如何用 LaTeX 写语言学论文 v1.0

程航

Leiden University Centre for Linguistics

30th July 2020

Contents

1 引言 **3**

1.1 LaTeX 写作的优势与劣势 3

1.2 语言学论文与 LaTeX 4

1.3 参考资料 4

2 符号 **5**

2.1 IPA 5

2.1.1 宏包 Packages 5

2.1.2 命令 Commands 5

2.1.3 辅音表 6

2.1.4 元音舌位图 7

2.2 拼音 9

2.2.1 宏包 9

2.2.2 命令 9

2.2.3 自定义新命令 10

2.3 语义符号 10

3 例句 **12**
3.1 举例 12
3.1.1 宏包 12
3.1.2 命令 12
3.1.3 脚注中的例子 14
3.2 注释 15
3.2.1 命令 15
3.2.2 New Leipzig 16
3.2.3 Glossaries 16
3.3 交叉引用 17

4 句法树 **17**
4.1 基本操作 18
4.2 Marking nodes 21
4.3 Movement 22
4.4 Domain 23

5 写在最后 **25**

References **26**

Don't panic!

1 引言

本文内容只局限于传统的理论语言学和应用语言学论文写作。与自然语言处理相关的论文因笔者不甚了解，故不作讨论。

1.1 LaTeX 写作的优势与劣势

LaTeX写作遵循内容与形式分离的理念，所有格式设置都依靠输入相应的指令完成。这使得使用 LaTeX 写作有很多 Word 无法实现的优势。

首先，纯文本写作让作者可以专注于内容生产本身。一方面，写作过程中不必操心呈现效果，只要自己秉持内容与形式分离的观念，就可以实现先写再排。因为使用 LaTeX 编辑时不编译就不会看到文本效果，也就不会被简单粗糙的初步排版效果丑到自己。在使用 word 这类所写即所见、即时渲染的富格式编辑器时很难不被眼见的格式干扰。另一方面，因为所有的操作都通过输入指令完成，作者可以忠于键盘，写作过程中完全不需要在键盘和鼠标间切换，可以使写作流程更顺畅。

其次，因为所有设置都依靠输入指令完成，使用 LaTeX 排版则有更高的自由度和可控性，而且设置更加精细。此外，我们可以将常用指令汇总，实现更好的模板化写作。从个体写作的角度来看，大部分设置是一劳永逸的，一次设置成功之后，以后可以直接复制粘贴相同的指令（或微调）即可，不必每次都从头来过。从合作的角度来说，所有编辑器读取.tex 文件后编译出的文档显示效果都是完全相同的，避免了因文档编辑器版本差异带来的显示问题。从投稿的角度来说，如果期刊提供了 LaTeX 模板，我们也只需引用模板微调内容即可，不必反复手动调整。

第三，LaTeX 写作社区很成熟。一方面，新的宏包在不断出现，命令使用也相应越来越简单，操作可能性也越来越丰富。另一方面，大多数需求都很容易在网上找到解答，只要将相关命令复制粘贴到自己文本中即可解决问题。

当然，因为所有设置都必须通过输入指令完成，宏包纷繁复杂，所以使用 LaTeX 学习成本较高，而且有些在如 Word 上很容易的操作在 LaTeX 中非常繁琐。因此，在选择编辑器的时候需要仔细斟酌自己的需求，根据不同场景选择最佳方案。

1.2 语言学论文与 LaTeX

总体来说，LaTeX 在编辑体量较大的文档时优势明显，操作过程鲜有卡顿。写博士论文和专著可以考虑使用 LaTeX。

内容上看，LaTeX 对特殊符号、公式、交叉引用、结构图更加友好，操作便捷、显示稳定；但图片和表格编辑在 LaTeX 中比较繁琐。因此，涉及特殊符号和公式的语义学论文首推使用 LaTeX。涉及较多交叉引用和特殊格式（特别是 glossing）的语法学论文或参考语法的写作，使用 LaTeX 也会使写作过程轻松不少。句法树、音节结构、音系推导等 LaTeX 和 Word 各有优劣，笔者更倾向于使用 LaTeX，但大家可以结合其他需求酌情选择。最后，如果文章涉及较多图片和表格，但几乎没有其他特殊符号和特殊格式，那么可能使用 Word 会更加方便。

此外，因为 LaTeX 最终会生成 pdf 文档，如需与不使用 TeX 的同事合作，或投稿到只接收 Word 文档的期刊，请慎重选择。目前来看，格式简单的 pdf 转成.docx 文档非常方便，但格式复杂的 pfd 转成.docx 后可能需要大量重调，比较麻烦。

1.3 参考资料

LaTeX 基本操作参考资料较多，本文不再赘述。本文只介绍与语言学论文写作直接相关的内容，特别介绍写作中涉及汉语和拼音等内容。

LaTeX 入门可参考Datta (2017) 的 *LaTeX in 24 Hours: A Practical Guide for Scientific Writing* 和Kottwitz (2015) 的 *LaTeX Cookbook* 等基础教程。中文版教程知乎上多推荐刘海洋 (2013) 的 «LaTeX 入门 », 有兴趣的读者还可以参考用户李阿玲的专栏文章。

专门针对写语言学论文教程可以参考 Adam Liter 的LaTeX workshop for Linguistics, 以及 Sebastian Nordhoff 和 Antonio Machicao y Priemer 于 2019 年 LOT winter school 相关课程的讲义。

在了解 LaTeX 基本操作之后，实际写作过程中我们一定还会不断遇到各种问题，需要我们善用搜索引擎，大多数问题都可以找到答案。一般问题可以参考Overleaf documentation；与某些具体功能相关的问题可以参考相关宏包的 documentation，笔者建议在使用每个宏包前都阅读一下 documentation。更为细节的问题可以在stackexchange社区上搜索以往问题或直接提问，提问前请务必了解社区提问礼仪。

2 符号

2.1 IPA

2.1.1 宏包 Packages

大多数情况使用 tipa宏包即可。[1]

```
\usepackage{tipa}
```

如果 tipa 中的符号没有你需要的，可以增加一个 tipx 包。

```
\usepackage{tipa}
\usepackage{tipx}
```

官方指南中也提到，如果对声调符号 (tone letters) 有特殊需求，可以使用更有针对性的宏包选项。

```
\usepackage[tone]{tipa}
```

注意：tipa 和 fontspec 存在兼容性问题，使用 tipa 的时候需要注意字体的设置。

2.1.2 命令 Commands

有三种输入方式。[2]

第一种是直接输入音标对应符号名称 (corresponding macro name)。

- *Iutput 1:* `[\textsecstress\textepsilon kspl\textschwa\textprimstress ne\textsci\textesh\textschwa n]`
- *Output 1:* [ˌɛksplə'neɪʃən]

第二种是直接输入音标对应的符号缩写（shortcut characters for symbols）。

- *Input 2:* `\textipa{[""Ekspl@"neIS@n]}`
- *Output 2:* [ˌɛksplə'neɪʃən]

第三种是创建 IPA 环境。

[1] 点击蓝色宏包名可直接跳转对应的宏包主页。下同。

[2] 以下示例均引自tipa documentation。

- *Input 3:*

```
\begin{IPA}
    [""Ekspl@"neIS@n]
\end{IPA}
```

- *Output 3:* [ˌɛksplə'neɪʃən]

上述三种方式中官方最推荐的是第二种，因其最为方便、美观。此外，说明中还特别建议在使用`\textipa{}`命令时将方括号`[]`放在花括号`{}`内，呈现结果会更美观。对比效果如下：

- `\textipa{[""Ekspl@"neIS@n]}`: [ˌɛksplə'neɪʃən]
- `[\textipa{""Ekspl@"neIS@n}]`: [ˌɛksplə'neɪʃən]

2.1.3 辅音表

绘制辅音表可以借助https://www.tablesgenerator.com生成表格，在自动生成的表格基础上再做格式调整。LaTeX 中制作表格一定要仔细，仔细，仔细……

表格可能会遇到两个问题：一是太宽溢出页面，二是太长需要延续到后页。[3]

表格太宽其实没有很好的解决方案。一种思路是直接减少列数避免表格过宽，如 12*10 的表格手动变成 6*20 的表格。另一种思路是压缩已有表格的字号和边距，可以通过在`\begin{tabular}`前增加`\resizebox{\textwidth}{!}{}`实现自动调整，也可以使用`\setlength{\tabcolsep}{npt}`命令手动调整。默认格式中的 n=6，我们可以根据实际情况自己调整数字至合适的宽度。

表格太长的话可以使用 package longtable。

`\usepackage{longtable}`

Longtable 的使用需要注意设置表头在每页开始前显示：

`\endfirsthead`前输入表格正式的表头
`\endhead`前输入每次新起一页时出现的表头

其他格式上的调整，如标题与表格间的距离、标题或表头与页边的距离等等，请视实际情况参考longtable documentation进行设置。

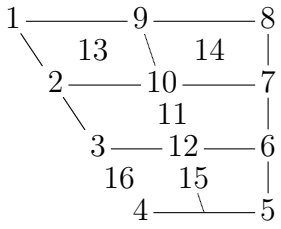
[3]本节内容感谢吴疆同学帮助。

2.1.4 元音舌位图

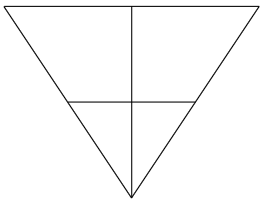
通过使用vowel宏包可以直接绘制元音舌位图。

```
\usepackage{vowel}
```

不加说明的情况下，`\begin{vowel}...\end{vowel}`环境生成四边形结构。



制作三角时,需特别说明`\begin{vowel}[triangle,three]...\end{vowel}`环境。



绘制元音图的基本命令为：`\putcvowel[l/r]{x}{y}`。其中：

- y 处输入图示中的具体位置 (上图中的数字)
- l/r 代表在在这个位置的左侧还是右侧（即不圆唇 vs. 圆唇）
- x 处输入具体的元音符号

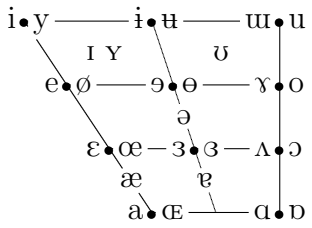
下面展示一个例子 (引自Nordhoff and Machicao y Priemer (2019))：

Input:

```
\begin{vowel}
\putcvowel[l]{\textipa{i}}{1}
\putcvowel[r]{\textipa{y}}{1}
\putcvowel[l]{e}{2}
```

```
\putcvowel[r]{\o}{2}
\putcvowel[l]{\textepsilon}{3}
\putcvowel[r]{\oe}{3}
\putcvowel[l]{a}{4}
\putcvowel[r]{\textscoelig}{4}
\putcvowel[l]{\textscripta}{5}
\putcvowel[r]{\textturnscripta}{5}
\putcvowel[l]{\textturnv}{6}
\putcvowel[r]{\textopeno}{6}
\putcvowel[l]{\textramshorns}{7}
\putcvowel[r]{o}{7}
\putcvowel[l]{\textturnm}{8}
\putcvowel[r]{u}{8}
\putcvowel[l]{\textbari}{9}
\putcvowel[r]{\textbaru}{9}
\putcvowel[l]{\textreve}{10}
\putcvowel[r]{\textbaro}{10}
\putcvowel{\textschwa}{11}
\putcvowel[l]{\textrevepsilon}{12}
\putcvowel[r]{\textcloserevepsilon}{12}
\putcvowel{\textsci\ \textscy}{13}
\putcvowel{\textupsilon}{14}
\putcvowel{\textturna}{15}
\putcvowel{\ae}{16}
\end{vowel}
```

Output:



2.2 拼音

2.2.1 宏包

标注拼音是写与汉语相关论文比较常见的需求，LaTeX 中输入拼音非常方便，借助xpinyin宏包可以通过输入数字来区别声调，也可以直接在汉字上标注拼音。注意，使用 xpinyin 宏包时推荐使用 XeLaTex 编译，关联xeCJK宏包，不要使用 pdfLaTeX 编译。

```
\usepackage{xpinyin}
```

2.2.2 命令

行文中直接输入拼音的基本命令是`\pinyin{ma1}` 'mā'。键入的拼音可以出现在正文中，可以出现在例句标注中，也可以出现在句法树中。

注意：

- lǘ 输入`\pinyin{lv2}`
- 轻声可以不标数字，也可以标 0
- 默认格式是每个音节后都有一个空格,我们可以通过增加空格指令`\pysep={}`实现分词：
  - `\pinyin{xing1qi1} \pinyin{yi1}` 'xīng qī yī'
  - `\pinyin[pysep={}]{xing1qi1} \pinyin[pysep={}]{yi1}`: 'xīngqī yī'
- 拼音可以加粗或斜体。
  - `\pinyin[format={\it}]{ma1}`: *mā*
  - `\pinyin[format={\bf}]{ma1}` **mā**

  注:笔者目前没能实现加粗且斜体, 命令`\pinyin[format={\bf{it}}]{ma1}`或`\pinyin[format={\bf\it}]{ma1}`均无效。

xpinyin 宏包另一个重要功能是为汉字标拼音, 有两个基本命令可以使用，显示效果一样。拼音格式可以根据需要调整，具体设置命令请参考xpinyin documentation。

- `\xpinyin{妈}{ma1}`: 妈(mā)
- `\xpinyin*{妈}`: 妈(mā)

2.2.3 自定义新命令

因每次要输入\pinyin[pysep={}]{}会比较繁琐，即使是复制粘贴也比较麻烦。我们可以直接在 preamble 设置格式，这样就可以省去[pysep={}]。

```
\usepackage{xpinyin}
\xpinyinsetup{pysep={}}
```

同样的，加粗和斜体的命令也比较复杂；而且\pinyin本身要输入的字符也较多，我们都可以直接通过设置\newcommand来减少麻烦。

- 如果使用独立的 preamble 文件，可以在直接在\usepackage{xpinyin}附近增加下面的设置。

```
\AtBeginDocument{
\newcommand{\p}[1]{\pinyin{#1}}
\newcommand{\ip}[1]{\pinyin[format={\it}]{#1}}
\newcommand{\bp}[1]{\pinyin[format={\bf}]{#1}}
}
```

- 如果 preamble 直接放在\begin{document}之前,可以省去\AtBeginDocument{}这条。
- 效果演示：
  - \p{ma1}: mā
  - \ip{ma1}: *mā*
  - \bp{ma1}: **mā**
- Newcommand 的语法解释:\newcommand{cmd}[args]{def}
  - cmd 是新使用的命令，本例中笔者使用 p 代表 pinyin,ip 代表斜体拼音，bp 代表粗体拼音。大家可以根据自己的喜好设置别的缩写。
  - args 说明论元个数，在 pinyin 的命令里论元只有一个，输入 1 即可。
  - def 代表被定义的原命令。

2.3 语义符号

LaTeX 提供了丰富的数学符号、逻辑符号、箭头、希腊字母等等，极大方便了我们输入语义符号。下面陈列部分常见符号。

- `$\neg$`: $\neg$
- `$\equiv$`: $\equiv$
- `$\in$`: $\in$
- `$\notin$`: $\notin$
- `$\cup$`: $\cup$
- `$\cap$`: $\cap$
- `$\subset$`: $\subset$
- `$\supset$`: $\supset$
- `$\exists$`: $\exists$
- `$\forall$`: $\forall$
- `$\alpha$`: α
- `$\beta$`: β
- `$\lambda$`: λ
- `$\tau$`: τ
- `$\Box$`: $\Box$
- `$\Diamond$`: $\Diamond$
- `$\leadsto$`: $\leadsto$
- `$\rightarrow$`: $\rightarrow$
- `$\Rightarrow$`: $\Rightarrow$
- `$\widetilde{abc}$`: $\widetilde{abc}$
- `$\overrightarrow{abc}$`: $\overrightarrow{abc}$

笔者作为语义小白，日常写作中较少涉及语义式。下面参考Nordhoff and Machicao y Priemer (2019) 简单举几个例子, 并推荐一个教程和一个网站LaTeX for logicians，更多操作烦请语义学大神指点。

- 集合论 Set Theory
  - `$\emptyset \subseteq \{\textrm{a,b}\}$`
  - $\emptyset \subseteq \{\textrm{a,b}\}$
- 命题逻辑 Propositional Logic
  - `$\lnot (P \lor Q ) \Leftrightarrow (\lnot P \wedge \lnot Q)$`
  - $\neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q)$
- 量词 Quantifiers
  - `$\exists x [$\textsc{woman}$(x)$ $\land$ \textsc{sleep}$(x)]$`
  - $\exists x[\textsc{woman}(x) \wedge \textsc{sleep}(x)]$
- Functional Application [4]
  - `$\lsem \alpha \beta \rsem = \lsem \beta \rsem (\lsem \alpha \rsem)$`
  - $[\![\alpha\beta]\!] = [\![\beta]\!]([\![\alpha]\!])$

[4]注意：meaning bracket 需要MnSymbol宏包。

3 例句

3.1 举例

3.1.1 宏包

通常使用gb4e, 注意配合命令`\noautomath`。

```
\usepackage{gb4e}
\noautomath
```

除 gb4e 以外，还可以尝试linguex或ling-macros。

3.1.2 命令

gb4e 宏包的基本指令如下：

```
\begin{exe}
\ex 今天星期一。
\ex
    \begin{xlist}
        \ex [*]{今天不星期一。}
        \ex [\#]{今天星期八。。}
    \end{xlist}
\end{exe}
```

- 效果如下：

(1) 今天星期一。

(2) a. * 今天不星期一。
b. # 今天星期八。

- 命令说明：
  - `[]`内`*`或`#`标注句法判断结果，例子需用`{}`包住。
  - 合法的例子（或无`[]`标注判断结果的例句)无需`{}`，用`{}`也不会报错。
  - `\begin{xlist}...\end{xlist}`命令可以多层嵌套。
  - xlist 的列表符号可以更换
    - `\begin{xlista}...\end{xlista}`: a. alphabetical (默认)
    - `\begin{xlisti}...\end{xlisti}`: i. roman
    - `\begin{xlistn}...\end{xlistn}`: 1. arabic

    - \begin{xlistI}...\end{xlistI}: I. Roman
    - \begin{xlistA}...\end{xlistA}: A. Alphabetical 。

下面四项命令用于调整例句编号：

- \exi: 自定义编号
  - 命令格式为\exi{自定义编号}[判断]{例句}
  - 自定义编号可以直接输入任意编号，如 (3) 或 (α) 等
  - 也可以通过交叉引用命令\ref{}指向某个例句。通过交叉引用自定义编号可以随被引例句编号的变动而改变。
- \exr：重复编号
  - 命令格式为\exr{引用标签}[判断]{例句}。
- \exp：编号加’
  - 命令格式为\exp{引用标签}[判断]{例句}。
  - 笔者该命令使用一直失败，原因不明。
- \sn：不参加编号，如下例 “昨天星期日”。

下面演示了上述指令的使用。注意，我们需要通过\label{}对被引用的例子加标签（3.3节 “交叉引用” 将会说明\label{}的使用）。

Input:

```
\begin{exe}
    \ex \label{test1}
    今天星期一。
    \sn 昨天星期日。
    \ex \label{test2}
    明天星期二。
    \exi{(3)} 今天星期一。
    \exi{($\alpha$)} 后天星期三。
    \exi {(\ref{test1})} 今天星期一。
    \exr{test2} 明天星期二。
\end{exe}
```

Output:

(3) 今天星期一。
昨天星期日。
(4) 明天星期二。
(3) 今天星期一。
(α) 后天星期三。
(3) 今天星期一。
(4) 明天星期二。

右对齐的说明性文字可以通过`\hfill`命令来添加。

Input:

```
\begin{exe}
    \ex 他二十岁。 \hfill (Zhu 1982: 103)
    \ex 他不是二十岁。

    \hfill \textit{negation}
\end{exe}
```

Output:

(5) 他二十岁。 (Zhu 1982: 103)
(6) 他不是二十岁。
negation

3.1.3 脚注中的例子

默认状态下，脚注中例句的编号会接上正文中，但实际写作中我们需要将脚注中的例句单独编号小写罗马数字。我们只需要在 preamble 中添加如下指令即可。[5]

[5]脚注例句编号示例：

(i) 这是一条脚注。
(ii) a. 这是另一条脚注。
b. 其实还有一条脚注。

```
\makeatletter
\pretocmd{\@footnotetext}{
\@noftnotefalse\setcounter{fnx}{0}%
\renewcommand{\thexnumi}{\roman{xnumi}}
}{}{}
\apptocmd{\@footnotetext}{
\@noftnotetrue
\renewcommand{\thexnumi}{\arabic{xnumi}}
}{}{}
```

3.2 注释

做语料标注建议使用leipzig宏包。Leipzig 宏包可以帮助我们自动生成 small capital 格式的 gloss，也可以在脚注或文档最后生成 glossaries。

```
\usepackage[mcolblock]{leipzig}
```

3.2.1 命令

给例句进行标注的基本命令如下：

Input: [6]

```
\begin{exe}
    \ex 他是一个好人。\\
    \gll \ip{Ta1} \ip{shi4} \ip{yi1}-\ip{ge} \ip{hao3}
     ↪  \ip{ren2}.\\
    \Tsg{} \Cop{} one-\Clf{} good person\\
    \glt `He is a good person.'
\end{exe}
```

Output:

(7) 他是一个好人。
Tā shì yī-ge hǎo rén.
3SG COP one-CLF good person
'He is a good person.'

使用说明：

[6]本段中出现的箭头是本文档编辑中自动生成的换行符，在实际论文写作中不需要，也不会出现。下同。

- \gll起是正式的 gloss 部分,\gll代表有两行内容需要根据空格对齐,\glll用于有三行内容需要对齐的情况。
- 每行末尾用\\隔开。
- 最后一行翻译可以用\glt也可以用\trans
- 默认状态下，逐词标注与翻译行间会有一个较大空格，可以使用cgloss宏包来解决这个问题。
  - \usepackage{cgloss}
  - cgloss 不是 TeX 默认安装的宏包，需要自己安装。也可以直接在 preamble 使用\input{cgloss.sty}来调用宏包软件。
  - 使用 cgloss 之后，\gll前的文字也需要使用\\进行分行。

3.2.2 New Leipzig

Leipzig documentation文档末尾提供了该宏包默认的标注。除此以外，我们还可以自定义命令。

自定义基本格式如下：\newleipzig{label}{short}{long}

- label 指在写标注时输入的内容，如下例中我们录入\Final{}。
- short 指标注中显示出的内容，如下例中会显示出 SFP。
- label 和 short 可以是一样的字符串。
- long 指 gloss 的完整内容，可以在 Glossaries 中出现，如下例中 sentence final particle。
- 例：\newleipzig{final}{sfp}{sentence final particle}

3.2.3 Glossaries

使用 Leipzig 宏包后，我们可以在文档末尾或脚注中自动生成 glossary，总共涉及两个命令。首先，在 preamble 里需要输入\makeglossaries; 在需要显示 glossary 的地方使用\printglossaries。

注意,Leipzig 有两类指令,一类是 glossary（\makeglossary和\printglossary），另一类是 glossaries（\makeglossaries和\printglossaries）。使用前者最终只显示 Leipzig 默认的 glossary，使用后者则可以显示所有。

3.3 交叉引用

交叉引用主要涉及到两个命令。一个是\label{}，用来给被引用的内容做上标记。标签名称可以按自己喜好设置。被引的内容可以是一个例子，也可以是一个章节。另一个命令是\ref{}，指向被引内容。

Input:

```
\subsection{交叉引用}
\label{sec:crossref}

例 (\ref{1}) 表达了张三使用 LaTeX 写例句的感受。交叉引用的具体使用方
↪ 法参考第\pageref{sec:crossref} 页第\ref{sec:crossref} 节的介绍。

    \begin{exe}
        \ex \label{1}
        交叉引用特别好用！
    \end{exe}
```

Output:

例 (8) 表达了张三使用 LaTeX 写例句的感受。交叉引用的具体使用方法参考第17页第3.3节的介绍。

(8) 交叉引用特别好用！

4 句法树

绘制句法树可以使用forest宏包或者qtree宏包。因笔者日常更多使用 forest，本文主要介绍 forest 的使用。

配置 forest 需要注意两个问题。第一，forest 部分命令的运行依赖 gb4e，因此，如果同时使用 gb4e 宏包的话，在 preamble 里写\usepackage{}命令必须注意运行顺序，forest 在前 gb4e 在后。

```
\usepackage{forest}
\usepackage{gb4e}
```

第二，forest 并不是专门针对句法树，因此我们在使用时需要注明语言学设置：

```
\usepackage[linguistics]{forest}
```

Linguistics 和非 linguistics 模式下生成的树形结构在节点的呈现方式上有区别，如下图所示（图片引自Nordhoff and Machicao y Priemer (2019)）。左侧是非 Linguistics 模式下的效果，右侧是 Linguistics 模式下的效果。

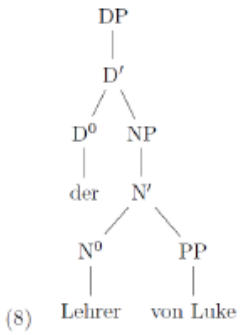


Fig. 1: without `linguistics`

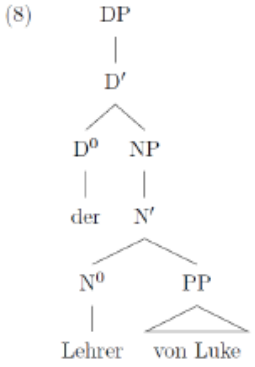
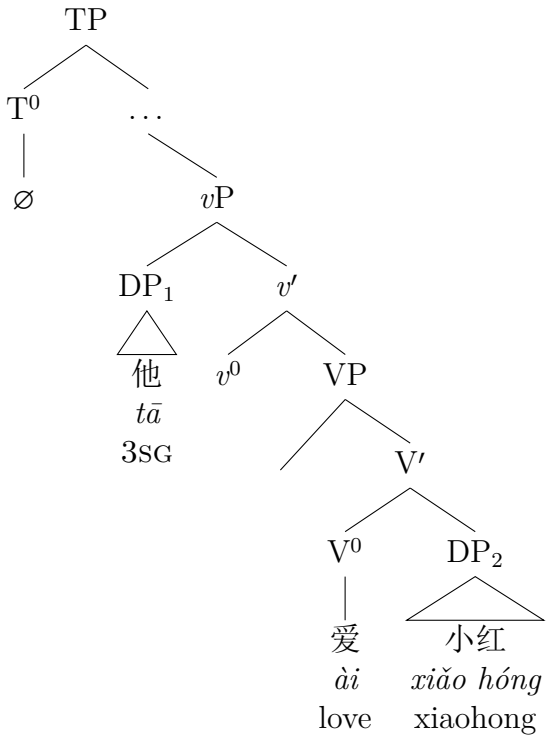


Fig. 2: with `linguistics`

4.1 基本操作

画树时需要通过`\begin{forest}...\end{forest}`创建 forest 环境。通过括号结构 (bracket notation) 来呈现内容。不过，建议通过层级结构和缩进来减轻自己写 bracket structure 的负担。

下面展示一棵简单的树。



使用到的命令如下：

```
\begin{forest}
    [TP
        [T$^{0}$ [$\emptyset$]]
        [$\dots$
            [,phantom]
            [\textit{v}P
                [DP$_1$ [他\\\ip{ta1}\\ \Tsg{}, roof]]
                [\textit{v}$^{\prime}$
                    [\textit{V}$^{0}$]
                    [VP
                        []
                        [V$^{\prime}$
                            [V$^{0}$ [爱\\\ip{ai4}\\love]]
                            [DP$_2$ [小
                             ↪ 红\\\ip{xiao3hong2}\\xiaohong,
                             ↪ roof]]
                        ]
                    ]
```

```
                ]
            ]
        ]
    ]
\end{forest}
```

使用说明：

- 汉字和拼音都可以出现在树里
  - 注意：不能出现关于拼音的格式设置，如`[pysep={}]`,`[format=\it]`等。如果希望显示不同格式，只能在 preamble 里设置好。
- 一个节点下可以出现多行内容，通过`\\`隔开
- 三角框用`,roof`标注
- 上下标分别是`_x`和`^x`
  - 最好使用数学环境`$...$`
  - 如果上下标多于一个字符，可以用花括号括起来，如`N$_{1a}$`, 显示效果为 N_{1a}。
- 强调 phonetically null 的时候可以用`$\emptyset$`打出 $\varnothing$ 符号
- 省略的内容可以通过`\ldots`作 ... 或直接打...
- 注意`[,phantom]`和`[]`的区别。`[,phantom]`也可以通过`{}`实现。

默认效果下每一级两个节点间的距离会随着内容的多少而改变，可以通过设置 nice/fine nodes 等效果使树形结构更平衡、美观。笔者在 preamble 使用的设置如下所示，更多设置可以参考 Stack Exchange 中的回答）以及forest documentation。

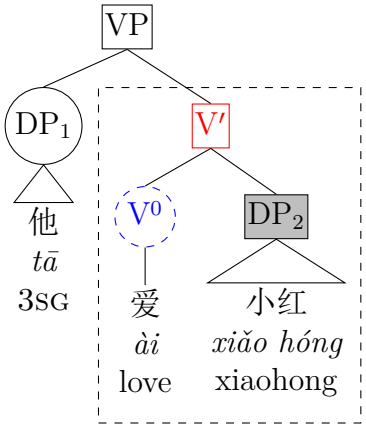
```
\AtBeginDocument{
\forestset{
    nice nodes/.style={
        for tree={
            inner sep=1pt, s sep=12pt,
            fit=band,
        },
    },
    default preamble=nice nodes,
}
}
```

4.2 Marking nodes

如果需要强调某些节点，可以加上边框或改变颜色等等。常用命令如下例：

- `draw`: 画方框
- `circle,draw`: 画圈
- `red,draw`: 红色方框
- `dashed, circle, draw`: 虚线圆圈
- `fill=blue`: 涂蓝色
- `tikz={\node [draw, fit to=tree] {};}`: 大框

我们对上一节中的树做一些改动，可以看到相应的标记效果。



命令如下：

```
\begin{forest}
    [VP, draw
        [DP$_1$, circle, draw
            [他\\\ip{ta1}\\ \Tsg{}, roof]]
        [V$^{\prime}$, red, draw, tikz={\node [dashed,
        ↪ draw,inner sep=6pt,fit to=tree] {};}
            [V$^{0}$, dashed, blue, circle, draw
                [爱\\\ip{ai4}\\love]]
            [DP$_2$, draw, fill=lightgray
                [小红\\\ip{xiao3hong2}\\xiaohong, roof]]
        ]
    ]
\end{forest}
```

本文显示的框式效果都是默认格式下的，框的大小、和节点标签的位置等等格式可以进一步调整，具体操作请参照forest documentation。

4.3 Movement

绘制 movement 轨迹使用\draw命令即可，但是需要给起始点和终点做好标记，并写明路径方向和线条类型。

命令格式：`\draw[X] (scr) to [out=Y, in=Z] (tgt);`

例如：`\draw[->] (N1) to [out=south west, in=south] (N2);`

说明：

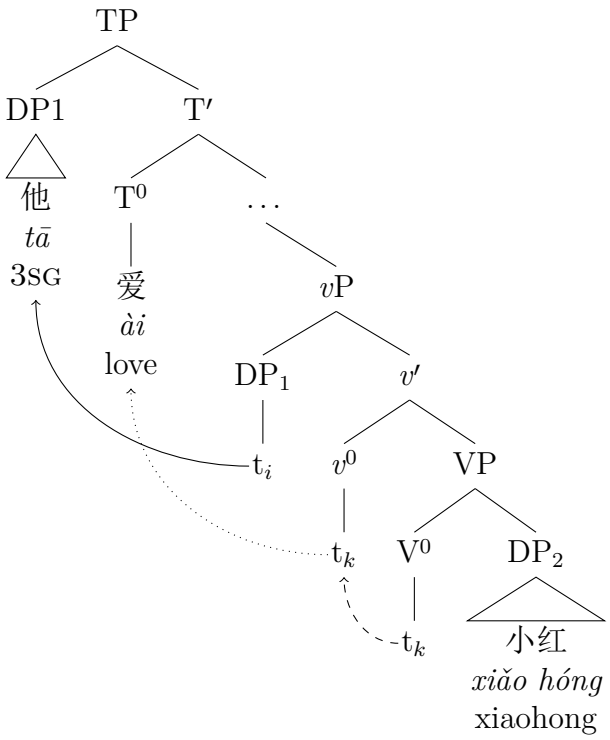
- X 写明线条类型可以是->, 可以是[->,dashed]
- scr 写明起始点的标签 (命令：name=)
- Y 写明线条出发的方向
- Z 写明线条到达的方向
- tgt 写明终点的标签

我们用之前的例子来演示一下效果。

```
\begin{forest}
    [TP
        [DP1 [他\\\ip{ta1}\\ \Tsg{}, roof, name=tgt1]]
        [T$^{\prime}$
            [T$^{0}$ [爱\\\ip{ai4}\\love, name=tgt2]]
            [$\dots$
                [,phantom]
                [\textit{v}P
                    [DP$_1$ [t$_i$, name=scr1]]
                    [\textit{v}$^{\prime}$
                        [\textit{v}$^{0}$ [t$_k$, name=mdl]]
                        [VP
                            [V$^{0}$ [t$_k$, name=scr2]]
                            [DP$_2$ [小
                            ↪ 红\\\ip{xiao3hong2}\\xiaohong,
                            ↪ roof]]
```

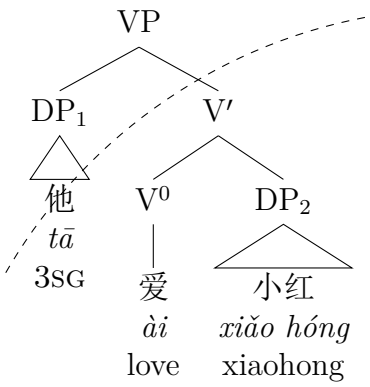
```
                        ]
                    ]
                ]
            ]
        ]
    ]
\draw [->] (scr1) to [out=west, in=south] (tgt1);
\draw [->,dashed] (scr2) to [out=west, in=south] (mdl);
\draw [->,dotted] (mdl) to [out=west, in=south] (tgt2);
\end{forest}
```

效果如下：



4.4 Domain

我们还可以通过绘制弧线来表明不同的 domain 或 phase，如下图所示。



涉及的命令如下：

```
\begin{forest}
    [VP
        [DP$_1$ [他\\\ip{ta1}\\ \Tsg{}, roof]]
        [V$^{\prime}$, name=v
            [V$^{0}$ [爱\\\ip{ai4}\\love]]
            [DP$_2$ [小红\\\ip{xiao3hong2}\\xiaohong, roof]]
        ]
    ]
    \draw[dashed] ([xshift=-80pt,yshift=-60pt]v) arc[start
     ↪ angle=150,end angle=100,radius=7cm, fit to=tree];
\end{forest}
```

本节内容参考 stack exchange 某个问题解答中的做法，各项参数代表的具体意义笔者暂时也不是很清楚，通过参考 arc 命令的意义，以及不断改变数据调试后大致有个猜想。

- 括号里的 v 对应相关节点的标签，如我在 V′ 后增加了 name=v 的标记。
- `[x,y]`大致是圆心相对于相关 node 的位置关系
- `[start angle, end angle]`大致是弧线起点和终点与横纵坐标轴的角度关系
- radius 标明半径

一个标准的 arc 命令结构如下：

```
\draw (x,y) arc (start:stop:radius);
```

其中：

- radius 标明半径
- 弧线起点为 (x,y)
- 弧线中心点为 (x-r*cos(start), y-r*sin(start))
- 弧线终点为 (x-r*cos(start)+r*cos(stop), y-r*sin(start)+r*sin(stop))
- 例：`\draw[red] (0,0) arc (30:60:3);`
  - 半径 3
  - 起点（0,0）
  - 中心点 (0+3*cos(30+180),0+3*sin(30+180))
  - 终点 (0+3*cos(30+180)+3*cos(60),0+3*sin(30+180)+3*sin(60))

(详情请参考：source)

此外,笔者暂时未解决在弧线旁加文字的问题。如`tikz={\node [dashed, draw] {};}`等命令中可以在`{}`中输入文字，即可显示。

5 写在最后

使用 LaTeX 写作虽然方便，但零碎的设置较多，长时间不用也很容易忘记。笔者以本文档汇总语言学论文写作中部分常用的命令，方便自己，希望也可以方便同行学者。

本人接触 LaTeX 写作时间也不长，了解也不深入，期待更多读者批评指正，提供经验，相互帮助。同时，考虑到每个人研究方向不同，常用的领域也不完全重合，笔者也创建了一个 overleaf 共享页面（链接），希望有兴趣的小伙伴可以留言、评论或增加内容。

联系方式：

email: chenghang0930@163.com

Website:

本文档下载地址：

本人 *preamble* 文档分享：

References

刘海洋. (2013). Latex 入门. 电子工业出版社, 北京.

Datta, D. (2017). *Latex in 24 hours: A practical guide for scientific writing.* Springer.

Kottwitz, S. (2015). *Latex cookbook.* Packt Publishing Ltd.

Nordhoff, S. & Machicao y Priemer, A. (2019). *Latex for linguists* [lecture handout]. LOT School in Amsterdam.