)}}{\sum_k count(x_k, y_k) * w_{(x_k, y_k)}} \tag{5}$$

其中 $w_{(xi,yi)}$ 表示训练样本（x_i, y_i）的权重。则整个训练样集的对数-似然值为:

$$\begin{aligned} & L(\{(x_i, y_i)\}) \\ & \equiv \log \prod_i q(y_i \mid x_i)^{\tilde{p}_w(x_i, y_i)} \\ & = \sum_i \tilde{p}_w(x_i, y_i) \log q(y_i \mid x_i) \end{aligned} \tag{6}$$

其中，$q(y|x)$是由最大熵模型定义的条件分布。通过使用如 L-BFGS[12]等方法优化目标函数（8）即可得到最大熵模型的参数的估计值。

4 子系统融合

本章主要介绍将各个子系统的输出进行融合的方法。假设对 m 个子系统进行系统融合，则对于测试集上的任何一个句子，首先用待融合的 m 个子系统分别对该句子进行句法分析，生成 m 棵句法树，然后用一个线性模型对这 m 棵句法树进行评分，并将得分最高的句法树作为最终结果。令 t 表示一棵句法树，则 t 的最终得分为：

$$g(t) = \sum_{i=1}^{m} \alpha_i * \log(p_i(t)) + \alpha_{m+1} * \log(p_{as}(t)) \tag{7}$$

α_i 为第 i 个特征的权重，权重的值可以使用最小错误率训练方法（文献[13]）来确定（将 MERT 的错误率函数定义为 1-F1-score）。公式（7）中的线性模型主要包含两类特征：第一类特征称为子**系统置信度**(类似于 Zhang 等在文献[14]中提出的模型置信度)，用 $p_i(t)$表示，即第 i 个子系统对句法树 t 的置信度。该置信度的计算方法如下：假设$<v_1, a_1>, \ldots, <v_k, a_k>$为从 t 中抽取的分类样本，其中，v_j表示第 j 个分类样本的特征向量，a_j表示第 j 个分类样本对应的动作，$q_i(a_j|v_j)$表示第 i 个子系统所使用的最大熵分类器对分类样本$<v_j, a_j>$输出的条件概率，则

$$p_i(t) = \prod_{j=1}^{k} q_i(a_j \mid v_j) \tag{8}$$

第二类特征为**扩展动作序列置信度** $p_{as}(t)$（与统计机器翻译中使用的语言模型特征类似），表示由$a_1, \ldots, a_k$组成一个扩展动作序列的概率。其中 a_j 是对动作 a_j 的扩展，其定义如下：

$$a_j \equiv \begin{cases} a_j\_POS(a_j) & a_j\text{为移进动作} \\ a_j & a_j\text{为规约动作} \end{cases} \tag{9}$$

$POS(a_j)$表示被 a_j 移进的元素的词性。我们通过观察发现，如果将原始的动作序列进行扩展以后，扩展的动作之间往往存在某些模式或联系。例如，生成由两个名词组成的名词短语的动作序列为：移进_NN, 移进_NN, 二元-NP-归约。因此，使用 $p_{\alpha}(t)$的目的就是希望通过对扩展动作序列的分布建模。从而对给定的一组扩展动作，可以从概率的角度来评价这组扩展动作生成一颗句法树的可能性。本文使用 N 元语言模型对扩展动作序列进行建模，并使用从训练集中抽取的扩展动作序列对语言模型进行参数估计。

5 实验

5.1 基准系统及实验数据

本文使用宾夕法尼亚大学的英文树库（Penn English Treebank）作为实验数据，并主要使用以下 3 个部分：02-21 节作为训练集，主要用于训练移进-归约句法分析器和 N 语言模型。22 节作为开发集，主要用于训练公式（9）中各个特征的权重。23 节作为测试集。我们删除了训练集和开发集中所有句法树的功能节点及空节点，并使用 Collins 在文献[15]中介绍的方法对句法树进行词汇化。我们实现了文献[10]中提出的移进-归约句法分析器作为本文实验的基准系统，与[10]中稍有不同的是，该句法分析器使用 Zhang[16]开发的最大熵工具作为其对移进-归约动作进行决策的分类器，分类器使用文献[10]介绍的特征进行分类。对于子系统生成部分，我们使用 3.3 节中提到的方法对最大熵模型的学习部分进行修改，使最大熵的学习部分能够同时考虑训练样本数量及权重。由于移进-归约句法分析器的输入是带有词性标注的句子，因此我们使用 SVMTool[17]作为本实验的词性标注器，该工具在测试集上的准确率为 96.81%。本文使用 EVALB 作为实验性能的评价工具。

5.2 实验结果及分析

第一组实验使用 5 元语言模型对所有子系统的输出进行融合，实验结果如表 1 所示。其中‘开发集’和‘测试集’分别表示在开发集及测试集上取得的性能，‘方案 1’和‘方案 2’分别表示使用 3.2 节介绍的方案 1 和方案 2 来更新分类样本的权重所取得的性能。‘$T(M)$’表示取得最佳性能时，参与融合的子系统的个数为 M。从表 1 中可以看出，对于开发集，基于方案 1 和方案 2 的系统融合后的性能分别比基准系统的性能提高了 2.09 和 2.16 个点，这说明了本文中提出的系统融合方法的有效性。对于测试集，基于方案 1 和方案 2 的系统融合后的性能则分别比基准系统的性能提高了 1.47 和 1.94 个点，这表明方案 2 更能有效的调整分类样本的权重，从而获得更好的系统融合性能。第二组实验主要是为了研究 N 元语言模型对系统融合的影响。在该组实验中，我们选择不同的 N 值，然后记录下融合后的系统在测试集上取得的性能[1]，实验结果如图 2 所示。从图 2 中可以看出，当 N 从 1 增大到 5 时，融合系统的性能随 N 的增加而提高，这表明使用基于 N 元语言模型的扩展动作序列置信度能够对最终结果的选择带来帮助。当 N 为 5 时，系统的性能达到最高，这表明使用 5 元语言模型对系统融合的帮助最大。当 N 从 5 增大到 10 的时候，融合系统的性能随 N 的增大而下降。这主要是由于随着 N 逐渐增大，数据稀疏问题越发严重，导致系统融合的性能逐渐下降。

[1] 在该组实验中，我们使用 3.2 节的方案 2 来更新分类样本的权重，并使用 12 个子系统进行整合，该设置是第一组实验中取得最高性能时的设置。

表1　系统融合在测试集和开发集上的性能

数据集	权值更新方法	Recall	Precision	F1-score
开发集	基准系统	81.98	86.52	84.19
	方案 1-T(9)	83.90	88.81	86.28
	方案 2-T(12)	83.92	88.93	86.35
测试集	基准系统	86.06	86.32	86.19
	方案 1-T(9)	87.50	87.81	87.66
	方案 2-T(12)	87.79	88.48	88.13

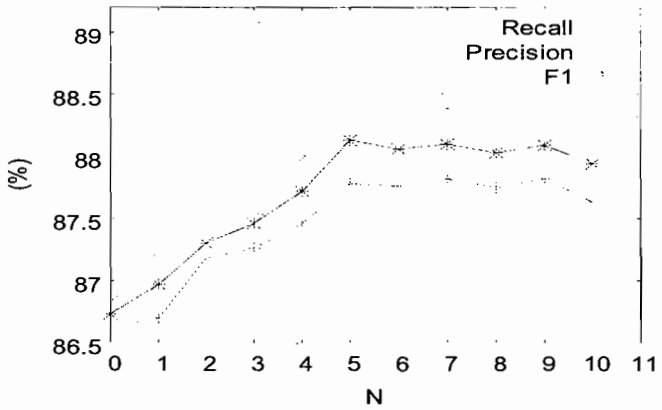

图2　系统融合的性能随 N 的变化曲线

6　相关工作

对于短语结构句法分析的系统融合，Henderson 和 Brill 在文献[4]中提出了两种不同的融合方法。第一种方法是将各个子系统输出的句法树进行相似度打分，得分最高的句法树作为最终结果。第二种方法是将各个子系统输出的句法树拆成一序列元组（constituent），通过统计每个元组出现的次数来判断该元组是否可能出现在最终结果的句法树中，其判断的标准为：如果一个元组在半数以上的句法树中出现，则该元组可能出现在最终结果中。Sagae 和 Lavie 在文献[5]中将这种方法做了进一步扩展，他们使用一个阈值来选择可能出现在最终句法树中的元组。Zhang 等人[14]从子系统的输出中选择一棵最优句法树作为最终结果，并使用模型置信度作为句法树质量的评价标准之一。以上方法与本文中方法的最主要区别在于，以上方法的系统融合都属于多模型系统融合，而本文中的系统融合则是基于单个模型。Henderson 和 Brill[7]研究了基于 Collins Parser 的单模型系统融合，然而，本文是对移进-归约句法分析器进行单模型系统融合。此外本文中提出的权重更新方法同时考虑到句子的权重和分类样本的权重，而且本文中的基于线性模型的系统融合方法能够同时考虑系统置信度和扩展动作序列置信度等特征。

7　总结

本文提出了一种面向移进-归约句法分析器的单模型系统融合的方法。在子系统生成阶段，根据移进-归约句法分析器的特点，本文提出了两种不同的权重更新方法来生成多套子系统。在子系统输出融合阶段，通过使用线性模型对各子系统输出的句法树进行评价，从而选出最终结果。此外，本文通过实验对比分析了两种权重更新方法的有效性以及线性模型中使用的特征对系统融合的影响。

参 考 文 献

[1] Yoav Freund and Robert Schapire. 2000. BoosTexter: A Boosting-based System for Text Categorization. Machine Learning, 39: 135-168.

[2] 李茂西, 宗成庆. 机器翻译系统融合技术综述. 中文信息学报, 2010, 24(04): 74-85.

[3] 杜金华, 等. 基于混淆网络解码的机器翻译多系统融合. 中文信息学报 2008, 22(04): 48-54.

[4] John Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: combining parsers. In *Proc. of* EMNLP 1999, pages 187-194.

[5] Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proc.* of HLT-NAACL 2006, pages 129-132.

[6] Yoav Freund and Robert Schapire. 1997. A decision theoretic generalization of on-line learning and an application to boosing. Journal of Computer and System Sciences, 55(1): 119-139.

[7] John Henderson and Eric Brill. 2000. Bagging and Boosting a Treebank Parser. In *Proc* of ANLP 2000, 34-41.

[8] Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of* ACL 1997, pages 16-23.

[9] Kenji Sagae and Alon Lavie. 2005. A Classifier-based Parser with Linear Run-Time Complexity. In *Proc.* of IWPT 2005.

[10] Kenji Sagae and Alon Lavie. 2006. A Best-First Probabilistic Shift-Reduce Parser. In *Proc.* of ACL-COLING 2006 (poster).

[11] Michael Collins and Brain Roark. 2004. Incremental Parsing with the perceptron algorithm. In *Proc.* of ACL 2004 pages 111-118.

[12] Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale Optimization. Mathematical Programming, 45: 503-528.

[13] Franz J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc.* of ACL 2003, pages 160-167.

[14] Hui Zhang, Min Zhang, Chew Lim Tan and Haizhou Li. 2009. K-Best Combination of Syntactic Parsers. In *Proc* of EMNLP 2009, pages 1552-1560.

[15] Michael Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. Phd thesis, University of Pennsylvania.

[16] Le Zhang. 2004. Maximum Entropy Modeling Toolkit for Python and C++ Reference Manual

[17] Jesús Giménez and Lluís Márquez. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Porc. of* LREC 2004 pages 43-46.