中国科学: 信息科学 2017 年 第 47 卷 第 8 期: 1036–1050
SCIENTIA SINICA Informationis
从大数据到大知识工程专刊 · 论文

利用编码器 – 解码器学习依存边翻译规则表示

陈宏申[1,2*], 刘群[1,3]

1. 中国科学院计算技术研究所智能信息处理重点实验室, 北京 100190, 中国
2. 中国科学院大学, 北京 100190, 中国
3. Dublin City University, Dublin 999014, Ireland
* 通信作者. E-mail: chenhongshen@ict.ac.cn

收稿日期: 2016–03–08; 接受日期: 2016–04–02; 网络出版日期: 2017–06–20
国家自然科学基金 (批准号: 61379086) 资助项目

摘要 统计机器翻译模型, 特别是基于句法的翻译模型, 其翻译单元在保留足够的翻译信息以及翻译单元在翻译新句子时的泛化能力上始终存在着一个平衡. 神经网络被成功用于统计机器翻译模型中的调序和端到端机器翻译中. 本文提出了一个新颖的基于神经网络的句法翻译规则编码解码器, 依存边转换翻译规则编码解码器 (DETED), 它利用一条转换翻译规则的源端以及源端的上下文作为输入, 以依存边转换翻译规则的目标端作为输出, 学习依存边转换翻译规则的源端依存边到目标端依存边的匹配关系. 它不仅保留了依存边 —— 这种最简单的句法翻译规则的灵活性, 保证了翻译规则的泛化能力, 同时通过上下文信息增强了转换翻译规则的匹配能力. 编码器 – 解码器的结构非常简洁, 它将翻译规则的源端作为输入, 同时解码翻译规则目标端的对应翻译并判别依存边的位置关系. 使用编码解码器对解码时所用到的依存边转换翻译规则打分. 在 3 个 NIST 测试集上的实验显示, 相较于基线系统, 平均有 1.39 个 BLEU 的提升.

关键词 依存边, 转换, 机器翻译, 神经网络, 编码器, 解码器

1 引言

近年来统计机器翻译见证了一系列的发展. 从翻译单元的角度, 可以分为基于词的翻译模型 [1]、基于短语的翻译模型 [2,3] 和基于句法的翻译模型 [4~13]. 在统计机器翻译中, 有两个重要因素, 分别是规则抽取和解码的过程. 前者关注如何定义翻译知识的形式, 并从双语平行语料中抽取翻译知识. 后者则利用翻译知识, 进行翻译. 翻译规则作为翻译知识的表示形式, 是核心组件之一, 其在相当程度上决定了统计机器翻译模型的表达能力.

相较于其他的翻译模型, 基于句法的模型在一些方面具有特别的吸引力. 由于有句法信息作为指导, 它能产生更符合句法结构的翻译, 并且能更好地处理长距离依存以及调序关系. 在基于句法的翻译

引用格式: 陈宏申, 刘群. 利用编码器 – 解码器学习依存边翻译规则表示. 中国科学: 信息科学, 2017, 47: 1036–1050, doi: 10.1360/N112016-00281
Chen H S, Liu Q. Learning dependency edge transfer rule representation using encoder-decoder (in Chinese). Sci Sin Inform, 2017, 47: 1036–1050, doi: 10.1360/N112016-00281

模型当中, 依存句法树结构被视为一种既包含了句法信息又包含了浅层语义的结构. 研究人员基于依存句法树构建了很多翻译模型.

Lin [8] 使用路径作为转换翻译结构. 路径是依存树的片段, 可以视为一个连续的或者非连续的短语. 每条路径只有一个根节点. 在翻译的时候, 他通过将相同根节点的不同路径合并起来生成翻译. Quirk 等 [10] 则使用树杈 (treelet) 进行树到树的翻译, 树杈是一个联通的依存树的子树. Shen 等 [11] 则在层次短语翻译规则的目标端增加 fixed 和 floating 的依存树结构构建串到依存树的翻译模型, 其中 fixed 结构是头节点 —— 依存节点集合中, 包含头节点的一颗依存子树, 而 floating 结构是仅包含依存节点集合的连续短语子树. Xie 等 [12] 使用头节点 —— 依存节点集合作为基本的翻译单元, 提出了基于依存树到串的翻译模型. 研究人员探索了不同的方法用于解构依存树, 采用同步文法 (除了 Lin [8]) 进行翻译.

Chen 等 [14] 提出了一种基于依存边的非同步过程的翻译模型. 他们将依存树最基本的单元 —— 依存边, 投射到目标端, 然后通过组合目标端依存边来生成目标端的句子. 他们提出的翻译单元相当简单, 一条依存边在依存树中捕捉一个头节点和一个依存节点的修饰关系, 在翻译的过程中具有相当的灵活性. 在图 1(b) 中, 规则①的源端依存边, "奥巴马" 是一个依存节点, 它修饰头节点 "发布". 相应地, 目标端依存边, "obama" 修饰 "issued". 特别地, 为了缓解翻译规则的数据稀疏问题, 他们甚至泛化一条转换翻译规则中的头节点或者依存节点. 在图 1(c) 中, 规则①中的依存节点 "奥巴马" 被泛化成了一个变量或者头节点 "发布" 被泛化成一个变量. 在翻译的过程中, 他们通过组合目标端依存边的方式来生成翻译句子, 因而其调序更加灵活.

尽管 Chen 等 [14] 所提出的依存边在翻译中有着较强的灵活性, 然而一条依存边仅包含一个头节点和一个依存节点, 因而翻译规则的所有上下文都被忽略了, 其上下文匹配的能力则明显受限. 特别地, 这种上下文的匹配能力还由于头节点和依存节点的词被泛化而遭到进一步削弱. 另一方面, 依存边这种最基本的结构单元有时还会遭遇歧义过大的问题. 图 1(d) 中, 由于 "发布" 在不同的上下文环境中存在不同的翻译, 同一个源端依存边 ⟨ 发布, 奥巴马 ⟩ 可以对应于多条目标端的依存边. 而且, 头节点和依存节点的泛化还会进一步恶化这个问题. 在解码的过程中, 过多的目标端依存边将极大地膨胀解码空间, 导致合理的搜索变得较为困难.

然而, 如果在规则表示上增加上下文信息, 那么翻译规则的上下文匹配能力会因此得到加强, 但却失去了翻译规则表示的灵活性, 并且在双语平行语料较少时, 低频的依存边上会更加稀疏. 因此, 有没有可能在增强翻译规则的上下文匹配能力同时又不牺牲它的知识表示的简洁性和应用上的灵活性呢? 同时, 由于一条依存边不仅描述了头节点的词和依存节点的词在源端和目标端翻译上的对应关系, 还描述了它们之间位置上的对应关系, 如左右关系、相邻或者不相邻的断续关系, 如何既增强模型翻译上的对应能力, 同时增强位置关系上的区分能力呢?

近年来神经网络被成功地运用于统计翻译模型的调序 [15,16] 和语言生成 [17~21] 之中. 本文提出了一种新颖的基于神经网络结构的依存边转换翻译规则编码解码器去解决这些问题. 依存边转换翻译规则编码解码器采用一个转换翻译规则的源端依存边作为输入, 以转换翻译规则的目标端依存边作为输出. 为了增强其上下文匹配能力, 同时将源端头节点和依存节点的上下文输入到编码解码器中. 当解码时, 使用它计算转换翻译规则的候选目标端依存边的概率, 然后将每个目标端依存边的概率作为新特征, 加入到对数线性框架中. 在 3 个 NIST [1)] 的测试集上, 依存边转换规则编码解码器平均有 1.39 个 BLEU [22] 的提升.

1) NIST, National Institute of Standards and Technology (美国国家标准与技术研究院). 该单位曾经组织机器翻译评测, 被业内称为 NIST 评测, 测试所用数据被称为 NIST 测试集.

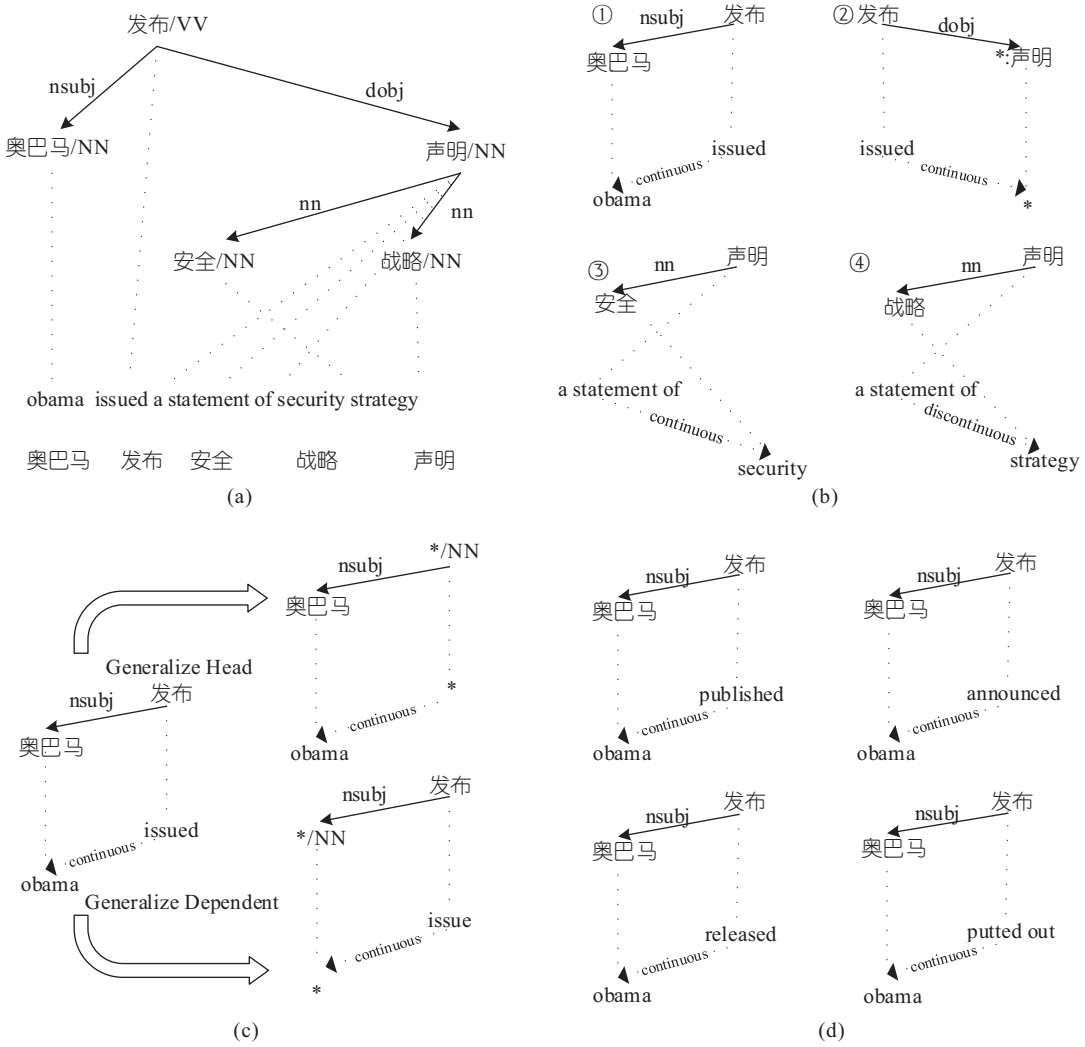

图 1　依存边转换翻译规则

Figure 1　Dependency edge transfer rules. (a) Dependency tree and word alignment; (b) extracted dependency edge transfer rules; (c) generalisation of transfer rules; (d) ambiguities of dependency edge transfer rules

2　基于依存边转换的翻译

依存边转换翻译规则从一棵源端依存树及其目标端译文和它们的词对齐中抽取. 图 1(b) 是图 1(a) 中抽取的依存边转换翻译规则. 基于依存边转换的翻译使用依存边作为其翻译单元. 一条边描述了头节点和依存节点之间的关系. 一条转换翻译规则由两条边组成, 一条源端的边和一条目标端的边. 源端依存边是通过句法分析得到的依存边, 将源端依存边定义为一个四元组 $\langle H_{\mathrm{src}}, D_{\mathrm{src}}, P_{\mathrm{src}}, R\rangle$, H_{src} 是源端依存边的头节点, D_{src} 是源端依存边的依存节点, P_{src} 是源端依存边头节点与依存节点的左右相对位置关系标记, R 是头节点与依存节点的依存关系标记. 在图 1(b) 中, 转换翻译规则①的源端依存边的头节点是 “发布”, 其依存节点是 “奥巴马”. 其依存关系标记 “nsubj” 表示一个名词作为主语. “奥巴马” 出现在 “发布” 的左边. 与源端依存边不同, 目标端依存边是通过源端依存边与源语言与目标语言的词对齐投射而来, 定义目标端依存边为另外一个四元组 $\langle H_{\mathrm{tgt}}, D_{\mathrm{tgt}}, P_{\mathrm{tgt}}, C\rangle$, H_{tgt} 是目标端依存

边的头节点, D_{tgt} 是目标端依存边的依存节点, P_{tgt} 是头节点与依存节点的左右相对位置关系标记, C 是头节点与依存节点相邻或者不相邻的断续位置关系标记. 对应地, 第一条转换翻译规则的头节点的目标端边的头节点是 “issued”, 依存节点是 “obama”. “obama” 出现在 “issued” 的左边, 且二者邻接.

依存边转换翻译规则还会通过泛化头节点或者依存节点来缓解数据稀疏的问题. 图 1(c) 中分别泛化了头节点和依存节点. 其中, 泛化头节点意味着 “奥巴马” (obama) 可以修饰任何动词, 这些依存边共享了相同的结构. 而泛化依存节点则表示任何一个名词都能修饰 “发布” (issued).

依存边转换翻译模型的翻译过程如图 2 包括 3 个部分. 对于一条输入的句子, 首先通过依存句法分析器得到其句法分析树. 然后, 利用依存边转换翻译规则, 将源端依存边投射到目标端. 最后, 通过组合目标端依存边, 生成目标端译文. 在每一个内部节点使用 Beam-search 算法, 枚举搜索目标端依存边的最优组合.

在基于依存边转换的翻译模型中, 其转换规则相当灵活. 因此, 目标端的句子生成在调序上非常灵活. 在目标端依存边上, 其仅有的位置约束包括头节点和依存节点的左右相对位置关系以及二者相邻或者不相邻的断续位置关系. 然而, 由于源端和目标端的对应关系是仅通过依存边建立起来的. 在实际的应用之中, 一条源端的依存边可能对应到数十乃至数百个目标端的边. 图 1(d) 展示了与图 1(b) 中规则①的源端依存边相同, 但目标端依存边不同的 4 条依存边转换翻译规则. 在规则抽取中, 它们的头节点 “发布” 在不同的上下文环境下存在 4 个不同的翻译候选. 但是, 在解码的过程中, 过多的 (且经常是不合适的) 候选目标端依存边, 给解码的过程中选择上下文匹配的目标端的边大大增加了难度. 因而导致在目标端句子的生成过程中面临了一个巨大的搜索空间. 此外, 如图 1(c) 中泛化了头节点或者依存节点的翻译规则, 虽然缓解了数据稀疏的问题, 但却是依存边转换翻译规则的歧义性的另一处来源.

3 依存边转换翻译规则编码解码器

本文提出了依存边转换翻译规则编码解码器 (DETED), 直接学习将源端的依存边匹配投射到目标端. 它不仅能增强依存边转换规则的上下文匹配能力, 而且由于它直接使用源端依存边四元组及其上下文所形成的数值向量解码目标端依存边, 其泛化能力更强, 因而在面临依存边在双语平行训练语料中不存在或出现次数较少而导致的数据稀疏问题时, 其能够通过词向量本身聚类效应, 缓解数据稀疏问题, 与此同时, 它还保存了依存边转换翻译规则的灵活性, 不增加翻译规则本身在信息存储上的负担.

依存边转换翻译规则编码解码器包含了一个编码器和一个解码器. 编码器以一条转换翻译规则的源端的边的所有信息作为输入.

解码器则包含了 4 个部分, 对应于目标端的边的 4 个组成元素 (目标端的头节点、目标端的依存节点、左右相对位置标记和断续标记), 包括目标端头节点解码器、目标端依存节点解码器、左右判别器和断续判别器. 图 3 展示了依存边翻译规则编码解码器的主体结构.

给定一条源端的边 e_{src} (以及其上下文信息), 希望得到目标端的边 e_{tgt}. 由于目标端的边包含了 4 个元素, 目标端头节点 (head_{tgt})、目标端依存节点 (dep_{tgt})、左右位置标记 (lr_{tgt}) 和断续标记 (cd_{tgt}). 假设 4 个元素相互独立, 只条件依赖于源端的依存边, 那么依存边转换规则解码器可以被定义为

$$p(e_{\text{tgt}}|e_{\text{src}}) = p(\text{head}_{\text{tgt}}|e_{\text{src}}) \times p(\text{dep}_{\text{tgt}}|e_{\text{src}}) \times p(\text{lr}_{\text{tgt}}|e_{\text{src}}) \times p(\text{cd}_{\text{tgt}}|e_{\text{src}}). \tag{1}$$

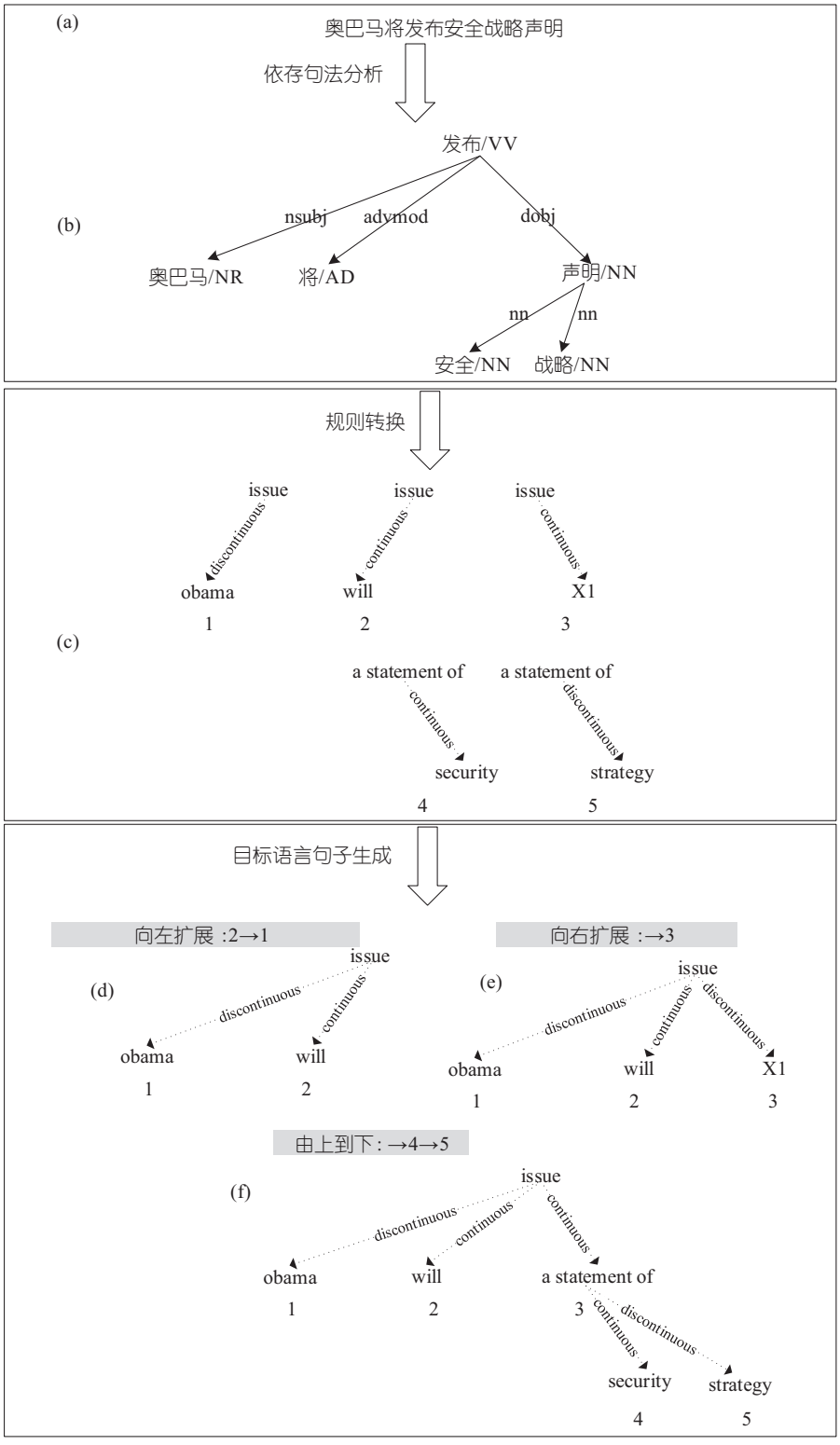

图 2　依存边转换翻译过程

Figure 2　Dependency edge transfer translation process. (a), (b) Analysis; (c) transfer; (d)∼(f) generation

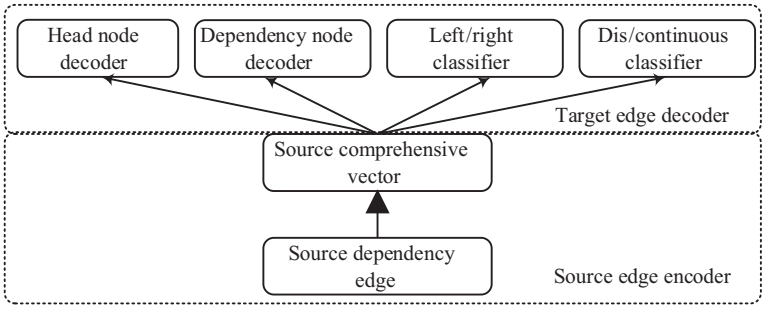

图 3 依存边转换翻译规则编码解码器

Figure 3 Dependency edge transfer rule encoder-decoder

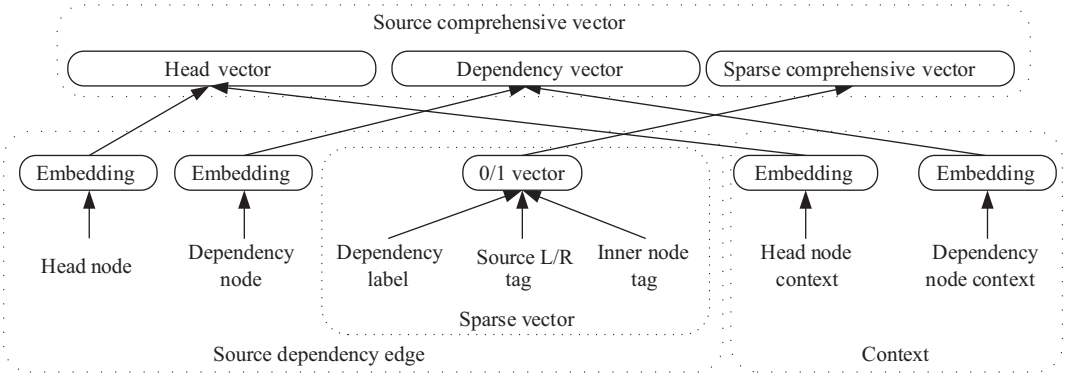

图 4 源端依存边编码器

Figure 4 Source dependency edge encoder

3.1 源端依存边编码器

源端依存边编码器使用前馈神经网络将源端的边编码成一个固定长度的源端信息综合向量 ($\boldsymbol{s}$). 在一条依存边转换翻译规则中, 一条源端的边仅包含 4 组信息. 此外, 还将源端头节点和依存节点的上下文作为输入, 增强转换翻译规则的上下文匹配能力. 通过一个位于源端头节点和依存节点的文本窗口, 取固定数量的上下文的词作为额外的输入. 在实验中, 将验证上下文在多大程度上增强了转换翻译规则的上下文匹配能力.

由于编码器的源端的边的四元组包含不同信息类型. 头节点和依存节点是词, 而左右位置关系标记和依存关系标记则是标记信息. 因而, 区分这两种类型的信息作为输入. 图 4 是编码器的结构示意图.

对于头节点 head_{src} 和依存节点 dep_{src}, 分别映射成头节点词向量 $\boldsymbol{e}_{\text{head}}$ 和依存节点词向量 $\boldsymbol{e}_{\text{dep}}$. 假设头节点在源句子词序列 $(w_1, w_2, \ldots, w_{n-1}, w_n)$ 是第 i 个词, 如果上下文窗口大小为 c, 则取词序列 $w_{i-c}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+c}$ 组成头节点的上下文 head_{con}. 然后将其映射成头节点上下文词向量 $\boldsymbol{e}_{\text{headcon}}$. 将头节点上下文词向量与头节点的词向量连接成 $\boldsymbol{x}_1$. 最后, 经过非线性变换, 得到头节点综合向量 $\boldsymbol{h}_{\text{head}}$:

$$\boldsymbol{x}_1 = \boldsymbol{e}_{\text{head}} \oplus \boldsymbol{e}_{\text{headcon}}, \quad \boldsymbol{h}_{\text{head}} = g(w_1 \boldsymbol{x}_1 + b_1),$$

其中, g 是 sigmoid 函数, $\oplus$ 是连接操作, w_1 为模型参数, b_1 为偏置项.

类似地, 依存节点词向量 $\boldsymbol{e}_{\text{dep}}$ 以及其上下文词向量 $\boldsymbol{e}_{\text{depcon}}$ 也被连接成 $\boldsymbol{x}_2$. 经过非线性变换, 得到依存节点综合向量 $\boldsymbol{h}_{\text{dep}}$:

$$\boldsymbol{x}_2 = \boldsymbol{e}_{\text{dep}} \oplus \boldsymbol{e}_{\text{depcon}}, \quad \boldsymbol{h}_{\text{dep}} = g(w_2\boldsymbol{x}_2 + b_2),$$

其中, w_2 为模型参数, b_2 为偏置项.

而对于标记信息, 则组合成 0/1 稀疏向量作为输入. 其中, 对于依存标记 deplabel, 定义映射特征函数为

$$f_{\text{deplabel}} = \begin{cases} 1, & \text{if} \quad \text{label} = \text{deplabel}, \\ 0, & \text{otherwise}. \end{cases}$$

对于左右相对位置关系 lrtag, 定义映射特征函数为

$$f_{\text{left/right}} = \begin{cases} 1, & \text{if} \quad \text{lrtag} = \text{left}, \\ 0, & \text{otherwise}. \end{cases}$$

此外, 还将依存节点是内部节点或者是叶节点也作为稀疏向量的组成部分, 定义其映射函数为

$$f_{\text{inner/leaf}} = \begin{cases} 1, & \text{if} \quad \text{dependent node is an inner node}, \\ 0, & \text{otherwise}. \end{cases}$$

稀疏向量 $\boldsymbol{x}_{\text{sparse}}$ 被直接转化成稀疏综合向量 $\boldsymbol{h}_{\text{sparse}} = g(w_3\boldsymbol{x}_{\text{sparse}} + b_3)$, 其中 w_3 为模型参数, b_3 为偏置项. 然后, 这 3 个向量被连接成源端综合信息向量 $\boldsymbol{s} = \boldsymbol{h}_{\text{head}} \oplus \boldsymbol{h}_{\text{dep}} \oplus \boldsymbol{h}_{\text{sparse}}$.

3.2 依存边转换规则解码器

给定源端信息综合向量 $\boldsymbol{s}$, 依存边转换翻译规则解码器产生目标端依存边. 目标端的边的 4 个元素同时产生. 与源端的边对应, 目标端的边同样的 4 个元素也包含两种类型. 目标端头节点 (head_{tgt}) 和目标端依存节点 (dep_{tgt}) 为字符串, 左右相对位置标记 (lr_{tgt}) 和断续位置标记 (cd_{tgt}) 可以视为二元分类的标记.

使用两种不同的网络产生目标端. 目标端依存边为源端依存边经过词对齐投射而来, 在出现源端到目标端一对多的对齐时, 或者源端到目标端一对一的对齐, 但目标端的对齐词周围有对空词, 在依存边转换翻译规则做对空词的拓展时, 目标端的头节点或者依存节点常常不是一个单词, 而是一个短语. 如图 1(b) 的规则③和④中, 源端依存边头节点 “声明” 所投射的目标端依存边头节点为 “a statement of ”, 其目标端依存边头节点由 3 个词的短语组成. 而规则①和②的目标端头节点则为一个词. 目标端的依存节点也存在类似的现象. 因而, 使用两个循环神经网络 (RNN) 解码目标端头节点和依存节点. 对于两个位置关系标记, 由于这里的位置关系判别是一个二分类的问题, 因而, 采用两个前馈神经网络进行判定.

目标端头节点和依存节点的解码器结构一致. 以目标端头节点的解码为例, 图 5 中循环神经网络的隐藏状态可以计算为

$$h_i = g(w_h h_{i-1} + w_s\boldsymbol{s} + w_{\text{head}}\boldsymbol{e}_{\text{head}} + w_y\boldsymbol{y}_{i-1} + b_h),$$

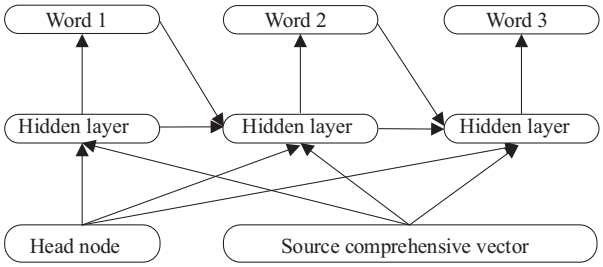

图 5 目标端头结点解码器

Figure 5 Target dependency edge head node decoder

其中, h_{i-1} 是上一个词对应的隐藏状态, $\boldsymbol{s}$ 是源端信息综合向量, $\boldsymbol{e}_{\text{head}}$ 是源端头节点词向量, $\boldsymbol{y}_{i-1}$ 是上一个产生的词的词向量, g 是 sigmoid 函数, w_h, w_s, w_{head} 和 w_y 为模型参数, b_h 为偏置项. 这里, 参考 Bengio 等 [23] 的方法, 通过连接输入层和输出层, 加速网络的收敛速度. 将源端头节点的词向量直接连接到目标端头节点的预测中, 因为源端头节点与目标端头节点的解码最为相关.

根据 h_i, 经过 softmax 分类函数即得到目标端头节点的第 i 个词的概率 y_{head_i}. 假设目标端头节点有 n 个词, 那么, 头节点的概率定义为 $y_{\text{head}} = \prod_i^n y_{\text{head}_i}$.

左右关系判定器和断续关系判别器的结构类似. 给定源端信息综合向量 $\boldsymbol{s}$, 由于左右位置关系和断续关系都是二元分类, 只需要知道两个分类结果中其中一个的分数便可得知另外一个分类的分数, 因而通过下面的式子进行预测:

$$f_{\text{lr}} = g(w_{\text{lr}} g(w_{\text{lrs}} \boldsymbol{s} + b_{\text{lrs}}) + b_{\text{lr}}),$$

其中, g 是 sigmoid 函数, w_{lr} 和 w_{lrs} 为模型参数, b_{lrs} 和 b_{lr} 为偏置项.

3.3 依存边转换规则编码解码器的训练

依存边转换规则的编码解码器的训练损失函数是由 4 个子损失函数组合而成, 这 4 个子损失函数, 对应于依存边转换规则解码器的 4 个组成部分. 定义损失函数如下:

$$J_\theta = J_{\text{head}} + J_{\text{dep}} + J_{\text{lr}} + J_{\text{cd}},$$

其中, J_{head} 是目标端头节点解码器的损失函数, J_{dep} 是目标端依存节点解码器的损失函数. J_{head} 和 J_{dep} 是左右位置关系和断续关系判别器的损失函数.

J_{head} 和 J_{dep} 采用负的对数似然函数:

$$J_{\text{head}} = \sum_n \log y_{\text{head}}^{n*}, \quad J_{\text{dep}} = \sum_n \log y_{\text{dep}}^{n*},$$

其中, y_{head}^{n*} 和 y_{dep}^{n*} 分别是目标端头节点和依存节点的概率.

J_{lr} 和 J_{cd} 则是间隔最大化函数, 但是这里定义间隔为 0:

$$J_{\text{lr}} = \max(0, y_{\text{lr}}^* - y_{\text{lr}}), \quad J_{\text{cd}} = \max(0, y_{\text{cd}}^* - y_{\text{cd}}),$$

其中, y_{lr} 和 y_{cd} 是正确分类的概率, y_{lr}^* 和 y_{lr}^* 是错误分类的概率.

这里左右位置关系判别和邻接与不邻接的断续位置关系判别的损失函数采用最大间隔函数, 以缓解标记偏置的问题. 对于一个训练实例, 当其正确的概率比错误的概率高出一定的间隔后, 便不再针

对该实例进行优化. 这里, 将间隔设置为 0, 这意味着一旦位置标记预测正确了, 即便是正确的概率只比错误的概率高一点 (如 0.51 比 0.49), 那么便不再通过误差反向传播使得正确概率变得更大. 如果采用负的对数似然损失函数, 即便正确的概率为 0.8, 也会因概率距离 1 仍存在 0.2 的差距而继续针对该实例进行优化.

4 解码

解码过程中, 使用依存边转换规则编码解码器来增强模型选择合适的目标端的边的能力. 具体而言, 为了使用该编码解码器, 在依存边转换翻译模型中增加一个新的特征. 将每一个目标端依存边在依存边转换规则编码解码器给出的概率作为特征, 加入到解码框架中.

在增加了新的特征后, 依存边转换翻译模型使用对数线性 [24] 框架, 寻找目标端的边的最佳组合. 它包含 13 个特征:

(1) 依存边转换翻译规则的前向和后向翻译概率;

(2) 依存边转换翻译规则的前向和后向词汇化翻译概率;

(3) 源端依存子树 fixed 短语片段的前向和后向翻译概率;

(4) 源端依存子树 fixed 短语片段的前向和后向词汇化翻译概率;

(5) 依存边转换翻译规则和源端依存子树 fixed 短语的规则数量惩罚;

(6) 虚构的依存边转换翻译规则;

(7) 目标端的词个数惩罚;

(8) 语言模型;

(9) 转换翻译规则的目标端依存边条件于源端依存边的概率.

算法 1 对依存边转换翻译规则编码解码器在解码时的使用过程进行了说明. 解码时, 给定一条源端的依存边, 解码器将源端依存边通过依存边转换翻译规则投射到目标端, 然后利用依存边转换翻译规则编码解码器, 计算该源端依存边为条件下的, 每一条候选目标端依存边的概率. 具体地, 将源端依

算法 1 在翻译解码时使用依存边转换翻译规则编码解码器计算目标端依存边的概率

Require:
源端依存树的节点 n;
依存边转换翻译规则集合 R;
依存边转换翻译规则编码解码器 D;
Ensure:
节点 n 作为头节点时, 所有源端依存边对应的候选目标端依存边的概率集合 P;
if n 不是叶子节点 **then**
  抽取该节点与其所有依存节点之间的源端依存边集合 E;
  for $e \in E$ **do**
    利用 R, 将 e 投射得到候选目标端依存边集合 F;
    将 e 输入到 D 中;
    在 D 的输出层, 计算 F 中每条候选目标端依存边的概率 p;
    将 p 放入集合 P 中;
  end for
  return P
end if

存边 (及其上下文) 作为依存边转换翻译规则编码解码器的输入, 然后获取候选目标端依存边的概率, 将其作为特征加入到对数线性框架中.

采用最小错误率训练 [25] 来调整翻译模型的权重参数.

5 实验

在中文 – 英文的翻译上设计了一系列实验以验证下列问题:

(1) 依存边转换规则编码解码器能够帮助模型更好地匹配源端和目标端的依存边吗?

(2) 不同的模块对翻译性能具体有多大的影响?

(3) 上下文信息在多大程度上能够增强模型的匹配能力?

(4) 解码时, 对解码速度有多大的影响?

训练语料包含了 260000 LDC (LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, LDC2005-T06) 数据的平行句对. 使用 NIST MT 评测的测试集 2002 作为开发集, 2003–2005 NIST 测试集作为最终的测试集.

使用 Stanford 句法分析器 [26, 27] 来分析源端的句子, 其输出为投射的源端依存树, 依存树中的各个节点的词被标注了词性标记, 并且依存树的各个边也被标记了依存标签. 使用 GIZA++, 在源端到目标端和目标端到源端两个方向上同时训练词对齐, 然后使用 "grow-diag-and" 精炼词对齐 [28]. 在 Gigaword 语料 Xinhua 的部分上, 使用 SRILM [29] 训练四元语言模型, 同时采用 Knser-Ney 平滑. 译文的质量评价则采用大小写不敏感的 BLEU-4 脚本[2)]. MERT 用于在开发集上调整参数权重, 以最大化 BLEU 值.

为了训练依存边转换规则编码解码器, 将源端和目标端的词汇表限制为 20000 个出现频率最高的词, 分别覆盖了大约 97.41% 和 99.13% 的中文和英文词. 所有的 OOV (out of vocabulary) 词都被映射成特殊的标签 UNK. 采用随机梯度下降, 以及 Adadelta [30] 训练模型. Batch 的大小设置为 64, 所有的参数维度设置为 300. 在目标端依存边的头节点和依存节点做对空词扩展后, 依存边转换翻译规则的训练实例数量达 7.5 M. 在实验条件下, 训练时间约需要一周.

5.1 系统

采用开源的基于短语的系统 Moses [28] (采用默认的配置) 作为基线系统. 基于依存边转换的翻译模型基线系统和增加了依存边转换规则编码解码器的模型采用相同的实验设置. beam 的阈值、大小和规则数量分别设为 10^{-3}, 100 和 100.

5.2 实验结果

表 1 展示了 3 个系统的 BLEU 值. 基线系统 DEBT 的 BLEU 好于开源的基于短语的系统 Moses. 在采用依存边转换规则编码解码器作为辅助后, 系统性能在 MT03, MT04 和 MT05 测试集上 BLEU 值分别提升了 1.23, 1.5 和 1.4. 这表明, 依存边转换规则编码解码器在解码时能够帮助解码器在解码的过程中选择更匹配的目标端的边, 以实现翻译性能的有效提升.

2) ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl.

表 1 BLEU-4 分数 (%) 在 NIST MT03∼05 测试集上 a)b)

Table 1 BLEU-4 scores (%) on NIST MT03∼05 a)b)

System	MT03	MT04	MT05	Average
Moses	32.30	33.43	31.44	32.39
DEBT	32.57	35.06	31.36	32.99
+DETED	**33.8***	**36.58***	**32.76***	34.38

a) "Moses" and "DEBT" are baseline systems.

b) "*" denotes the performance is significantly better than baseline system ($p < 0.01$).

表 2 不同模块下的 BLEU-4 分数 (%)

Table 2 BLEU-4 scores (%) of different components

System	MT03	MT04	MT05	Average
DEBT	32.57	35.06	31.36	32.99
+head_{tgt}	33.52	36.35	31.67	33.85
+dep_{tgt}	33.43	35.81	31.40	33.55
+lr_{tgt}	33.30	36.32	32.10	33.91
+cd_{tgt}	33.41	36.50	32.39	34.10
+DETED	33.80	36.58	32.76	34.38

5.3 不同模块对性能提升的影响

目标端的依存边包含 4 个部分, 即头节点 (head_{tgt})、依存节点 (dep_{tgt})、左右相对位置关系 (lr_{tgt}) 和相邻或不相邻的断续位置关系 (cd_{tgt}). 通过将目标端依存边条件于源端依存边的概率表示为某一个模块的概率, 来分析各个模块对性能的影响. 以头节点为例, 为了分析头节点的影响, 式 (1) 被修改为

$$p(e_{\text{tgt}}|e_{\text{src}}) = p(\text{head}_{\text{tgt}}|e_{\text{src}}), \tag{2}$$

其表示仅评估给定源端依存边时目标端头节点的概率. 此时, 目标端依存边的概率被定义为目标端头节点的概率. 类似地, 比较其他 3 个模块对性能的影响. 表 2 展示了各个模块的影响. +head_{tgt} 为仅评估目标端头节点的概率, +dep_{tgt} 为仅评估目标端依存节点的概率, +lr_{tgt} 为仅评估目标端头节点与依存节点左右位置类别的概率, +cd_{tgt} 为仅评估目标端头节点与依存节点是否邻接的断续位置关系的概率. 而 +DETED 则为采用式 (1) 时, 4 个模块同时用于计算目标端依存边的概率.

与基线系统比较, 各个模块单独使用时, 其性能均有不同程度的提升. 可以看到, 采用两个位置关系的分类概率 (33.91 和 34.10 BLEU), 其表现略好于采用目标端头节点或目标端依存节点概率 (33.85 和 33.55 BLEU). 但 4 个模块同时起作用时, 性能提升明显要高于各个模块单独使用时的效果. 我们认为, 这是因为 4 个模块能够从 4 个方面来同时对候选目标端依存边做出评估, 因而带来的总体效果提升上的收益好于各个模块单独使用的效果.

5.4 上下文窗口大小的影响

进一步地, 经验地分析上下文信息在多大程度上增强了依存边转换规则编码解码器的性能. 从表 3 看到, 即使没有任何上下文信息, 依存边转换规则编码解码器仍能在一定程度上取得性能提升 (均值: 33.99 比 32.99). 这在很大程度上是转换规则编码解码器的泛化能力所带来的. 在解码时, 翻译

表 3 BLEU-4 分数 (%) 在 NIST MT03~05 测试集上, 采用不同的上下文作为输入 c)d)

Table 3 BLEU-4 scores (%) on NIST MT03~05 test set, with different contexts as input c)d)

System	MT03	MT04	MT05	Average
DEBT	32.57	35.06	31.36	32.99
+nocon	33.56	36.06	32.37	33.99
+con1	33.80	**36.58**	**32.76**	34.38
+con2	33.73	36.52	32.45	34.23
+con3	**33.94**	36.24	32.49	34.22

c) "noncon" denotes none context as input.

d) "con1, con2, con3" denotes the context window is of size 1, 2, 3, respectively.

表 4 解码 NIST MT03 测试集的时间

Table 4 Decoding time cost on NIST MT03

System	Decoding time cost (s)	Diff
Moses	1081.71	–
DEBT	1090.89	–
+DETED	1297.83	+18.97%

解码器经常会碰到数据稀疏的问题, 甚至在抽取的规则中找不到完全匹配的源端依存边. DEBT 通过使用前面所提到的泛化后的依存边转换规则来进行翻译. 而泛化后的依存边, 其头节点或者依存节点被泛化成变量, 这进一步削弱了转换规则的上下文匹配能力, 且为翻译规则的选择增加了噪音和困难. 另一方面, 翻译规则的翻译概率和词汇化翻译概率均通过极大似然估计来学习, 因而在数据稀疏时, 其概率估计更可能因为某个噪声数据导致估计有偏. 而依存边转换规则编码解码器由于利用神经网络的向量连续的特征表示, 词向量本身也具有一定的聚类和相似属性, 其泛化能力更强, 因而能够缓解因翻译规则数据稀疏导致的概率估计偏差过大的问题.

当增加上下文的词汇作为输入, 在上下文窗口大小变为 1 时, 性能得到了预期中的提升, 这显示了近邻的词作为上下文信息的重要性. 然而, 当把上下文的窗口开的更大一点, 上下文窗口大小变为 2 和 3, 性能反而没有进一步提升. 在一定程度上, 这一方面表明了上下文信息 (特别是源端头节点和依存节点最近的左右两个词) 对翻译规则选择的重要性, 另一方面也表明上下文信息越多并不意味着模型的性能就一定越强. 在实验中, 上下文窗口为 1, 头节点和依存节点左右两边各选 1 个词时的性能已经足够好了. 而且, 更大的上下文窗口也意味着计算量的增加, 窗口大小为 1 时的计算代价也相对较小一些.

5.5 对解码速度的影响

依存边转换翻译规则编码解码器使用深度神经网络进行运算, 在解码过程中因计算量的增加对翻译的速度有一定的影响, 但这种影响有多大呢? 表 4 对不同的翻译系统在解码时的速度做了一次经验性的分析. 从表 4 中可看到, 依存边转换翻译模型的解码耗时与 Moses 相当. 在增加了本工作后, 相较于基线系统, 耗时仅增加了 18.97%. 特别地, 在实现时, 还通过对已经解码过的高频依存边转换翻译规则的概率进行缓存, 避免了部分重复计算.

6 相关工作

Schwenk [21] 提出了一个前馈神经网络对源端到目标端的短语片段进行打分. Devlin 等 [19] 也使用了一个前馈神经网络, 每次预测目标端短语的一个词. Auli 等 [17] 和 Cho 等 [18] 则使用 RNN 编码解码器结构, 学习短语/句子之间的表示. Kalchbrenner 等 [20] 提出使用卷积 n-gram 模型作为编码器和一个混编反向的 CGM 与 RNN 作为解码器. 这些模型都关注短语或者句子层面的建模. 他们的模型设计较为复杂. 与他们的模型相比, 我们则关注基于句法的翻译规则的表示学习.

Xiong 等 [31] 提出了最大熵模型用于预测两个将合并的短语块之间的调序关系. 与 Xiong 等 [31] 类似, Nguyen 等 [32] 提出了用于层次短语调序的模型. Li 等 [15, 16] 分别提出了用于括号转录语法 (inversion transduction grammar) 和基于短语的翻译的神经网络调序模型. 我们的模型主要关注一条依存边中的头节点和依存几点之间的位置关系. 不仅关注左右相对位置关系, 同时关注它们之间是否相邻.

7 结论

本文展示了一个基于神经网络的依存边转换翻译规则编码解码器, 用于增强依存边转换翻译模型的翻译规则匹配能力. 提出使用编码解码器的结构, 利用前馈神经网络作为编码器, 将源端依存边以及上下文编码成一个综合向量, 然后利用解码器解码目标端的依存边. 解码器一方面采用两个循环神经网络, 解码翻译对应信息 (目标端头节点和目标端依存节点), 另一方面还同时对位置关系 (目标端头节点左右相对位置关系以及断续位置关系) 进行分类判断. 采用依存边转换规则编码解码器后, 基于依存边转换翻译的基线模型提升了平均 1.39 个 BLEU. 此外, 分析了不同模块对性能提升的影响, 还展示了对于实验设置, 少量的上下文已经足够获得可观的性能提升.

参考文献

1 Brown P F, Pietra V J D, Pietra S A D, et al. The mathematics of statistical machine translation: parameter estimation. Comput Linguist, 1993, 19: 263–311

2 Koehn P, Och F J, Marcu D. Statistical phrase-based translation. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Edmonton, 2003. 48–54

3 Marcu D, Wong W. A phrasebased, joint probability model for statistical machine translation. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, Philadelphia, 2002. 133–139

4 Chiang D. A hierarchical phrase-based model for statistical machine translation. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Ann Arbor, 2005. 263–270

5 Ding Y, Palmer M. Synchronous dependency insertion grammars: a grammar formalism for syntax based statistical MT. In: Proceedings of the Workshop on Recent Advances in Dependency Grammars, 2004. 90–97

6 Graehl J, Knight K. Training tree transducers. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Boston, 2004. 105–112

7 Huang L, Knight K, Joshi A. Statistical syntax-directed translation with extended domain of locality. In: Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas, Cambridge, 2006. 66–73

8 Lin D. A path-based transfer model for machine translation. In: Proceedings of the 20th International Conference on Computational Linguistics, Geneva, 2004. 625–630

9 Liu Y, Liu Q, Lin S X. Tree-to-string alignment template for statistical machine translation. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, Sydney, 2006. 609–616

10 Quirk C, Menezes A, Cherry C. Dependency treelet translation: syntactically informed phrasal SMT. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Ann Arbor, 2005. 271–279
11 Shen L B , Xu J X, Weischedel R. A new string-to-dependency machine translation algorithm with a target dependency language model. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, Columbus, 2008. 577–585
12 Xie J, Mi H T, Liu Q. A novel dependency-to-string model for statistical machine translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Edinburgh, 2011. 216–226
13 Yamada K, Knight K. A syntax-based statistical translation model. In: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, Toulouse, 2001. 523–530
14 Chen H S, Xie J, Meng F D, et al. A dependency edge-based transfer model for statistical machine translation. In: Proceedings of the 25th International Conference on Computational Linguistics, Dublin, 2014. 1103–1113
15 Li P, Liu Y, Sun M S. Recursive autoencoders for ITG-based translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Washington, 2013. 567–577
16 Li P, Liu Y, Sun M S, et al. A neural reordering model for phrase-based translation. In: Proceedings of the International Conference on Computational Linguistics, Dublin, 2014. 1897–1907
17 Auli M, Galley M, Quirk C, et al. Joint language and translation modeling with recurrent neural networks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Seattle, 2013. 1044–1054
18 Cho K, van Merrienboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, 2014. 1724–1734
19 Devlin J, Zbib R, Huang Z Q, et al. Fast and robust neural network joint models for statistical machine translation. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, Baltimore, 2014. 1370–1380
20 Kalchbrenner N, Blunsom P. Recurrent continuous translation models. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Seattle, 2013. 1700–1709
21 Schwenk H. Continuous space translation models for phrase-based statistical machine translation. In: Proceedings of the International Conference on Computational Linguistics, Mumbai, 2012. 1071–1080
22 Papineni K, Roukos S, Ward T, et al. BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, 2002. 311–318
23 Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model. J Machine Learn Res, 2003. 1137–1155
24 Och F J, Ney H. Discriminative training and maximum entropy models for statistical machine translation. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, Philadelphia, 2002. 295–302
25 Och F J. Minimum error rate training in statistical machine translation. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, Sapporo, 2003. 160–167
26 Chang P C, Tseng H, Jurafsky D, et al. Discriminative reordering with Chinese grammatical relations features. In: Proceedings of the 3rd Workshop on Syntax and Structure in Statistical Translation, Boulder, 2009. 51–59
27 Levy R, Manning C. Is it harder to parse Chinese, or the Chinese treebank? In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, Sapporo, 2003. 439–446
28 Koehn P, Hoang H, Birch A, et al. Moses: open source toolkit for statistical machine translation. In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, Prague, 2007. 177–180
29 Stolcke A. Srilm—an extensible language modeling toolkit. In: Proceedings of the International Conference on Spoken Language Processing, Denver, 2002. 901–904
30 Zeiler M D. Adadelta: an adaptive learning rate method. arXiv: 1212.5701
31 Xiong D Y, Liu Q, Lin S X. Maximum entropy based phrase reordering model for statistical machine translation. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, Sydney, 2006. 521–528
32 van Nguyen V, Shimazu A, Nguyen M L, et al. Improving a lexicalized hierarchical reordering model using maximum entropy. In: Proceedings of MT Summit XII, Ottawa, 2009

Learning dependency edge transfer rule representation using encoder-decoder

Hongshen CHEN[1,2*] & Qun LIU[1,3]

1. *Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190*, China*;
2. *University of Chinese Academy of Sciences, Beijing* 100190*, China*;
3. *Dublin City University, Dublin* 999014*, Ireland*
* Corresponding author. E-mail: chenhongshen@ict.ac.cn

Abstract In existing statistical machine translation models, especially syntax-based models, there has always been a trade-off between the amount of information a translation unit preserves and its ability to generalize when translating new sentences. Neural networks have been successfully employed in reordering and end-to-end machine translation problems. In this paper, we propose a novel syntactic translation rule encoder-decoder based on neural networks. It is a dependency edge transfer rule encoder-decoder (DETED) that leverages the source side of a transfer rule and local context as input, and outputs the target side of that in order to learn the source-to-target matching of the dependency edge transfer rules. It shares not only the benefit of dependency edge, which is the most relaxed syntactic constraint, in order to ensure its generalization ability, but also the local context as additional information in order to improve its matching ability. The structure of the encoder-decoder is quite concise. With the source side of a translation rule as the input, it decodes the corresponding target side of the translation rule, and makes it clear the positional relation of the dependency edge. The generator is used to re-score the transfer rules when decoding. Experiments on three NIST test sets are presented. The results indicate a significant performance improvement with an average BLEU score of 1.39 above the baseline value.

Keywords dependency, transfer, machine translation, neural network, encoder, decoder

Hongshen CHEN was born in 1991. He received a B.S. degree in computer science from Sichuan University, Chengdu, China, in 2012. Currently, he is a Ph.D. candidate at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include natural language processing, machine translation, and deep learning.

Qun LIU was born in 1966. Dr. Liu is a researcher and professor. His research interests include natural language processing, machine translation, and Chinese language processing.