汽车问答摘要推理

colab code:
https://drive.google.com/drive/folders/1z_h-hzQf9fXXcuE_Z6fzWERg8Ik3BR56?usp=sharing

研讨课内容：

1. 数据描述
2. 处理方法
    - 自定义切词
    - 多核批处理
    - Vacab转换
    - 词向量拼接
    - Marsk计算
    - 标示符填充
3. encoder decoder流程
4. seq2seq训练
5. loss计算
6. colab训练
7. 评估测试
8. gready search
9. beam search

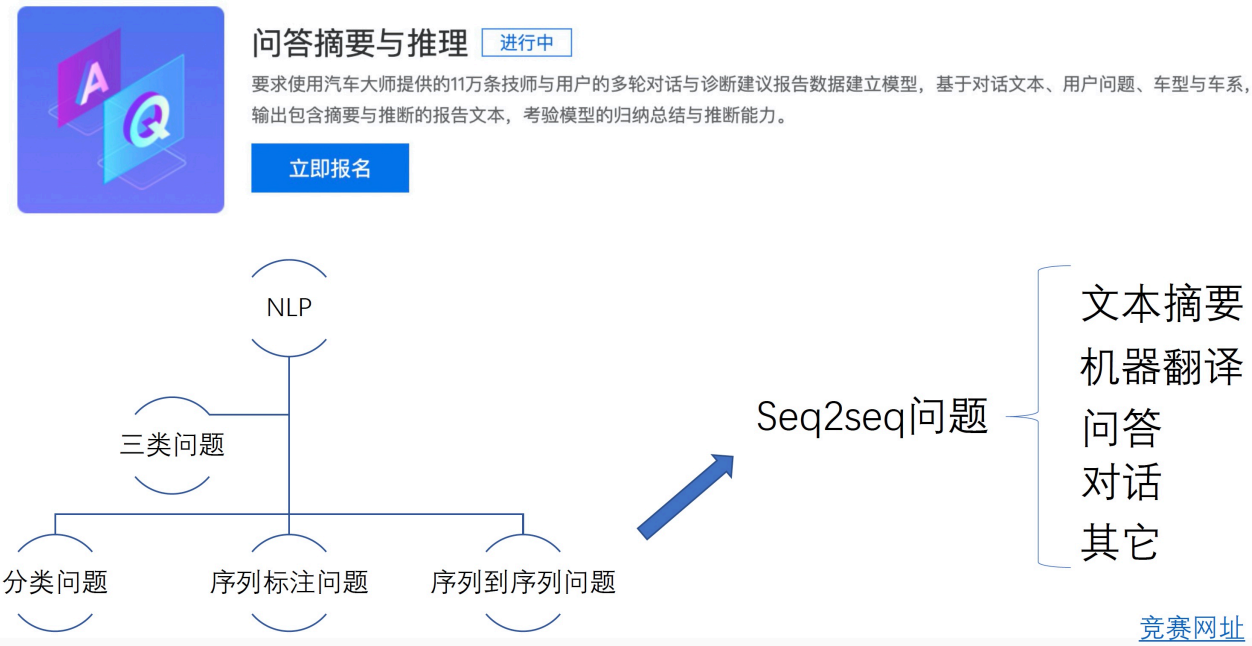
```
# homework1-week1
1.跑通preprocess.py
2. 跑通data_loader.py

3. 跑通build_w2v.py
最终通过我们的数据训练出相应的词向量
数据地址：https://aistudio.baidu.com/aistudio/competition/detail/3
```

```
# Homework-week2
1. 补全rnn_encoder.py，完成Encoder的结构
2. 补全rnn_decoder.py，完成Attention和Decoder的结构
3. 完成sequence_to_sequence.py
4. python ./seq2seq_tf2/bin/main.py看看能不能把整个模型的训练跑通哟!


```

数据介绍



竞赛网址

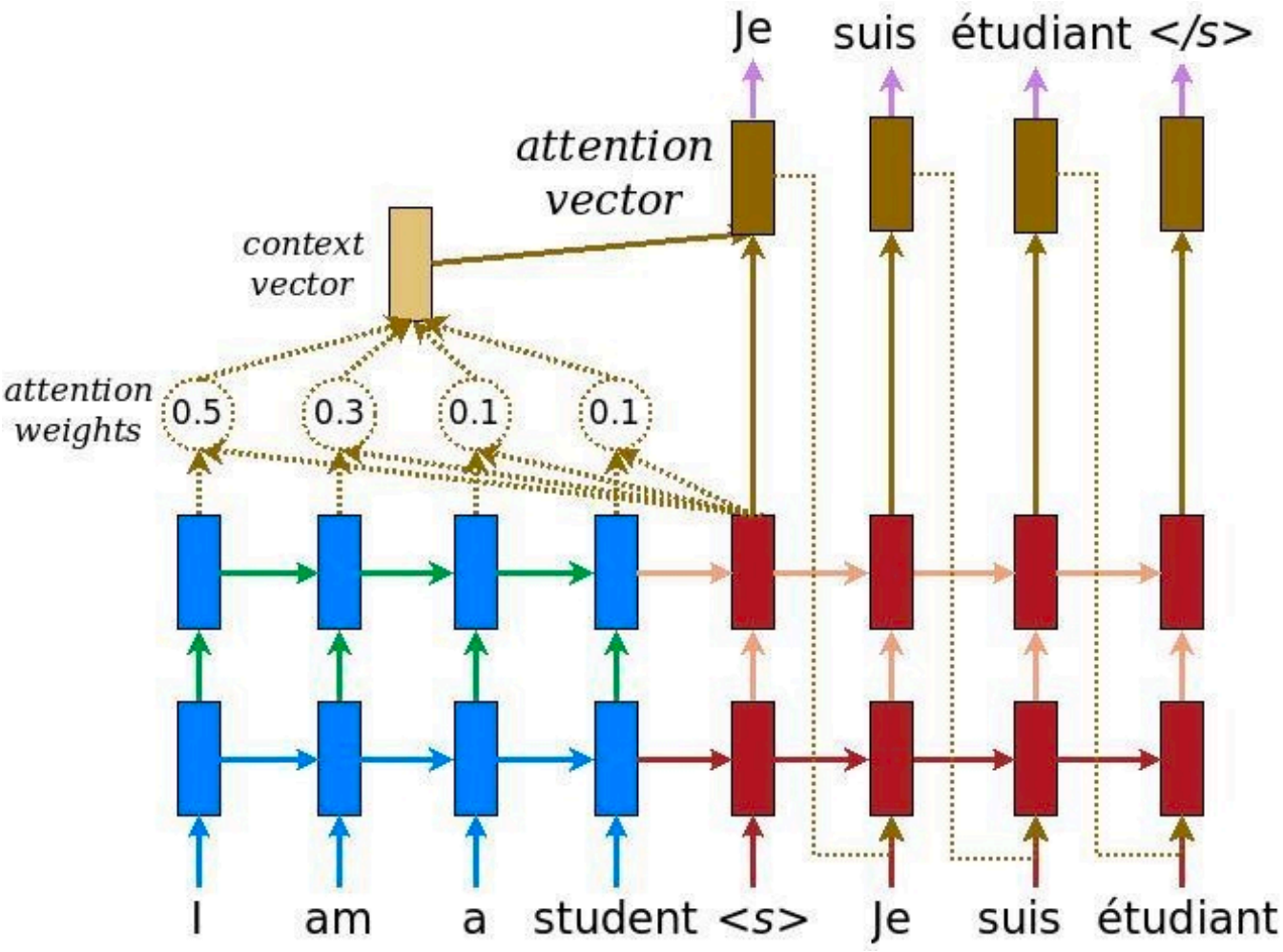
代码结构

```
代码结构

+ notebook 课件
    ....
+ result 结果保存路径
    ....
+ seq2seq_tf2 模型代码
    ....
+ utils 工具包
    + config  配置文件
    + data_loader 数据处理模块
    + multi_proc_utils 多进程数据处理
+ data  数据集
    + AutoMaster_TrainSet 拷贝数据集到该路径
```

```
    + AutoMaster_TestSet   拷贝数据集到该路径
    ....
```

Seq2Seq

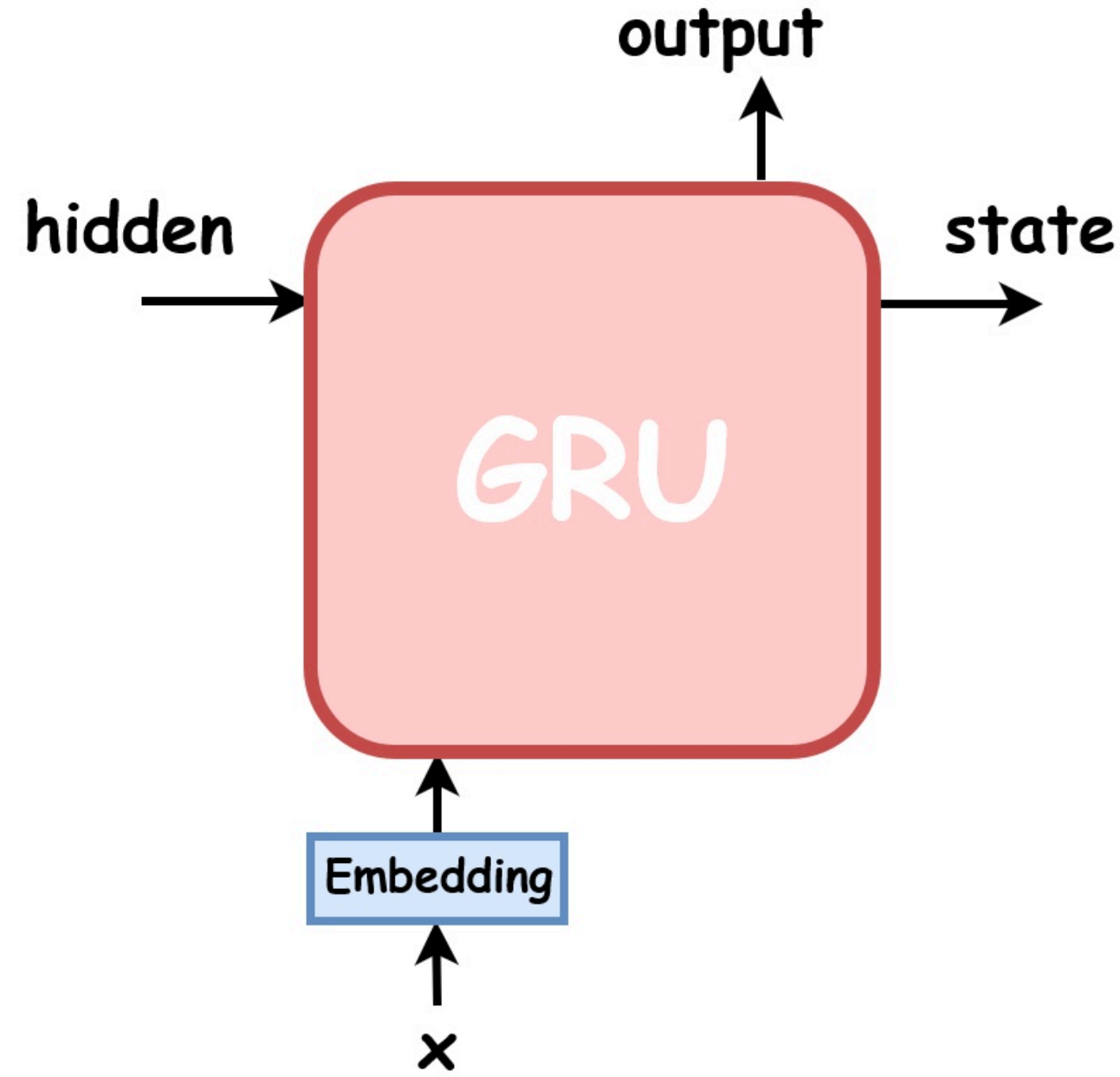


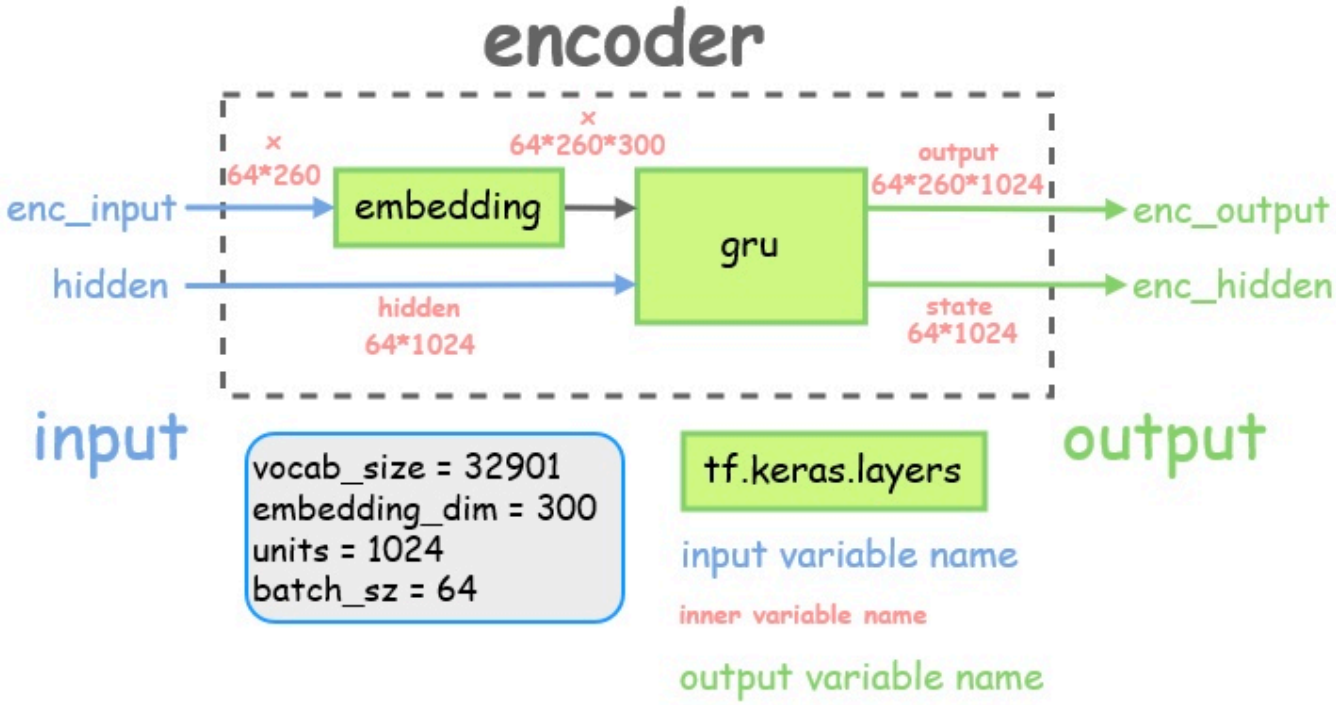
$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

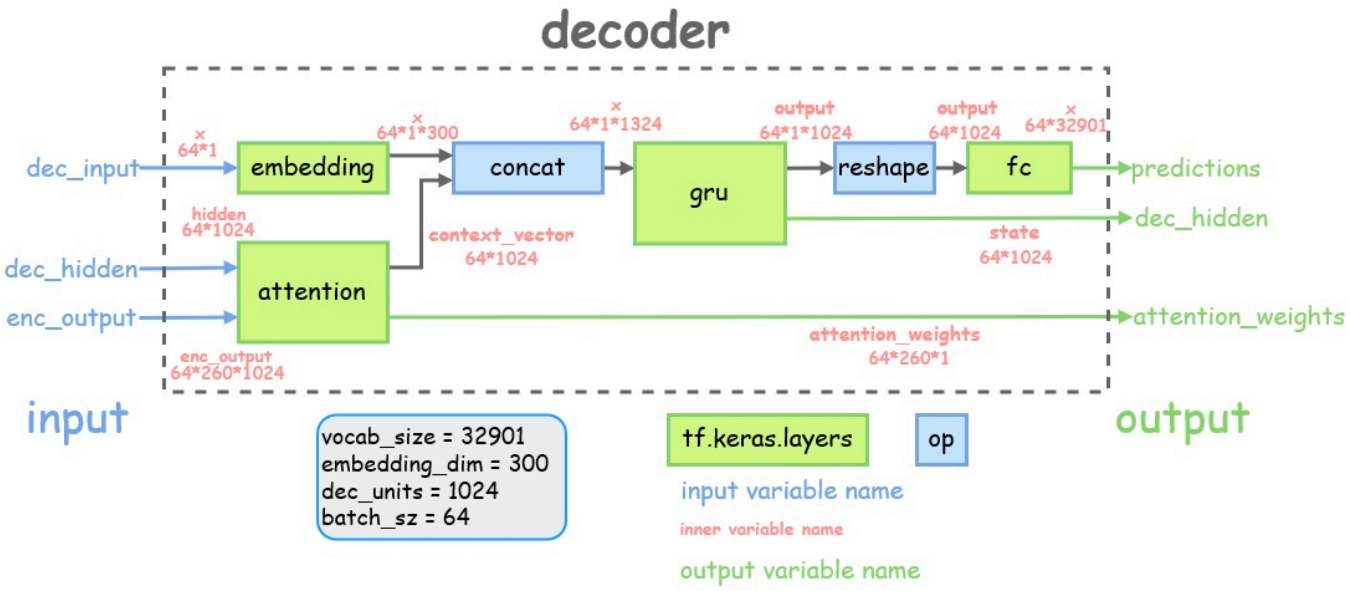
$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1} \boldsymbol{h}_t + \boldsymbol{W_2} \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases} \quad (4)$$


output
hidden
state
GRU
Embedding
x


encoder
x
64*260
x
64*260*300
output
64*260*1024
enc_input
embedding
gru
enc_output
hidden
enc_hidden
hidden
64*1024
state
64*1024
input
output
vocab_size = 32901
embedding_dim = 300
units = 1024
batch_sz = 64
tf.keras.layers
input variable name
inner variable name
output variable name

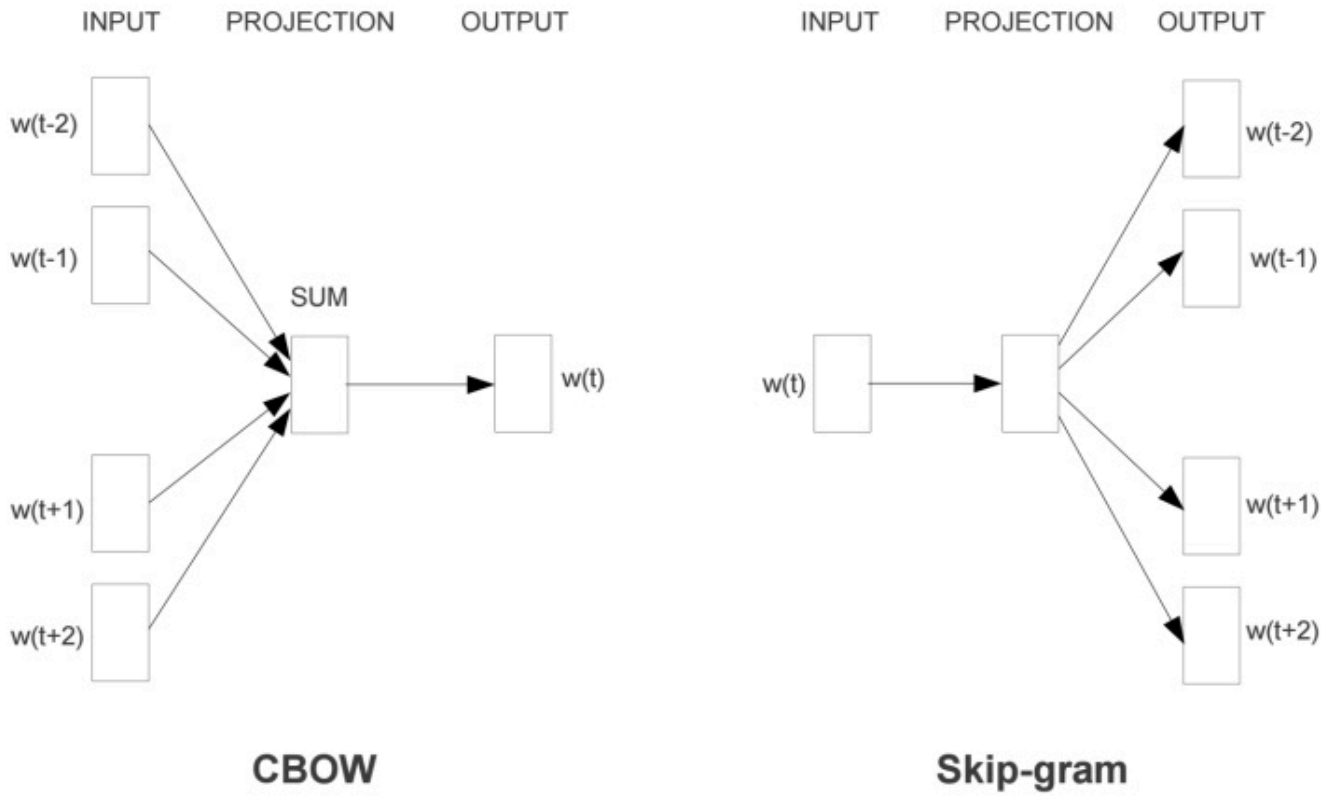


Teacher Forcing

Marsk

Loss

Word2Vec



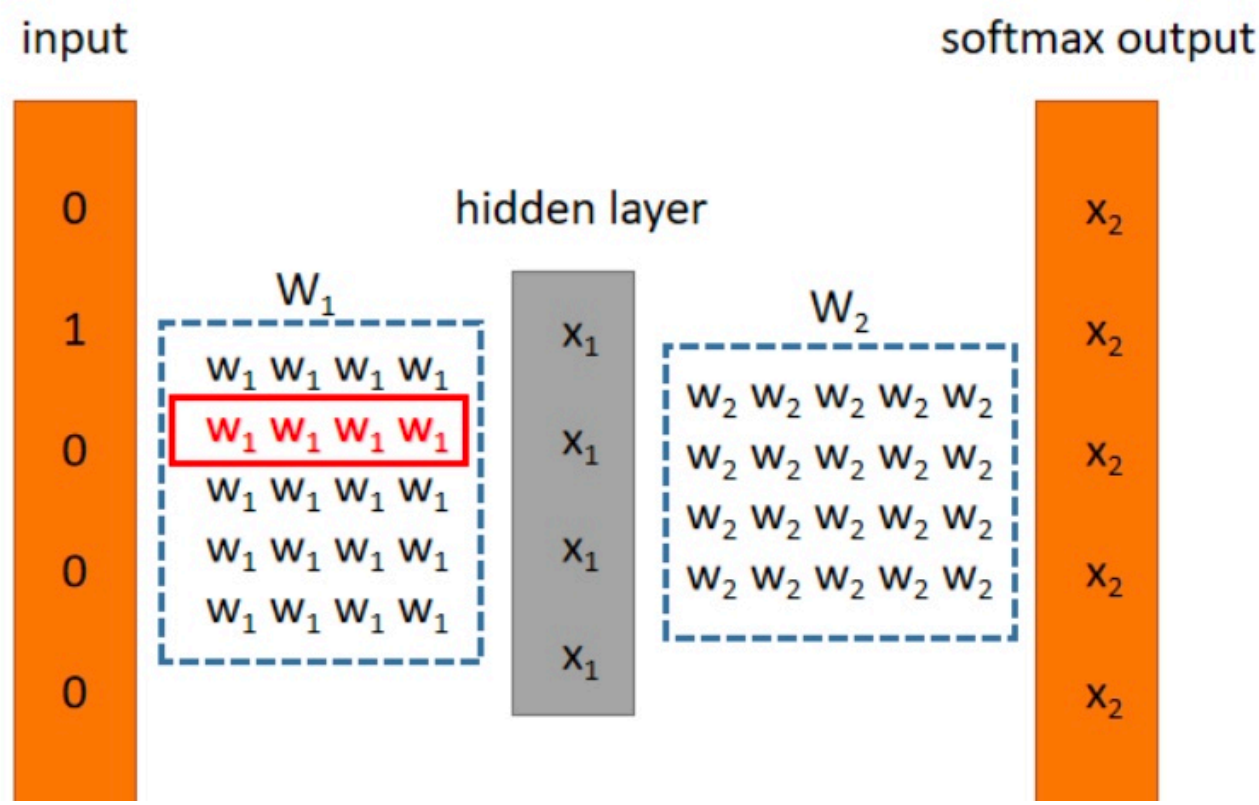
我们先来分析一下 `skip-gram` 的样本格式。 `skip-gram` 不同于 `CBOW` ， `CBOW` 是基于上下文预测当前 `input word` 。而 `skip-gram` 则是基于一个 `input word` 来预测上下文，因此一个input word会对应多个上下文。我们来举个栗子 `“[熟练掌握 java 熟悉 python shell 熟练使用 git svn]”` ，如果我们固定 `skip_window=2` 的话，那么 `熟悉` 的上下文就是 `[熟练掌握, java, python, shell]` ，如果我们的 `batch_size=1` 的话，那么实际上一个 `batch` 中有四个训练样本。

上面的分析转换为代码就是两个步骤，第一个是找到每个 `input word` 的上下文，第二个就是基于上下文构建 `batch` 。

首先是找到 `input word` 的上下文单词列表：

`Skip-Gram` 模型是通过输入词来预测上下文。因此我们要构造我们的训练样本。

对于一个给定词，离它越近的词可能与它越相关，离它越远的词越不相关，这里我们设置窗口大小为5，对于每个训练单词，我们还会在[1:5]之间随机生成一个整数R，用R作为我们最终选择 `output word` 的窗口大小。这里之所以多加了一步随机数的窗口重新选择步骤，是为了能够让模型更聚焦于当前 `input word` 的邻近词。



Greedy search

Beam search