机器学习算法
与自然语言处理

# Paper-Picture-Writing-Code：零基础入门高质量论文绘图

李北

2022.11.26

东北大学计算机科学
与工程学院人工智能系
Department of Artificial Intelligence, School
of Computer Science and Engineering, NEU

# 项目简介



https://github.com/MLNLP-World/Paper-Picture-Writing-Code

**Paper-Picture-Writing-Code 是由MLNLP社区成员共同参与完成的科研画图项目，旨在帮助刚接触科研的同学快速入门，助力科研第一步！**

**社区的愿景是促进国内外自然语言处理，机器学习学术界、产业界和广大爱好者之间的交流和进步，特别是初学同学们的进步。**

# 项目成员



李北
东北大学

刘乾
SEA AI LAB

黄仕爵
哈尔滨工业大学（深圳）

张钰迪
哈尔滨工业大学

覃立波
哈尔滨工业大学

# 项目简介



论文绘图用　　工具

# 论文绘图用　　工具

了Python、PPT和Visio之外，也使用Latex绘　出精美的论文图。

**可绘**　：折线图、柱状图、散点图、注意力图、模型结构图、分类图、时间线

**特　点**：精美、　　、高　　量

Paper-Picture-Writing-Code → LaTeX

# 项目动机

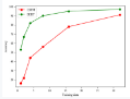
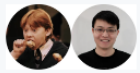
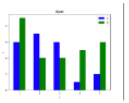
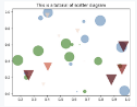
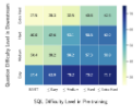


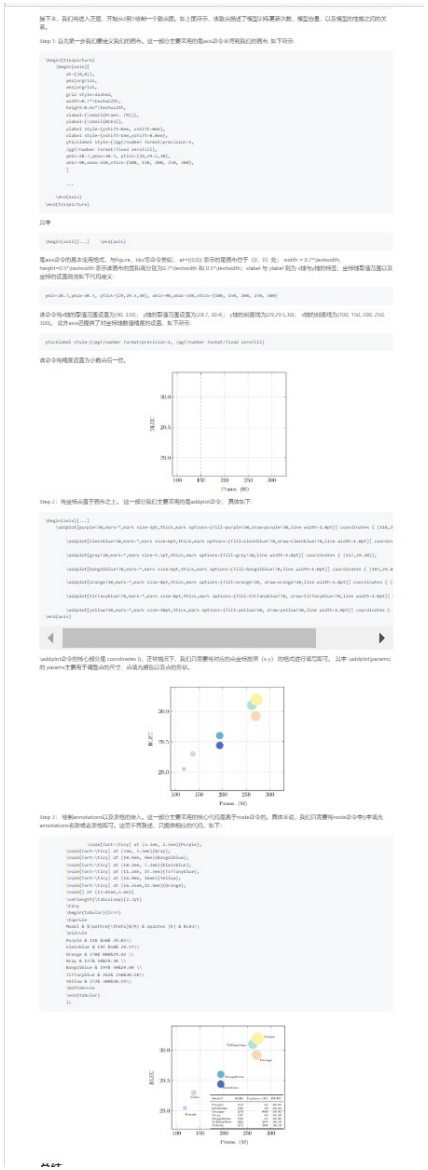
对于很多科研新手来说论文图片的绘制常常成为论文写作中一大难题，图像失真、图示不合规范等问题会导致论文质量的降低，LaTex原生支持的图包绘制的矢量图具有不会出现失真、便于实时修改和维护等优点，为了帮助一些同学提高论文写作画图的质量和效率、学会更好地利用**LaTex画图**和**Matplotlib画图**，本项目基于LaTex和matplotlib开源论文中常用的**画图代码**，希望能对大家的论文写作图片绘制有所帮助。

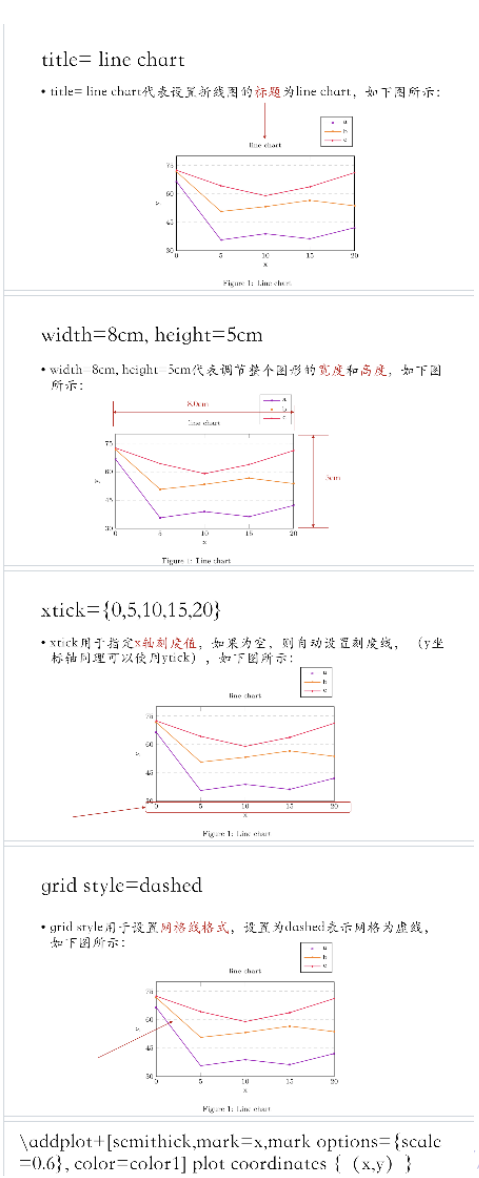
> 本项目的特色：
>
> 1. **代码开源**：为初学者提供了可学习和可复用的代码样例，方便大家快速入门。
> 2. **笔记详细**：帮助此项目基础上进一步上手实践，也便于快速定位自己代码中的错误。
> 3. **类别广繁**：涵盖AI论文画图中常用的几类画图代码。

# 项目特色



| 图类别 | Latex代码 | Latex笔记 | Matplotlib代码 | Matplotlib笔记 | 贡献者 |
|---|---|---|---|---|---|
| | 引言 | | | | |
|  | 折线图、带误差条的折线图 | 折线图、带误差条的折线图 | 折线图 | 折线图 |  |
|  | 柱状图 | 柱状图 | 柱状图 | 柱状图 | |
|  | 散点图 | 散点图 | 散点图 | 散点图 | |
|  | 注意力分布 | 注意力分布 | 注意力分布 | 注意力分布 | |
| | 注意力序列可视化 | 注意力序列可视化 | | | |
| | 结构图 | 结构图 | | | |
| | 分类图 | 分类图 | | | |
| | 时间线 | 时间线 | | | |





# 项目分类—折线图

机器学习算法与自然语言处理

| 图类别 | Latex代码 | Latex笔记 | Matplotlib代码 | Matplotlib笔记 | 贡献者 |
|---|---|---|---|---|---|
| 折线图 | 折线图<br>带误差条的折线图 | 折线图<br>带误差条的折线图 | 折线图 | 折线图 | |

Latex

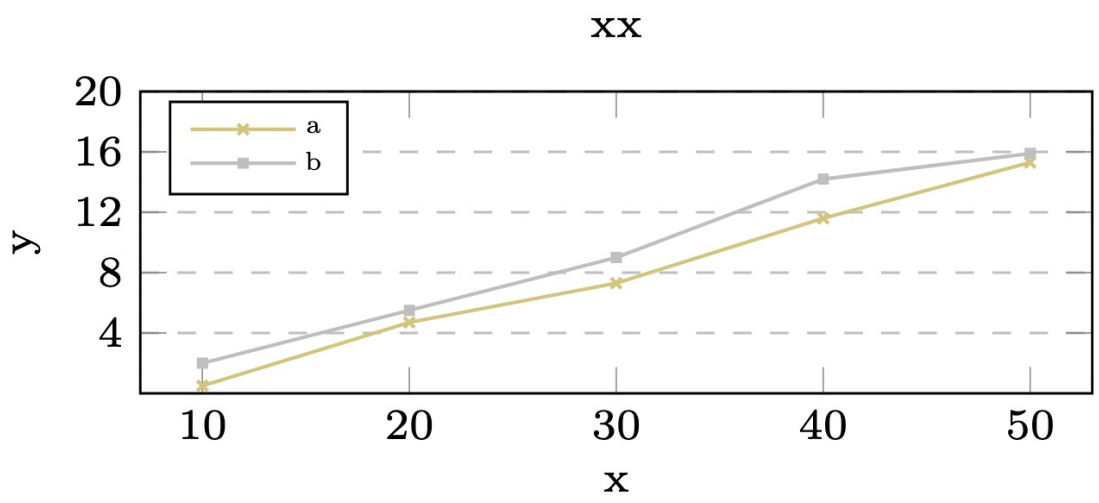
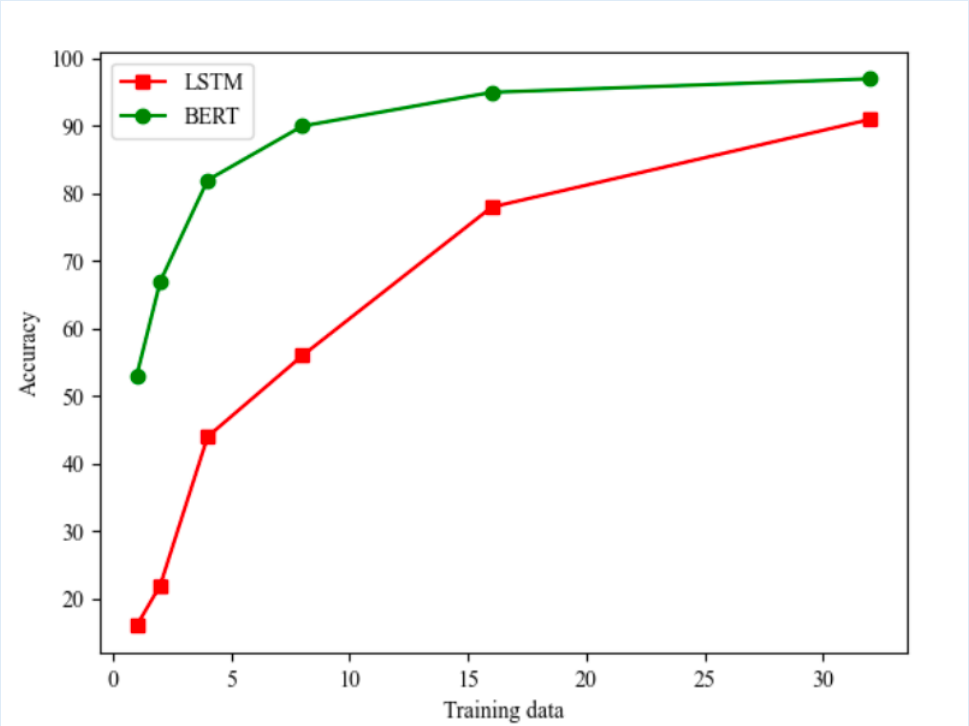



Figure 1: xxxx

Matplotlib

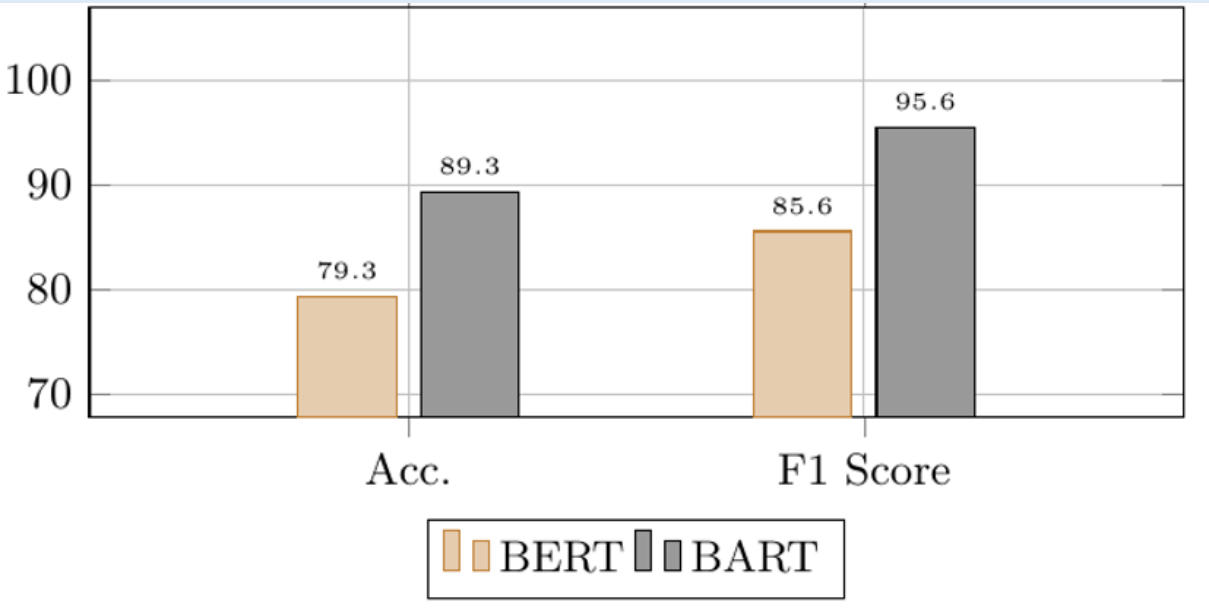


# 项目分类—柱状图

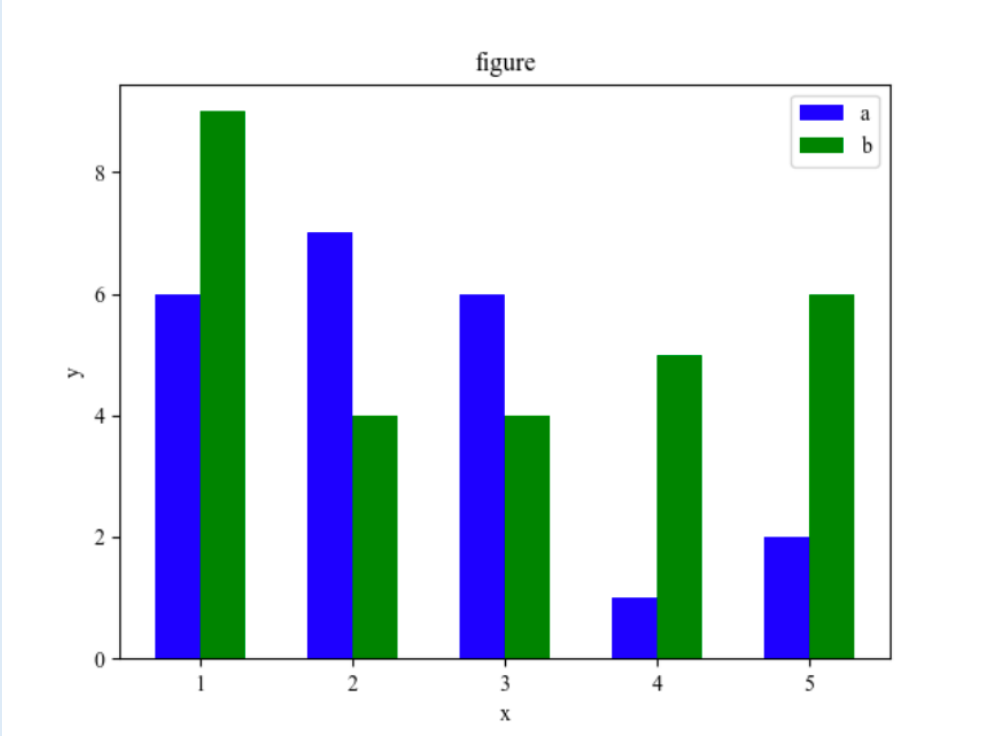
机器学习算法与自然语言处理

| 图类别 | Latex代码 | Latex笔记 | Matplotlib代码 | Matplotlib笔记 | 贡献者 |
|---|---|---|---|---|---|
| 柱状图 | 柱状图 | 柱状图 | 柱状图 | 柱状图 | |

Latex




Matplotlib

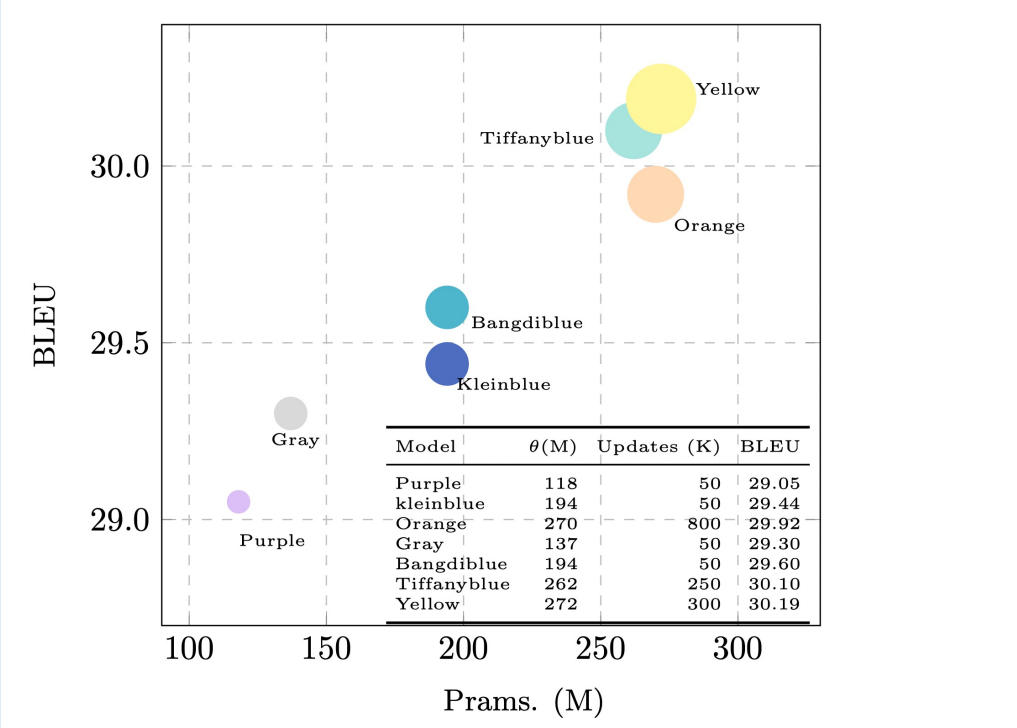


# 项目分类—散点图

机器学习算法与自然语言处理

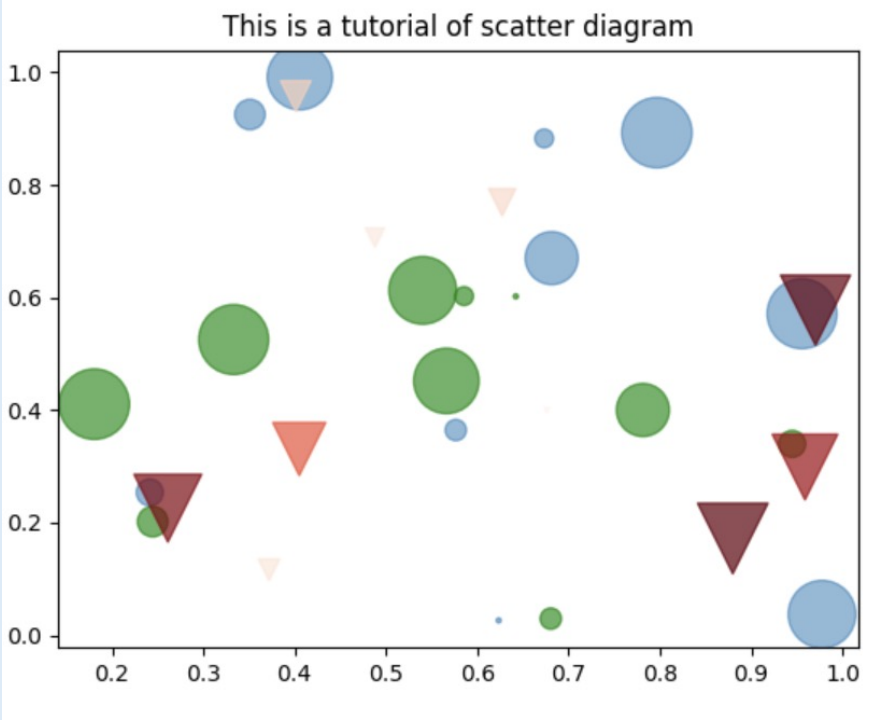
| 图类别 | Latex代码 | Latex笔记 | Matplotlib代码 | Matplotlib笔记 | 贡献者 |
|---|---|---|---|---|---|
| 散点图 | 散点图 | 散点图 | 散点图 | 散点图 | |

Latex




| Model | $\theta$(M) | Updates (K) | BLEU |
|---|---|---|---|
| Purple | 118 | 50 | 29.05 |
| kleinblue | 194 | 50 | 29.44 |
| Orange | 270 | 800 | 29.92 |
| Gray | 137 | 50 | 29.30 |
| Bangdiblue | 194 | 50 | 29.60 |
| Tiffanyblue | 262 | 250 | 30.10 |
| Yellow | 272 | 300 | 30.19 |

Matplotlib



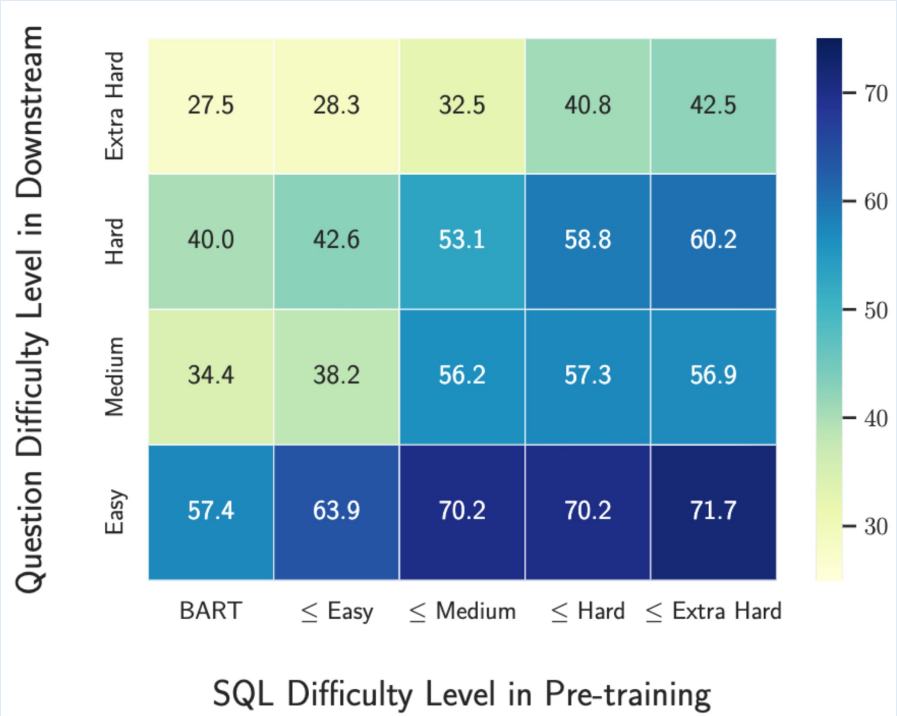
# 项目分类—注意力分布

机器学习算法与自然语言处理

| 图类别 | Latex代码 | Latex笔记 | Matplotlib代码 | Matplotlib笔记 | 贡献者 |
|---|---|---|---|---|---|
| 注意力分布 | 注意力分布 | 注意力分布 | 注意力分布 | 注意力分布 | |

Latex

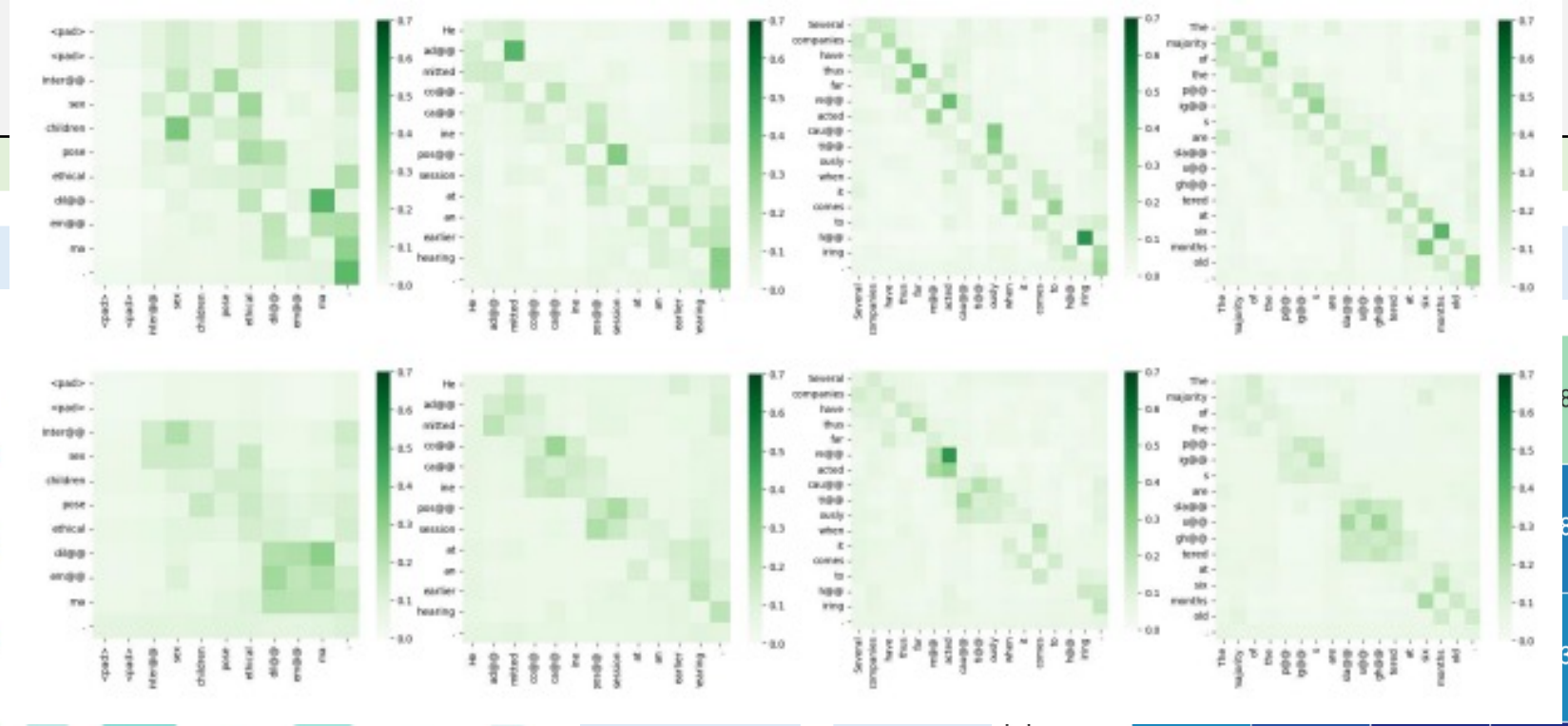
Matplotlib

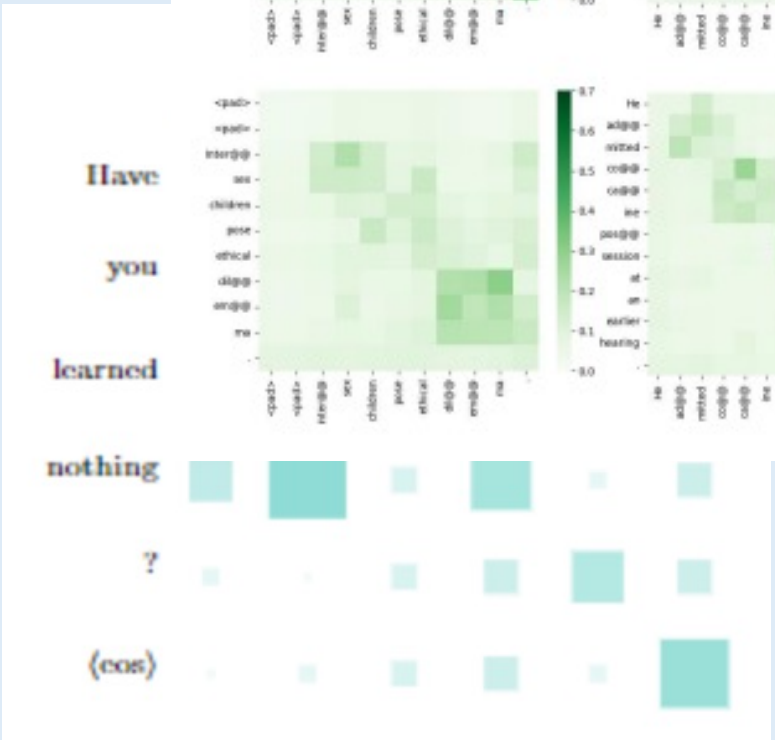


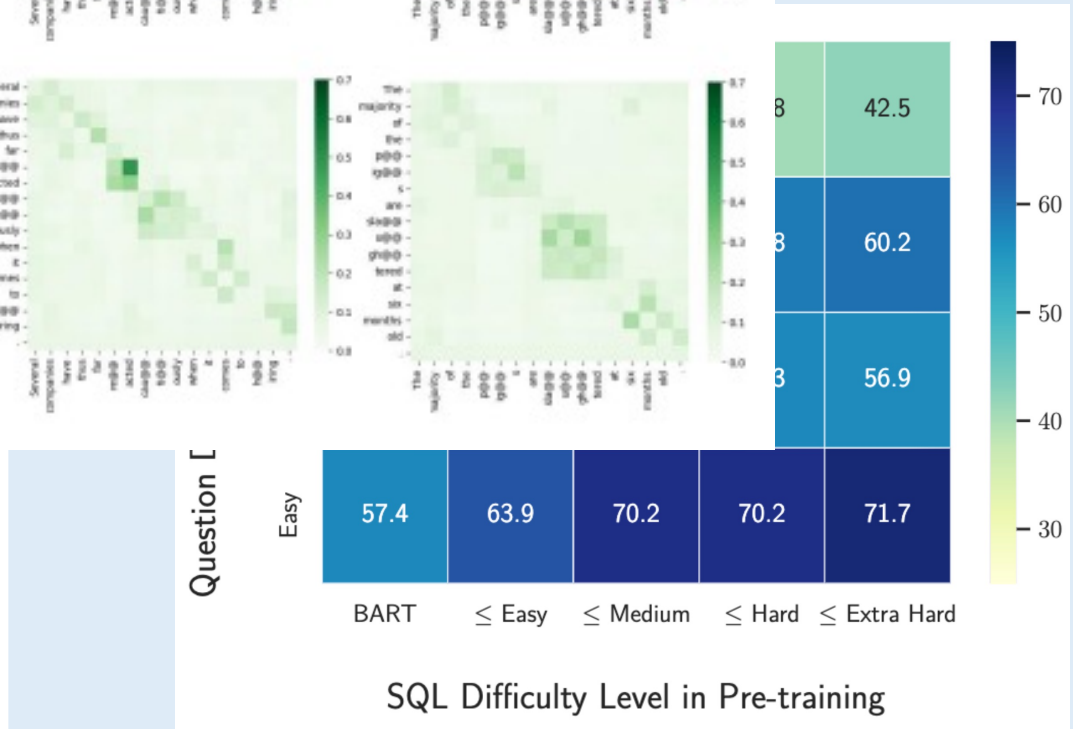
# 项目分类—注意力分布

机器学习算法与自然语言处理

| 图类别 | Latex代码 | Latex笔记 | Matplotlib代码 | Matplotlib笔记 | 贡献者 |
|---|---|---|---|---|---|
| 注意力分布 | | | | | |





# 项目分类—注意力分布



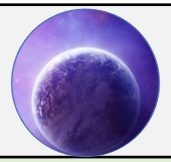
| 图类别 | Latex代码 | Latex笔记 | Matplotlib代码 | Matplotlib笔记 | 贡献者 |
|---|---|---|---|---|---|
| 注意力序列可视化图 | 注意力序列可视化 | 注意力序列可视化 | _ | _ | KINGDOM HEARTS 10th ANNIVERSARY |

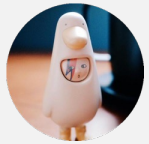
Latex

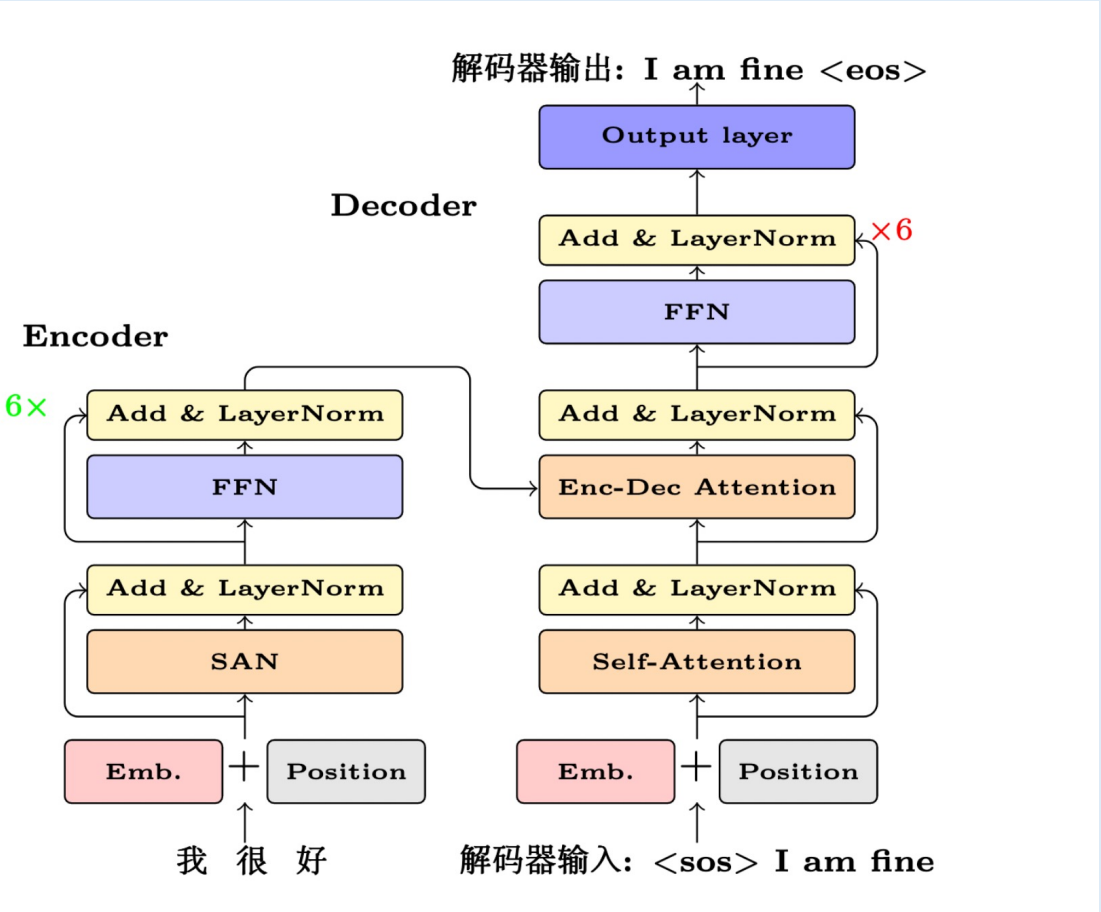
Who are the only players listed that played in 2011 ? [HEAD] player | year | round | result | opponent [ROW] 1 ray mond van bar ne ve ld | 2009 | quarter - final | won | j elle k la as en [ROW] 2 ray mond van bar ne ve ld | 2010 | 2 nd round | won | bre nd an d olan [ROW] 3 ad rian le w is | 2011 | final | won | g ary and erson

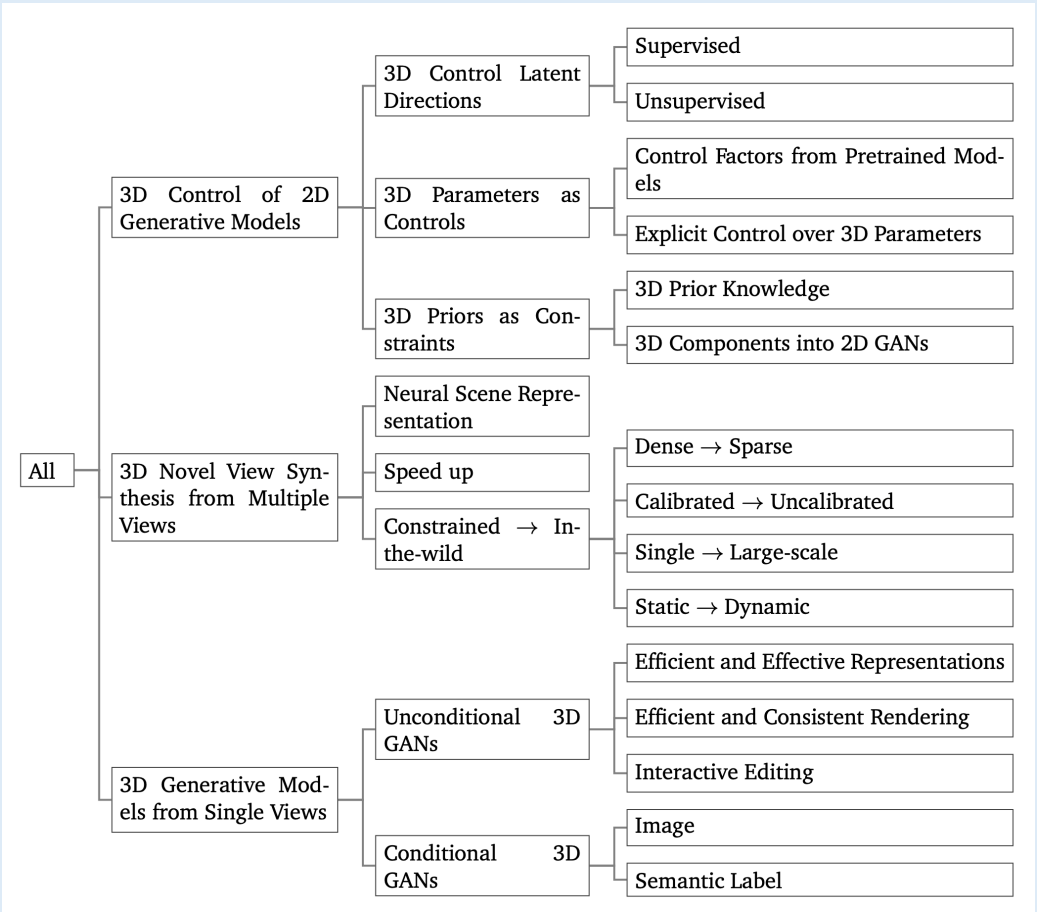
# 项目分类—模型结构图

机器学习算法与自然语言处理

| 图类别 | Latex代码 | Latex笔记 | 贡献者 |
|---|---|---|---|
| 结构图 | 结构图 | 结构图 |  |

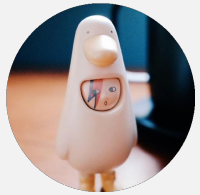
| 图类别 | Latex代码 | Latex笔记 | 贡献者 |
|---|---|---|---|
| 分类图 | 分类图 | 分类图 |  |



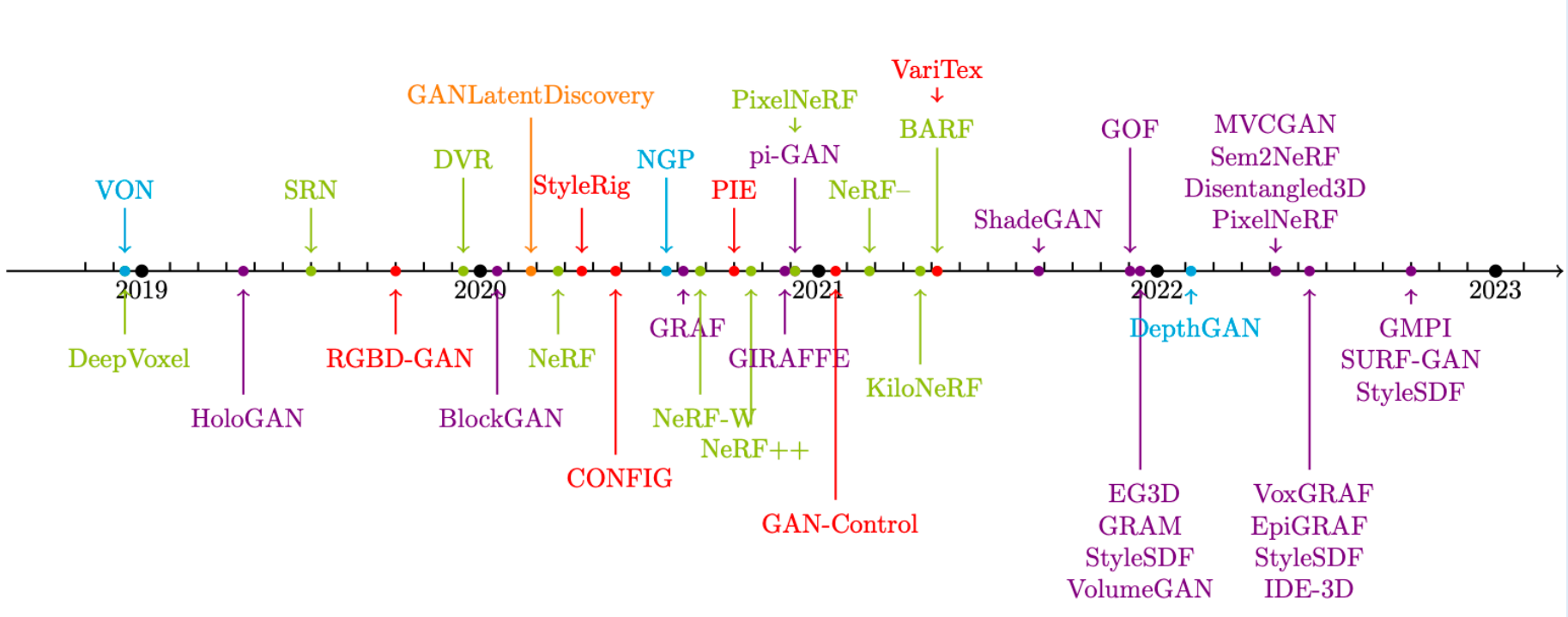



# 项目分类—时间线

机器学习算法与自然语言处理

| 图类别 | Latex代码 | Latex笔记 | Matplotlib代码 | Matplotlib笔记 | 贡献者 |
|---|---|---|---|---|---|
| 时间线 | 时间线 | 时间线 | – | – |  |

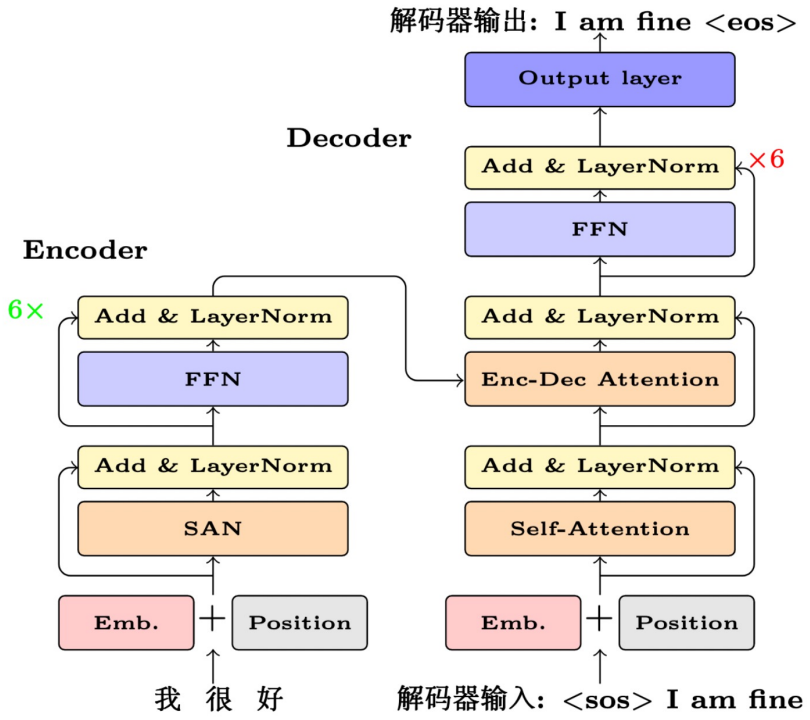
Latex

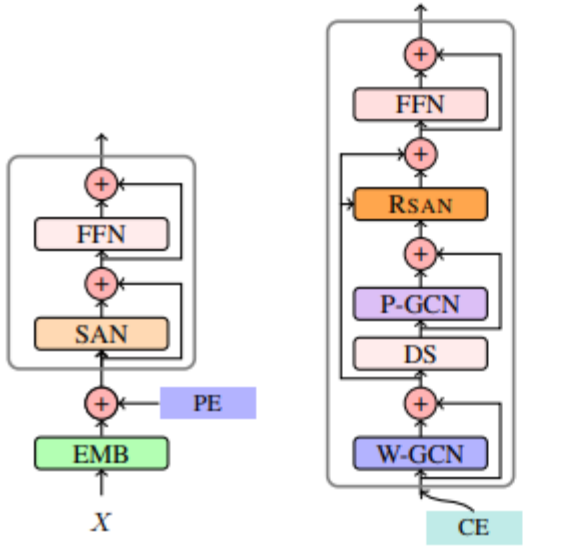


# LaTeX画图由浅入深

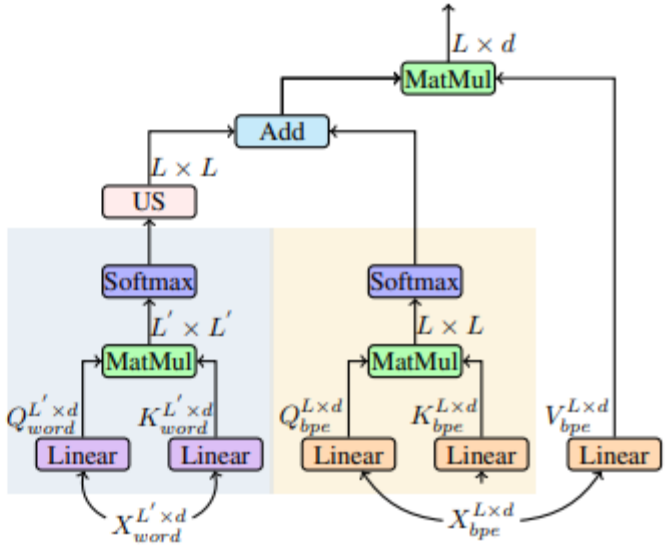


- LaTex画图主要是依赖于Tikz工具包，熟悉掌握包的各个属性有利于大家快速画出心之所想
- 最重要的还是本身对图的设计，无论是LaTeX还是PPT等都是不同的呈现形式
- 任何复杂的图像都可以拆解成一个个子图，也就是Tikz中的 \node




Standard Transformer　　Universal Multiscale Transformer




RSAN module

- \node可以定义不同样式的图形，并支持文字的填充

```
\node[[line width], [shape], [draw], fill=[color], fill opacity=[], ...,  [line shape]] (Names)(Optional) at
(positions) {textual annotations};
```

- 要掌握其具体属性

```
line width: thick、thin 等;
shape: rectangle,  circle 等;
draw: 可选项，若选择，则绘制出轮廓;
fill: 用于对物体进行颜色填充，具体颜色可参考RGB定义，采用fill=[color]进行赋值。
fill opacity: 用于设置填充颜色的透明度，取值范围为[0,1]。
[line shape]: 设置线形的形状，主要选择如下，  dotted、dashed等。
```

- 例如SAN：\tikzstyle{Sanode} = [minimum height=1.4em,minimum width=7em,inner sep=3pt,rounded corners=1.5pt,draw,fill=orange!30];

- 首先我们定义Transformer各个层对应的node

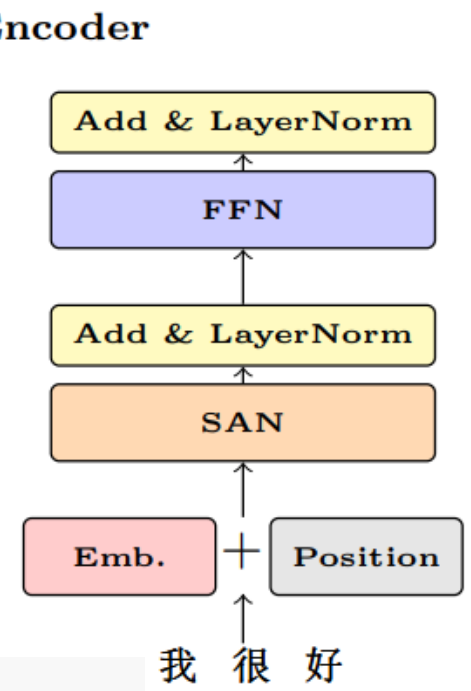
Step 1： 为模型图中的每一个组成单元定义特定的形状参数。这里主要采用的是\tikzstyle{}命令来进行设定。

```
\tikzstyle{Sanode} = [minimum height=1.4em,minimum width=7em,inner sep=3pt,rounded
corners=1.5pt,draw,fill=orange!30];
\tikzstyle{Resnode} = [minimum height=1.1em,minimum width=7em,inner sep=3pt,rounded
corners=1.5pt,draw,fill=yellow!30];
\tikzstyle{ffnnode} = [minimum height=1.4em,minimum width=7em,inner sep=3pt,rounded
corners=1.5pt,draw,fill=blue!20];
\tikzstyle{outputnode} = [minimum height=1.4em,minimum width=7em,inner sep=3pt,rounded
corners=1.5pt,draw,fill=blue!40];
\tikzstyle{inputnode} = [minimum height=1.4em,minimum width=3.5em,inner sep=3pt,rounded
corners=1.5pt,draw,fill=red!20];
\tikzstyle{posnode} = [minimum height=1.4em,minimum width=3.5em,inner sep=3pt,rounded
corners=1.5pt,draw,fill=black!10!white];
\tikzstyle{standard} = [rounded corners=3pt];
```

- 首先调用刚刚定义的node,并进行node之间的箭头连接

```
    ""对encoder端的组成部分进行布局的设计""
    \node [Sanode,anchor=west] (sa1) at (0,0) {\tiny{$\textbf{SAN}$}};
    \node [Resnode,anchor=south] (res1) at ([yshift=0.3em]sa1.north) {\tiny{$\textbf{Add \& LayerNorm}$}};
    \node [ffnnode,anchor=south] (ffn1) at ([yshift=1em]res1.north) {\tiny{$\textbf{FFN}$}};
    \node [Resnode,anchor=south] (res2) at ([yshift=0.3em]ffn1.north) {\tiny{$\textbf{Add \& LayerNorm}$}};

    \node [inputnode,anchor=north west] (input1) at ([yshift=-1em,xshift=-0.5em]sa1.south west)
{\tiny{$\textbf{Emb.}$}};
    \node [] (add) at ([yshift=-1.6em,xshift=3.5em]sa1.south west) {$+$};
    \node [posnode,anchor=north east] (pos1) at ([yshift=-1em,xshift=0.5em]sa1.south east)
{\tiny{$\textbf{Position}$}};
    \node [anchor=north] (inputs) at ([yshift=-3em]sa1.south) {\begin{CJK*}{UTF8}{gbsn}\scriptsize{$\textbf{
我\ \ 很\ \ 好}$}\end{CJK*}};
    \node [anchor=south] (encoder) at ([xshift=0.2em,yshift=0.6em]res2.north west)
{\scriptsize{\textbf{Encoder}}};
```



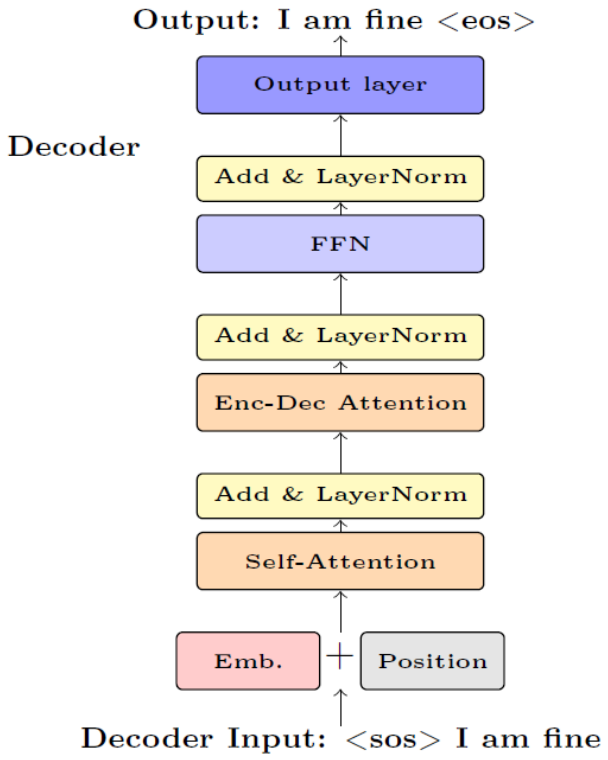

- 连接使用\draw

```
\draw [->] (sa1.north) -- (res1.south);
\draw [->] (res1.north) -- (ffn1.south);
\draw [->] (ffn1.north) -- (res2.south);
\draw [->] ([yshift=-1em]sa1.south) -- (sa1.south);
\draw [->] ([yshift=-0.3em]inputs.north) -- ([yshift=0.6em]inputs.north);
```

- 同理我们可以画出Decoder端

```
Decoder:
    \node [Sanode,anchor=west] (sa2) at ([xshift=3em]sa1.east) {\tiny{$\textbf{Self-Attention}$}};
    \node [Resnode,anchor=south] (res3) at ([yshift=0.3em]sa2.north) {\tiny{$\textbf{Add \& LayerNorm}$}};
    \node [Sanode,anchor=south] (ed1) at ([yshift=1em]res3.north) {\tiny{$\textbf{Enc-Dec Attention}$}};
    \node [Resnode,anchor=south] (res4) at ([yshift=0.3em]ed1.north) {\tiny{$\textbf{Add \& LayerNorm}$}};
    \node [ffnnode,anchor=south] (ffn2) at ([yshift=1em]res4.north) {\tiny{$\textbf{FFN}$}};
    \node [Resnode,anchor=south] (res5) at ([yshift=0.3em]ffn2.north) {\tiny{$\textbf{Add \& LayerNorm}$}};
    \node [outputnode,anchor=south] (o1) at ([yshift=1em]res5.north) {\tiny{$\textbf{Output layer}$}};
    \node [inputnode,anchor=north west] (input2) at ([yshift=-1em,xshift=-0.5em]sa2.south west)
{\tiny{$\textbf{Emb.}$}};
    \node [] (add) at ([yshift=-1.6em,xshift=3.5em]sa2.south west) {$+$};
    \node [posnode,anchor=north east] (pos2) at ([yshift=-1em,xshift=0.5em]sa2.south east)
{\tiny{$\textbf{Position}$}};
    \node [anchor=north] (outputs) at ([yshift=-3em]sa2.south) {\begin{CJK*}{UTF8}
{gbsn}\scriptsize{$\textbf{解码器输入: $<$sos$>$ I  am  fine}$}\end{CJK*}};
    \node [anchor=east] (decoder) at ([xshift=-1em,yshift=-1.5em]o1.west) {\begin{CJK*}{UTF8}
{gbsn}\scriptsize{\textbf{Decoder}}\end{CJK*}};
    \node [anchor=north] (decoutputs) at ([yshift=1.5em]o1.north) {\begin{CJK*}{UTF8}
{gbsn}\scriptsize{$\textbf{解码器输出: I  am  fine $<$eos$>$ }$}\end{CJK*}};
```

```
 \draw [->] (sa2.north) -- (res3.south);
 \draw [->] (res3.north) -- (ed1.south);
 \draw [->] (ed1.north) -- (res4.south);
 \draw [->] (res4.north) -- (ffn2.south);
 \draw [->] (ffn2.north) -- (res5.south);
 \draw [->] (res5.north) -- (o1.south);
 \draw [->] (o1.north) -- ([yshift=0.5em]o1.north);
 \draw [->] ([yshift=-1em]sa2.south) -- (sa2.south);
 \draw [->] ([yshift=-0.3em]outputs.north) -- ([yshift=0.6em]outputs.north);
```



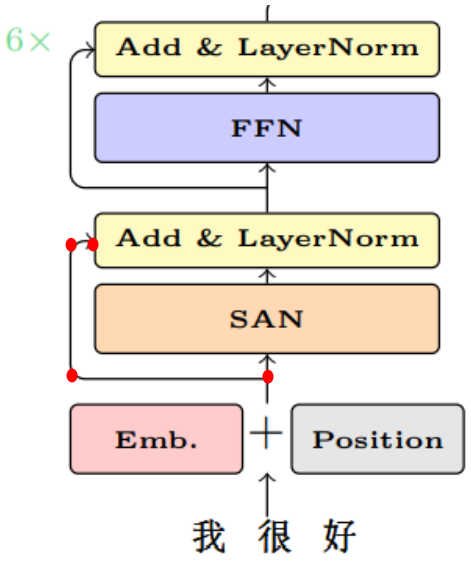
# LaTeX画图由浅入深



- 最后就是残差连接和编码-解码注意力连接

```
Encoder-Decoder:
    \draw[->,standard] ([yshift=-0.5em]sa1.south) -- ([xshift=-4em,yshift=-0.5em]sa1.south) --
([xshift=-4em,yshift=2.3em]sa1.south) -- ([xshift=-3.5em,yshift=2.3em]sa1.south);
    \draw[->,standard] ([yshift=0.5em]res1.north) -- ([xshift=-4em,yshift=0.5em]res1.north) --
([xshift=-4em,yshift=3.3em]res1.north) -- ([xshift=-3.5em,yshift=3.3em]res1.north);
```

- South（上侧）North（下侧） West （左侧）East（右侧）
- Rounded Corner所以是圆润的拐角
- 核心思想是定义了4个点的坐标，之后顺序连接，基于基点{SAN}
  - 第一个点：基点的下侧y轴偏移0.5em
  - 第二个点：基点左下侧y轴偏移0.5em，x轴偏移4.0em
  - 第三个点：基点左上侧y轴偏移2.3em，x轴偏移4.0em
  - 第四个点：基点的上侧y轴偏移2.3em，x轴偏移3.5em



# 未来计划



- 目前LaTeX和Matplotlab版本只是初级版本，后续会继续丰富内容，打造更精美的画图教程
- 支持更多的平台，例如Jupyter, Notebook等，便于大家学习访问
- 应的绘图视

# MLNLP社区介绍

MLNLP( 机器学习算法与自然语言处理 )社区是国内外知名自然语言处理社区，受众覆盖国内外NLP硕博生、高校老师以及企业研究人员。

社区的愿景是促进国内外自然语言处理，机器学习学术界、产业界和广大爱好者之间的交流和进步，特别是初学同学们的进步。

机器学习算法
与自然语言处理

# THANKS