第七次作业（自动对联系统）

交叉信息研究院　交叉研14　　王国赛　　2014311423

2016年12月22日

本次实验我们基于phrase-based SMT（基于的短语的统计机器翻译）的思想实现了一个自动对对联（或对诗）的程序。该程序对于使用者给出的中文上联，会自动地对出字数相等的中文下联。我们实现的对联生成程序具有如下的特色：(1) 考量了末尾字的平仄；(2) 混合使用了PTM (Phrase translation model)，IPTM (Invertied phrase translation model)，LM (Language model)三种模型进行最优下联的生成；(3) 生成多个倾向性不同的候选下联供使用者选择。

本文接下来将分为如下小节：第1节介绍数据集来源和预处理，以及模型的预处理方式；第2节具体介绍对于使用者给出的上联如何生成对应的下联；第3节通过样例分析展示对联程序的效果；最后第4节对程序中的一些问题进行讨论，并提出可能进一步优化的方向。

1　数据集和模型的处理

1.1　数据来源和预处理

本文介绍的基于SMT的对联生成模型需要对仗的句对用于训练。数据的数量和质量对对联生成的效果至关重要。我们采用了网络学堂上提供的《全唐诗》古诗数据集，并筛选出了所有律诗的第二联和第三联（即颔联和颈联。此两联通常对仗），然后进行了数据去重处理（原始数据集中存在完整重复的诗句），最后得到的38270个句对作为程序训练用的数据集。

1.2　模型的预处理

对于对联生成模型，我们的预处理流程主要包括分词、生成对仗词表和统计词频三个步骤。我们分别简述如下。

1.2.1　分词

本文介绍的对联生成方法是基于短语的而不是基于单个字的，因而我们需要对训练样本，和使用者输入的上联进行分词处理。由于汉语尤其是古汉语的精简性，我们可能会得到许多长度为1的短语（即一字为一词），这种情况是可以接受的。

我们使用了一个公开的Python中文分词工具——“结巴”中文分词[1]，对于训练样本中的每一个句子，以及使用者输入的上联进行分词处理。第4节中讨论了使用公开分词工具可能存在的问题。

我们具有一个较强的先验知识，即对联的对仗特性决定训练样本中上下联的分词格式应当是一致匹配的。然而使用分词工具或自己实现的分词算法对于上下联的分词可能存在分歧。对于这种情况，我们采取一种简单的对“切分点”取并集的方式，对分词结果进一步处理，使得上下联分词结果相同。例如，分词工具对于句对“笑树花分色，啼枝鸟合声”的分词结果为：

笑|树花|分色
啼枝|鸟|合声

我们将其进一步处理为：

笑|树|花|分色
啼|枝|鸟|合声

1.2.2　对仗词表

我们构造一张对仗词表$T_{antithesis}$，该表每行为一个键值对<word, match_list>，其中match_list为所有曾经和词word形成搭配过的词构成的列表。对于每一个句对中，分词得到的每一个词w，我们都将它在对偶句中对应的词加入到对仗词表中以w为键的match_list中。例如，和“花”字形成匹配的汉字有：

叶:276 草:158 月:143 水:139 柳:127 鸟:121
竹:101 酒:73 雪:65 雨:60 ……

其中的数代表该词在数据集中和“花”字形成搭配的次数。从而对仗词表中键为“花”词的表项为：

$$T_{antithesis}[\text{花}] = \{\text{叶}, \text{草}, \text{月}, \text{水}, \text{柳}, \text{鸟}, \text{竹}, \text{酒}, \text{雪}, \text{雨}, \cdots\cdots\}$$

这张表将用于下联生成时，每个位置候选词的选取。我们在第2节中介绍具体细节。

1.2.3 统计词频

我们在分词完成之后对每个词的出现次数进行统计，并用一个哈希表记录每个词的词频（出现次数）。

2 下联的生成

使用者每次对程序输入一个上联（中文组成的不含标点的句子），程序生成下联的主要步骤为：

(1) 分词，
(2) 生成并筛选候选词表，
(3) 计算下联生成模型$P(S|F)$中的各子模型：PTM，IPTM，LM，
(4) 求解最优下联，返回结果给使用者。

我们分步骤介绍具体的下联生成方法。

2.1 对上联分词

我们用前文提到的分词工具对使用者输入的上联分词。我们逐一检查上联中的每一个词是否曾在出现数据集中出现过。如果某个词未曾在数据集中出现过，则我们将其拆开为单个汉字，并递归检查每个汉字是否存在于数据集中。若我们依然发现了未曾见过的汉字，则程序无法推断该字和何字形成对仗，这时程序停止下联生成并将情况告知使用者。

2.2 生成、筛选候选词表

我们逐一检查上联中的词，并将以其为键的对仗词表表项的值（即和该词形成过匹配的词的列表）记录下来，从而构造出候选词列表。形式化地，设上联FS（first sentence）已被拆分为$w_1 w_2 \cdots w_N$，其中N为上联分词得到的词数，则我们定义候选词表$T_{candidate}$为：

$$T_{candidate}[i] = T_{antithesis}[w_i](i = 1, 2, \cdots, N) \tag{1}$$

特别地，为了满足平仄的要求，我们使用汉字拼音转换工具pinyin[2]对上联最后一个词的最后一个字查询得到声调（1、2、3、4声）。如果上联末尾字是平声（1、2声），则我们删除$T_{antithesis}[w_N]$中所有的平声字以确保上下联末尾的平仄相对[1]。反之亦然。

为了减少后续计算（动态规划算法）的时间开销，我们对候选词表进行进一步筛选。我们筛去候选词表的生僻词（词频小于预设阈值的词），并依据每一项$T_{candidate}[i]$中每个词在数据集中和w_i形成对仗匹配的次数对$T_{candidate}[i]$各词进行降序排列，并挑选出$\min\{T_{candidate}[i].length, t\}$个词（$t$为预设的阈值），更新表项$T_{candidate}[i]$。

这样筛选符合我们的预期：我们希望我们对出的下联没有难以理解的生僻字/词，我们也希望下联中和上联对仗的词确实对仗（形成稳定匹配）。这样我们可以在计算最优下联之前，先筛掉明显不靠谱的候选词。

2.3 下联生成模型计算

我们参考了文献[3]中使用的部分模型，形式化地构造问题如下：设使用者输入的上联为$F = f_1, f_2, \cdots, f_N$，其中$f_k$是分词后得到的上联中的第k个词，我们的目标是对出下联$S = s_1, s_2, \cdots, s_N$，使得$P(S|F)$最大。我们定义最优下联$S^*$为：

$$S^* = \arg\max_S P(S|F) \tag{2}$$

$$= \arg\max_S \sum_i^M \lambda_i \log h_i(S, F) \tag{3}$$

其中$h_i(S, F)$是第i个特征函数，M是选用特征函数的个数。我们选取$M = 3$个基于短语的SMT领域常用的特征函数来计算S^*。表格1列举了我们选用的几个特征函数的含义。

Phrase translation model (PTM) 模型的核心是计算下联中的词s_i翻译（匹配、映射、对仗）为上联中的词f_i的概率$P(f_i|s_i)$[2]。我们通

[1]一般而言对联的上联应以仄声字结尾，下联以平声字结尾。但假如使用者的输入以平声字结尾，我们可以理解为程序在对给出的下联对上联。

[2]我也不知道为何此概率被称为PTM而不是被称为Inverted PTM。本文我们沿用了文献[3]的定义。

Table 1: 特征函数的定义

$h_i(S,F)$的定义	含义
$h_1(S,F)=\prod_{i=1}^{N} p(f_i\|s_i)$	Phrase translation model
$h_2(S,F)=\prod_{i=1}^{N} p(s_i\|f_i)$	Inverted phrase translation model
$h_3(S,F)=p(S)$	Language model

过词在数据集上相对频度来估计该概率：

$$P(f_i|s_i)=\frac{count(f_i,s_i)}{\sum_{k=1}^{m} count(f_k,s_i)} \tag{4}$$

其中m是数据集中所有和s_i形成对仗（匹配）的不同的词数，f_k是和s_i形成过对仗的词。

我们用类似的方式计算Inverted phrase translation model (IPTM)模型。其核心也是通过相对频度来估计概率：

$$P(s_i|f_i)=\frac{count(f_i,s_i)}{\sum_{k=1}^{m} count(s_k,f_i)} \tag{5}$$

我们通过训练数据集基于词的bigram的马尔可夫模型来构造语言模型$P(S)$。具体的，下联S形成的概率被估计为：

$$\begin{aligned} P(S) &= P(s_1,s_2,\cdots,s_N) \\ &= P(s_1)\prod_{t=2}^{N} P(s_t|s_1,\cdots,s_{t-1}) \\ &\approx P(s_1)\prod_{t=2}^{N} P(s_t|s_{t-1}) \end{aligned}$$

我们用相对频度来估计每一项$P(s_t|s_{t-1})$。对于出现次数为0的bigram，我们通过Lidstone平滑和Jeffreys-Perks法则来进行平滑处理，估计其相对频度。具体地，对于bigram w_2w_1，其概率被估计为

$$P(w_2|w_1)=\frac{count(w_2w_1)+\lambda}{count(w_1)+\lambda|V|} \tag{6}$$

其中$|V|$为单词表的长度。根据Jeffreys-Perks法则，我们取$\lambda=0.5$。

2.4 求解最优下联

回顾我们最终需要求解的是$S^*=\arg_S \max P(S|F)$，其中$S=\{s_1,s_2,\cdots,s_N\}, s_i\in T_{antithesis}[i]$。

由于我们采用的基于bigram的马尔可夫模型存在子结构（前缀子链）最优性，我们可以用动态规划算法来求解这个最优化问题。动态规划算法的初始条件和转移概率公式此处不赘述。

第2.3节中尚未解决的一个问题是参数$\lambda_k(k=1,2,3)$的选取（参见公式3）。我起初根据实验观测经验性地选取了一组固定的值，效果还不错。但是我想到如果我选取多组不同的λ_k参数，可以生成多个最优下联，这些下联可能风格各异各有优劣，因为不同的λ_k参数组合本身就决定了在多个因素中做权衡时的倾向性。例如，选取更大的λ_3倾向于求解得出语言模型概率更高，从而更加通顺的下联；而选取更大的λ_2则倾向于求解出对仗更加严谨工整的下联。因而我决定对于每一个输入的上联，我通过多组不同的λ_k参数的组合，生成了一系列候选的最佳下联，去重之后返回给使用者供其选择。在第3节我们观察对比程序生成的多个下联。

3 对联结果展示与分析

3.1 生成下联选例展示

在以下各例中，黑色字为使用者（我自己）输入的上联（或下联），程序输出的结果用蓝色句或橙色句表示。其中橙色的结果为我个人认为质量相对更好的结果。句子中的空格为程序进行分词的切分点。

风月 情 浓
江山 梦 断
松筠 意 霈
林园 日 暖
尘埃 路 远
江山 事 细

深悲 黄鹤 孤舟 远
独对 青山 万里 长

独对 西飞 旷海 深
独对 白云 万里 高
独对 青山 万里 寒
独对 青山 万里 高

海 上 生 明 月
山 中 有 白 云
江 中 死 暗 风
山 中 落 白 风
山 中 落 白 云
春 风 起 白 云

芳菲 次第 常 相续
芒刺 参差 远 著行
芒刺 参差 不 著行
芒刺 方圆 每 著行

鱼戏 莲叶 东
露凝 稻花 北
露凝 渔舟 晚

苟 利 国 家 生死 已
佯 云 天 处 礼仪 犹
佯 铓 乡 氏 礼仪 犹
徒 云 山 色 白头 时
佯 美 人 处 礼仪 犹
徒 云 天 处 礼仪 犹
徒 云 山 客 礼仪 犹
徒 云 山 路 礼仪 犹
不 长 江 上 白头 时

江上 可 采莲
天涯 堪 拾翠
怀中 堪 拾翠
人间 不 沽酒
天涯 堪 渔父

烟花 易 冷
日月 难 寒
星象 难 寒
钟鼓 不 归

冬尽 今宵 促
年开 明日 催
年开 明日 长
年开 何处 来

冰消 出 镜 水
梅散 来 钩 山
梅散 过 江 山
梅散 归 钩 山
梅散 入 云 山
梅散 来 花 山
梅散 落 花 枝

一帆 风雨 路 三千
三径 烟霞 人 十二
千里 关山 云 一片
三径 泥沙 门 十五
三径 烟霞 山 十二

黄河 入 海 流
华岳 归 山 色
白云 归 山 色
华岳 临 江 落
白云 无 人 去
华岳 归 山 落

3.2 暴露局限性的对联选例

我们通过一些例子展示我们提出的基于Phrase-based SMT思想的下联生成程序的局限性。

3.2.1 上联存在比喻的修辞手法

本体和喻体的语义映射关系难以刻画。如：

遥 山 丽 如 绮
近 水 清 似 弦
近 水 清 似 珠
远 水 清 似 云
迥 水 秾 似 貅
远 水 风 入 云
近 水 秾 似 貅
远 水 清 似 貅

3.2.2 上联意象集中或相关联

bigram模型难以刻画跨度较大的意象间的相关性。如：

晴川 历历 汉阳 树
春草 依依 江上 山
春草 依依 书札 山
春草 依依 武陵 人
春草 疏疏 武陵 溪
春草 疏疏 书札 山

3.2.3 上联包含专有名词

分词工具需要将人名等专有名词纳入考虑。例如可以加入额外的人名词库。否则下联生成模型无法对人名或其他专有名词形成良好的匹配。

生子 当 如 孙 仲 谋
乖离 今 似 女 伶 反
乖离 不 似 去 云 断
乖离 入 似 女 伶 反
乖离 不 为 客 老 去
乖离 入 似 女 公 作

4 讨论与未来改进方向

4.1 语料库大小

增大数据集总是可以进一步提升算法的性能。文献[3]从网络上爬取了数十万条对联数据（包含对仗古诗句），而我们的模型仅用3万余条古诗句对进行训练，因而对于许多上联不能对出如意的下联。

另外语料库的一种特殊形式：特殊词库对于下联生成模型质量的提升也是显著的。例如文献[3]的作者曾经介绍过他们开发的系统对出的一个下联：

李敖对联强
鲁迅绝句多
(横批)语妙天下

如果没有特殊词库，可能对联生成模型不会识别出李敖这个词来（李敖一个现代人，古代语料库里应当不怎么出现这个人名）。

4.2 分词工具

第1.2.1节中介绍了我们采用的分词工具。该分词工具是由现代汉语语料库训练得到的，因而在对于古汉语的分词时可能存在相对更大的误差。不过在实验中我们观察到对大多数的古诗句，分词工具的准确度还是可以接受的（这一定程度上是因为古文的精炼性导致一字一词的现象普遍）。一种改进方式是基于我们采用的数据集重新训练分词工具的模型，重新训练下联生成模型。

4.3 Bigram模型

Bigram模型对于距离较远、跨度大的词之间的关系刻划能力弱。但是用更高阶的N-gram模型可能存在稀疏性带来的问题。或许我们可以尝试利用神经网络模型进行sequence到sequence的建模刻划。但是神经网络模型也对训练集的规模提出了更高的要求。

4.4 语言模型的平滑技术

本文采取的是简单易实现的Lidstone平滑。采用一些回退技术例如Katz backoff平滑技术或许可以进一步提升语言模型的鲁棒性。具体的表现需要通过实验来论证。

4.5 情感分析或主题分析

上下联体现的情感或主题应当相似或相关。我们可以尝试一些对语义的挖掘，使得对出的下联在语义层面上和上联的情感或者主题更加一致。例如，表达壮志报国的上联，对出表达闺怨的下联就不是很恰当。

4.6 格律的优化

格律带来的约束条件相对容易实现，我们只需要筛掉声调不符合要求的候选字（词）即可。我没有加入过多格律的限制主要是对联相对于古诗而言格律要求较弱，另外也因为我本身不太懂格律相关的知识。

一个潜在的问题是多音字读音的判断。我使用的汉字转换拼音工具对于多音字只是输出了概率最大的声调，但不能通过上下文准确判定该环境下的读音。这是另一个NLP问题了，通过无监督学习可能难以实现，毕竟字的读音是需要标注的。

4.7 对重复字的挑选或排除

由于时间所限，我实现的模型未考虑字重复的问题。一般而言，上联中若出现叠字词，下联也应当用叠字词匹配。例如：

晴川历历汉阳树
芳草萋萋鹦鹉洲

此外上联和下联中的相同位置或不同位置，或者下联内部不同位置间一般不会出现重复的字。这些逻辑可以在动态规划算法计算每条最

优子路径时通过和前文的检查进行实现，也可以通过更加严谨准确的方式去实现。这样可以进一步提升生成下联的质量。

References

[1] Jieba中文分词 https://github.com/fxsjy/jieba

[2] 汉字拼音转换工具（Python版） https://github.com/mozillazg/python-pinyin

[3] Jiang L, Zhou M, He J. Generating Chinese Couplets and Quatrain Using a Statistical Approach[C]// International Conference on Computational Linguistics. 2008:377--384.